딥러닝 예제 및 실습 I

수업 4

수업목표

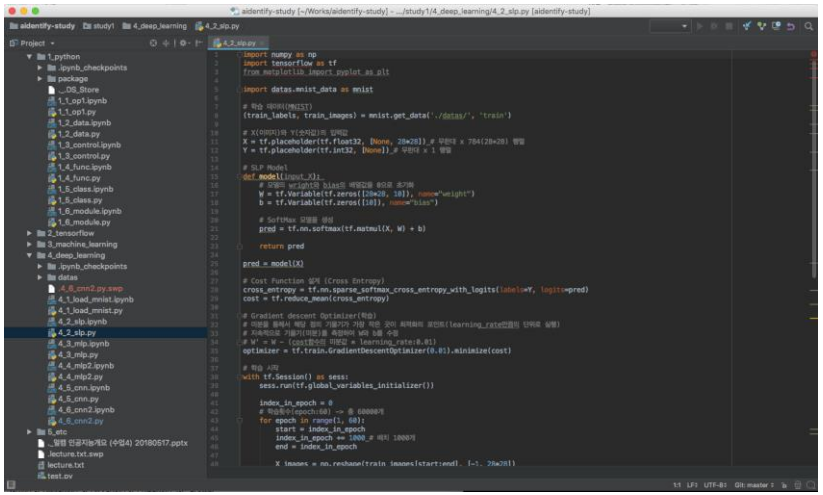
- 파이썬 언어에 대한 이해 & 실습
- 텐서플로우 실습
- 머신러닝 실습 : Linear & Logistic Regression
- 딥러닝 실습 : SLP, MLP, CNN
- 프레임워크 비교 및 분석
- 딥러닝 서비스 구현 사례: 패션 아이템, 음식 이미지 분석

파이썬에 대한 이해 & 실습

파이썬?

- 1991년, Guido van Rossum이 발표
- 고수준(High Level) 언어
- 다양한 사용자 층을 보유 (학생 ~ 전문가)
- 폭넓은 OS기기 지원과 풍부한 외부 라이브러리
- Python 버젼 (Ver. 2 vs Ver. 3)
- Google
- .pyc (bytecode)

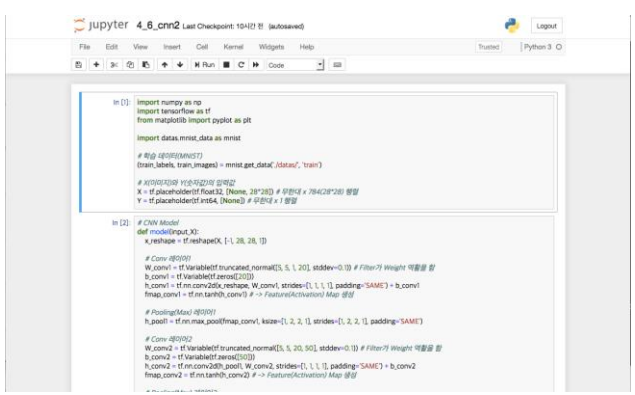
파이썬 - 실행

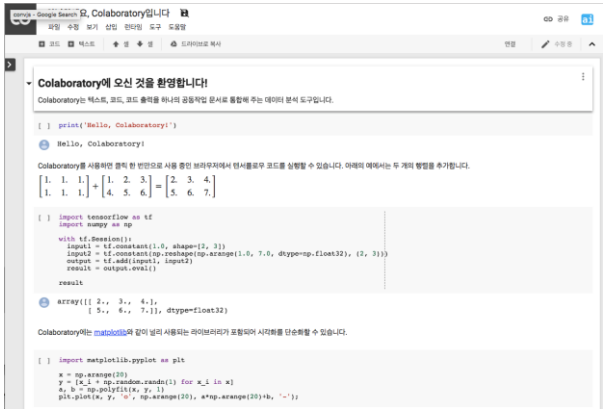


IDE (pycham)

```
erichong81@hong-yeonghun-ui-MacBook-Air:~$ python3
Python 3.6.5 (default, Mar 30 2018, 06:41:53)
[GCC 4.2.1 Compatible Apple LLVM 9.0.0 (clang-900.0.39.2)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> a = tf.constant(1)
>>> b = tf.constant(2)
>>>
>>> sess = tf.Session()
>>> sess.run(a+b)
3
>>>
```

CLI - Interactive mode





Jupyter(설치형), Colaboratory(by Google) - Web-based editor

[파일 직접 실행]
ex) python3 파일명.py

https://colab.research.google.com (구글)
https://jupyter.nims.re.kr(수리과학연구소)

파이썬 - 예제 다운로드

- 다운로드

git clone https://github.com/francoisryu/aidentify-study.git

파이썬 - 설정

- **Python3**
  - 패키지 관리자 설치
    sudo apt install python3-pip

파이썬 라이브러리 관리
- native(pip3)
- anaconda
- virtualenv

- **PyCham**
  - 설정
    기존 project 닫는다
    라이브러리 경로 확인 (python cli에서 'os' import 후 'print(os.__file__)' 명령으로 확인)
    File > Default Settings > Project Interpreter > 설정 > 경로 찾아 추가
    (/usr/bin/python3.5인지 확인)

- **Jupyter 'jupyter notebook'**
  - 필수 라이브러리 설치
    sudo pip3 install jupyter
  - 커널 설정 & 추가
    python3 -m ipykernel install --user

파이썬 기본문법 - 변수 & 자료형 & 연산

자료형	변수 = 형(자료) 데이터
문자	text = "hello World" text = 'hello World'
숫자	number = 10 octal = 0o10 (8진수) hex = 0xAB (16진수)
실수	float = 1.23 float = -1.23
참/거짓	bool = True bool = False

연산의 예) [source 1-1]

```
text = "hello" + "World"=>  "helloworld"
text*2      =>     "helloworldhelloworld"

14 / 2    => 7 (나누기)
5 % 2     => 1 (나머지)
5 // 2    => 3 (소수점 버림)

1. + 2.2     =>  3.2

2의 10승 => 2**10 => pow(2, 10)

2E5  => 2 * 지수 10의 5승 => 20000
1.23E-2  => 1.23 * 지수 10의 -2승의 곱 =>0.0123
```

파이썬은 ;와 같은 문자가 끝에 없습니다.

연산라이브러리 : math, numpy

상수는 대체적으로 대문자(TEXT), 변수는 소문자(text)를 사용합니다

파이썬 기본문법 - 데이터 자료형

자료형	변수 = 형(자료) 데이터
튜플 (고정배열)	tuple = () tuple = (1,) tuple = (1,2,(3,))
리스트 (유동배열)	list = [] list = [1,] list = [1,2,[3,]
사전형	dic = {'idx1':val1,'idx2':val2}

[source 1-2]

연산의 예)

tuple = (1, 2, '1', '2')
tuple[0] => 1
tuple + (3, '3') => (1, 2, '1', '2', 3, '3')
tuple[:2] => (1,2)

del(tuple[1])
append(tuple[1])

list = [1, 2, '1', '2']
list + [3, '3'] => [1, 2, '1', '2', 3, '3']
tuple[:2] => (1,2)

del(list[1])
append(list[1])

list + tuple (에러 : 같은 형 끼리만 연산됨)

dic = {'1':2, 1:'2'}
dic['1'] => 2
dic = {'t': [1,2,3]}

index는 0부터 시작

파이썬 기본문법 - 제어문

명칭	변수 = 형(자료) 데이터

조건	if 조건문1:
	  조건문1이 참일때 수행
	elif 조건문2:
	  조건문2가 참일때 수행
	else
	  조건이 거짓일때 수행

while순환	while 조건문: 조건이 참인 동안 수행
for순환	for 변수 in 데이터 자료형: 변수를 사용한 수행 문장

제어문의 예) [source 1-3]

```
condition = 1

if condition == 1:
    print("1")
elif condition == 2:
    print("2")
else:
    print("else")

while condition <= 10:
    print(condition)
    condition = condition + 1

conditions = [1,2,3]
for i in conditions:
    print(i)
```

indent로 구문의 모양을 강제

if와 비슷한 switch는 제공되지 않습니다.

파이썬 기본문법 - 함수, 클래스, 모듈

[source 1-4]

함수 사용 예)

```
def add(a, b):
    return a+b

print(add(1, 2))
=> 3
```

[source 1-5]

클래스 사용 예)

```
class operation:
    def __init__(self):
        self.result = 0
    def add(self, a, b):
        self.result = a+b
    def div(self, a, b):
        self.result = a/b
    def get(self):
        return self.result

op = operation()
op.add(1, 2)
print(op.get())
=> 3
```

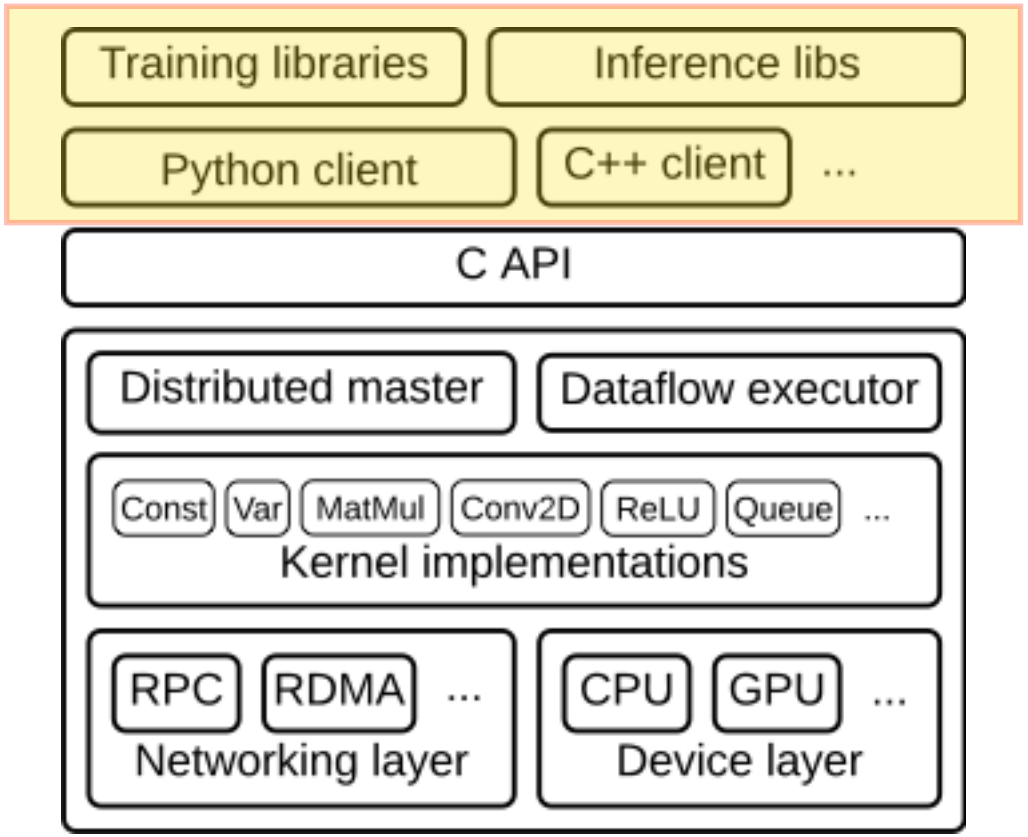
[source 1-6]

모듈 사용 예)

```
import package
package.mod_func()
예) import tensorflow
    tensorflow.Session()

from package import module
module.func()

import package as module
module.func()
예) import tensorflow as tf
    tf.Session()

from package import *
func()    => __init__.py 에서 정의 필요
```

클래스에 접근제한자 public, private, protected 제공안함

파이썬 & GPU



텐서플로우 아키텍처 구조
https://www.tensorflow.org/extend/architecture

CPU vs GPU

OpenCL(Computing Language) 호환

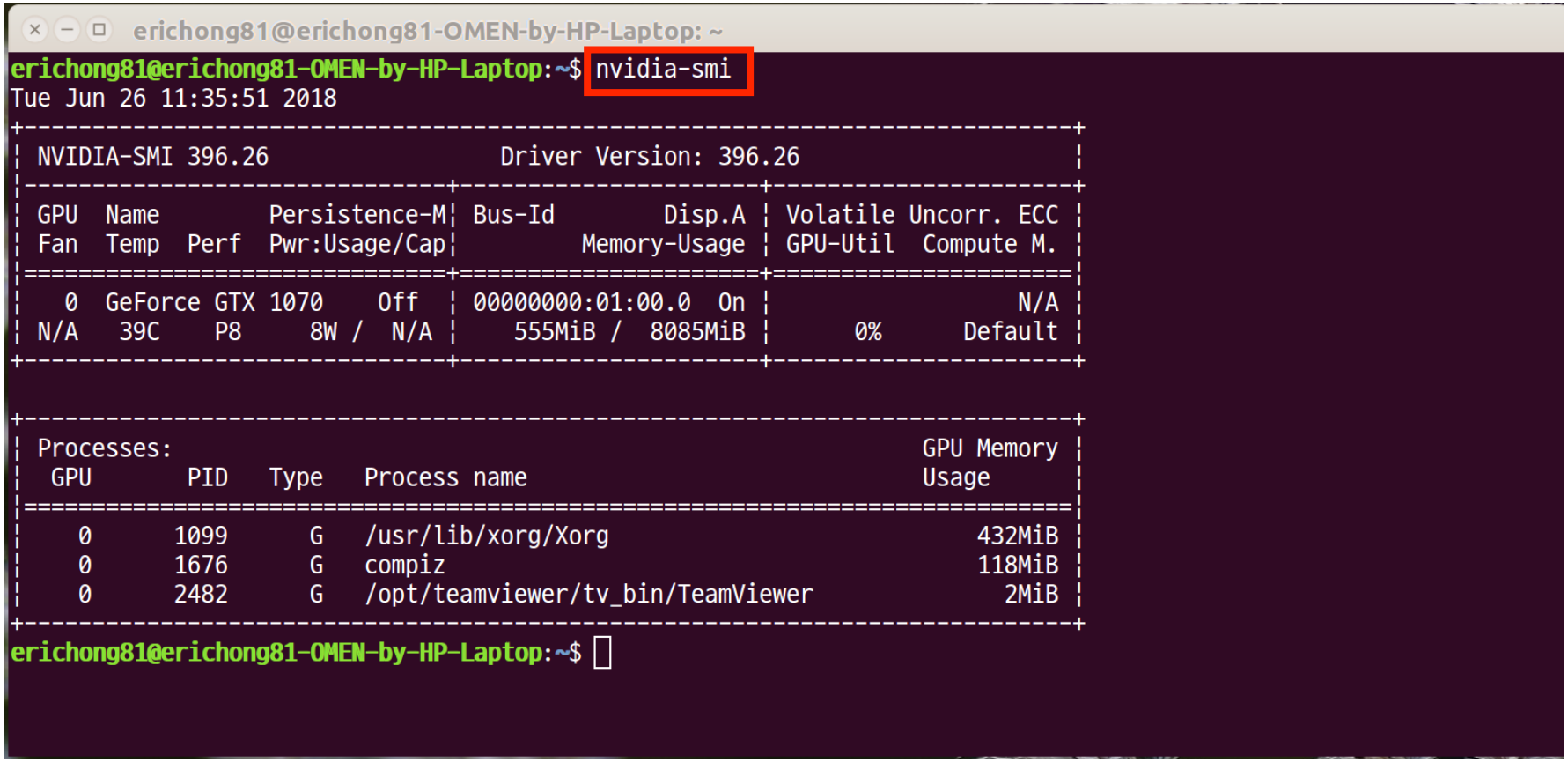
CUDA(Compute Unified Device Architecture)

GPGPU(General-puporse GPU)

Nvidia VGA Driver / CUDA Toolkit

파이썬 & GPU

nvidia-smi : **Nvidia** System Management Interface



텐서플로우 실습

설치

현재 Tensorflow 버전은 1.8

지원 OS

- 맥OS 10.12.6 (시에라) 이후 버전
- 우분투 16.04 이후 버전
- 윈도우 7 이후 버전

GPU 버전의 경우

- GPU Driver
- CUDA(Compute Unified Device Architecture) Toolkit 9.0
- CuDNN(CUDA Deep Neural Network Library) SDK v7

지원 언어

- Python
- Java
- Go
- C

웹 버전 : https://colab.research.google.com

텐서플로우 - 설정

- 텐서플로우 & 관련 라이브러리 설치

  sudo pip3 install tensorflow matplotlib numpy
  sudo apt install python3-tk

- 콘솔에서 import 되는지 확인

```
[코드 실행]
import tensorflow
tensorflow.Session()
```

Your CPU supports instruct ions that this TensorFlow binary was not compiled to use: AVX AVX2
=> 위와 같은 경고가 난다면, '**export TF_CPP_MIN_LOG_LEVEL=2**' 실행하고, ~/.bashrc에 추가해준다

Hello World

[source 2-1]

```
cd aidentify-study/study1/

jupyter notebook



import tensorflow as tf

# 상수를 생성하여 기본 그래프에 추가 (상수 명령값을 출력하는 텐서를 하나 만든다)
hello = tf.constant('Hello, World!')

# 세션 시작
sess = tf.Session()

# 명령 실행
print(sess.run(hello))



sess.close()
```

[source 2-2]

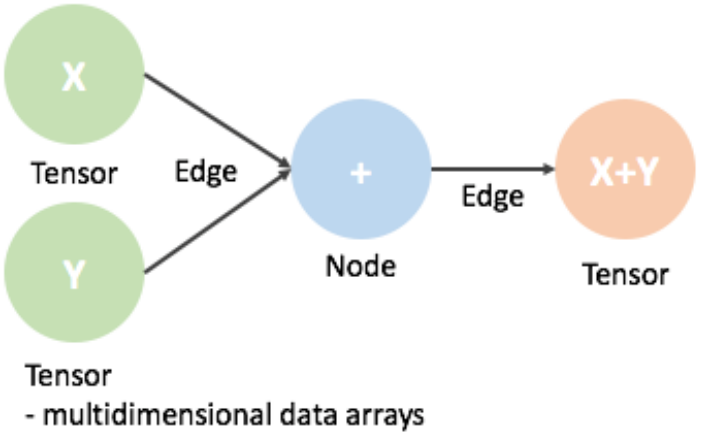
[개선]

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

....print(hello.eval())
```

계산 예제

[source 2-3]

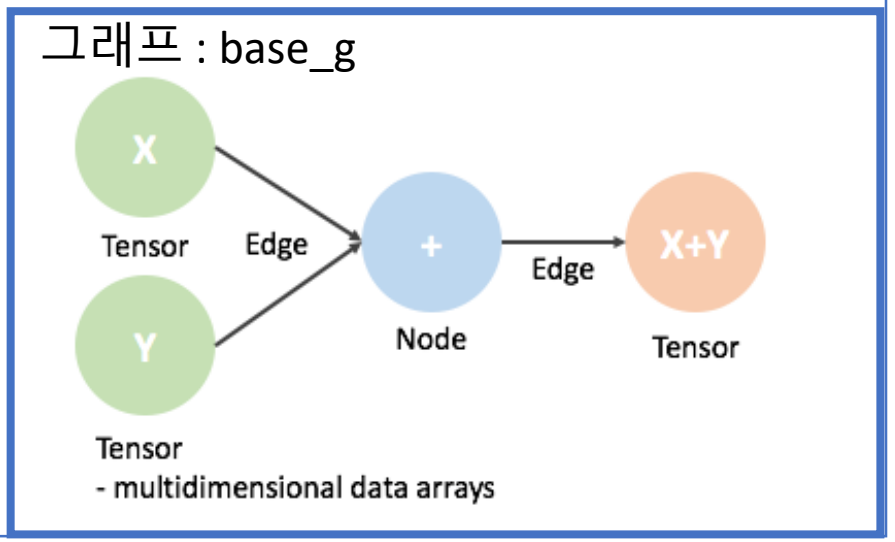
```
import tensorflow as tf

# 상수를 생성하여 기본 그래프에 추가 (상수 명령값을 출력하는 텐서를 하나 만든다)
x = tf.constant(10)
y = tf.constant(5)

# add 노드, 파이썬의 기본 '+ 연산자'에 tf.add(x, y)를 재정의 한 것
x_y = x + y


# 세션 시작
with tf.Session() as sess:
    # 그래프 계산 명령 실행
    print(x_y.eval())
```



계산 예제 (graph 추가)

[source 2-4]

```
import tensorflow as tf

#그래프 생성
g = tf.Graph()
with g.as_default() as base_g:
    with base_g.name_scope("g1") as scope:
        # 상수를 생성하여 정의된 그래프에 추가 (상수 명령값을 출력하는 텐서를 하나 만든다)
        x = tf.constant(10, name="X")
        y = tf.constant(5, name="Y")
        x_y = tf.add(x, y, name="ADD")

# 세션 시작
with tf.Session(graph=g) as sess:
    # 그래프 계산 명령 실행
    print(sess.run(x_y))
```



Graph와 세션

- 그래프를 정의하고, 세션으로 한번에 실행
- 그래프는 정의하지 않으면 기본으로 1개 주어짐
- 그래프 내에 텐서가 배치, 세션 실행 시, 연결된 노드가 모두 실행 됨
- 세션에 실행 환경에 대한 설정 가능 (원격 / 로컬)
- ‘Session.graph’로 그래프 접근 가능

Data 처리

상수(Constant)

```
x = tf.constant(10, name="x")
```

변수(Variable) - 세션 이후, 초기화 필요함

```
w = tf.Variable(tf.random_normal([3, 2]), name="W")
w = tf.Variable(2, name="W")
b = tf.Variable(w.initialized_value() + 3, name="B")
```

```
with tf.Session() as sess:                                  [source 2-5]
...     sess.run(w.initializer)
...     print(sess.run(w))
```

전체 초기화

```
init = tf.initialize_all_variables()
tf.Session() as sess:
...     sess.run(init)
...     print(sess.run(w))                                  [source 2-6]
```

치환형(Placeholder)

```
x = tf.placeholder(tf.int32, shape=(2, 2))                  [source 2-7]
y = tf.matmul(x, x)  # 행렬 곱 (x * x)

with tf.Session() as sess:
  #print(sess.run(y))  # placeholder 입력 값 할당 전에 출력하면 에러 발생

  matrix = [[1, 2],
            [1, 2]]
  print(sess.run(y, feed_dict={x: matrix}))  # 성공적으로 출력
```

데이터 저장 / 로드

저장

[source 2-8]

```
import tensorflow as tf

x = tf.Variable(2)

init_op = tf.initialize_all_variables()

saver = tf.train.Saver()

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

save_path = saver.save(sess, "/tmp/model.ckpt")
```

로드

[source 2-9]

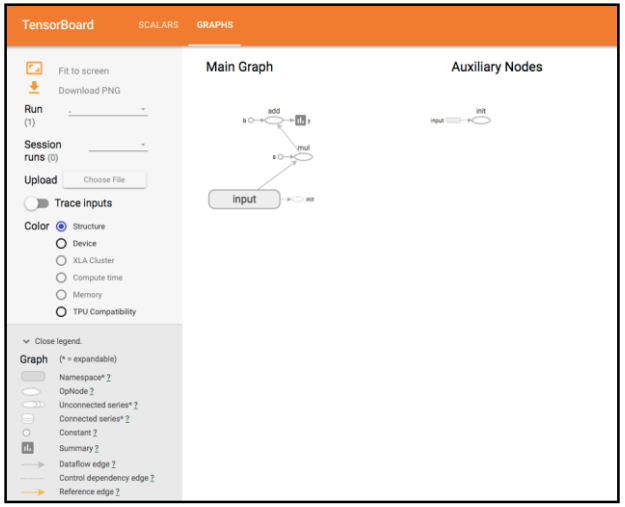
```
import tensorflow as tf

x = tf.Variable(0) # 담을 데이터 변수

saver = tf.train.Saver()

sess = tf.Session()

saver.restore(sess, "/tmp/model.ckpt") # 로드

print(sess.run(x)) # 로드된 값을 출력해본다
```

TensorBoard

텐서플로우에서 데이터의 흐름을 시각화 해주는 툴 (텐서플로우에서 기본으로 제공)

[source 2-10]

```
with tf.name_scope('op'):
    a = tf.constant(2, name="a")
    b = tf.constant(1, name="b")

with tf.name_scope('input'):
    x = tf.placeholder(tf.int32, name="X")
    y = tf.Variable(0, tf.int32, name="Y")

y = a * x + b

tf.summary.scalar('x', x)
tf.summary.scalar('y', y)

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

    # TensorBoard 그래프용 로그 기록
    writer = tf.summary.FileWriter("/tmp/tensorboard", sess.graph)

    # summary 정보를 파일로 저장
    merged = tf.summary.merge_all()

    for i in range(10):
        summary, t = sess.run([merged, y], feed_dict={x: i})
        writer.add_summary(summary, i)

    writer.close()
```



머신러닝 실습

- Linear Regression

- Logistic Regression

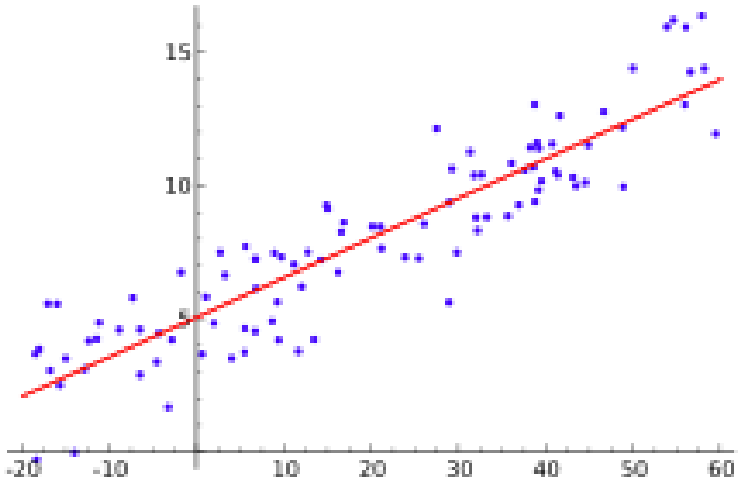
텐서플로우 틀 잡기

1. 데이터 준비(배열 값, 이미지..)
2. 입력 값 셋팅(데이터가 많은 경우 placeholders 사용)
3. 모델 셋업 (linear, logistics, multilayer…)
4. cost & 최적화 설계
5. 세션 시작(with 변수 초기화)
6. 학습 (배치와 루프 설정 -> optimizer실행)
7. 테스트(임의의 샘플 데이터로 cost를 구함)
8. 학습과 테스트 비교(필요하면 시각화 함)
9. 값 예측

위 내용은 Single 그래프의 경우로, multi-graph의 경우에는 구조를 나누어서 세션에서 결과를 조합하기도 함

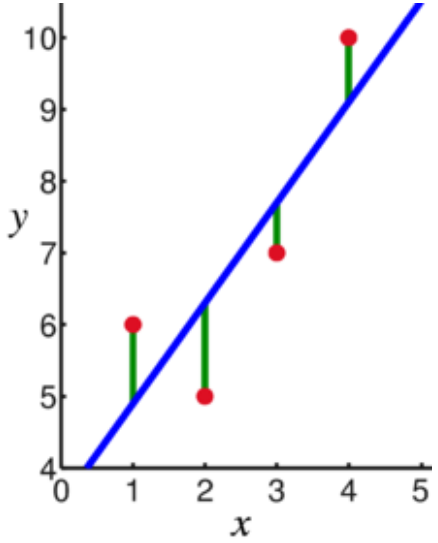
```
result = sess.run(sess.graph.get_tensor_by_name( scope명 ))
```

Linear Regression



$$Y = f(X) + \epsilon.$$

Y = Weight * X + Bias



최소제곱법 $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$

실측(데이터)과 예상(함수) 간의 거리를 제곱하여 평균한다

[source 3-1] : linear Regression 기본

[source 3-2] : 파라메터 추가

[source 3-3] : 데이터 구조 변경(행렬) & 데이터 파일읽기

Linear Regression

[source 3-1]

```
# 1) 데이터 준비
train_X = [2.52, 4.22, 9.65, 4.22, 7.6, 4.2, 2.2, 3.5]
train_Y = [3.28, 7.22, 17.44, 8.22, 13.11, 9.2, 5.3, 6.3]

# 2) 입력 값 셋팅
X = tf.placeholder("float")
Y = tf.placeholder("float")

W = tf.Variable(0., name="weight")
b = tf.Variable(0., name="bias")

# 2) 모델 셋업(linear)
pred = W*X + b

# Cost & 최적화 설계
cost = tf.reduce_mean(tf.square(pred-Y))
optimizer = tf.train.GradientDescentOptimizer(0.01).minimize(cost)
```

Linear Regression

[source 3-1]

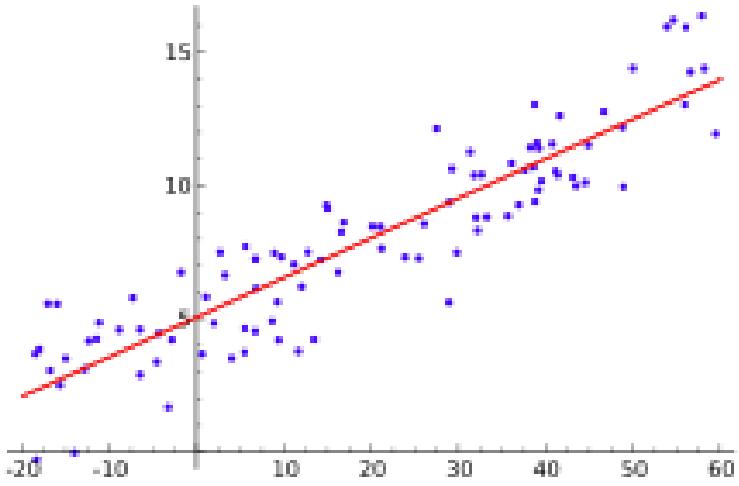
```
# 5) 세션 시작
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())

    # 6) 학습 (epoch:3000)
    for epoch in range(3000):
        sess.run(optimizer, feed_dict={X: train_X, Y:train_Y})

                                    # cost계산
        training_cost = sess.run(cost, feed_dict={X: train_X, Y:train_Y})
        training_W = sess.run(W)
        training_b = sess.run(b)
        print(epoch, training_cost, [training_W, training_b])

    print("학습완료! (cost : " + str(training_cost) + ")")
```

Linear Regression

[source 3-1]

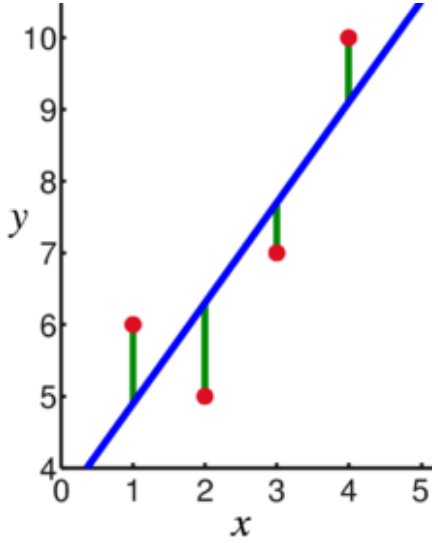
```
# 7) 테스트
test_X = [6.83, 4.668, 8.9, 7.91, 5.7, 8.7, 3.1, 2.1]
test_Y = [1.84, 2.273, 3.2, 2.831, 2.92, 3.24, 1.35, 1.03]

testing_cost = sess.run(cost, feed_dict={X: test_X, Y: test_Y})
print("테스트 완료! (cost : " + str(testing_cost) + ")")

# 8) 학습과 테스트 cost비교(절대값)
print("테스트와 학습의 cost차이 : ", abs(training_cost - testing_cost))
# [화면에 찍어보는 기능 추가 가능]

# 9) 값 예측
print("x가 3.3일때 : " + str(sess.run(pred, feed_dict={X: 3.3})))
```

Linear Regression - 다중 인자



$$Y = f(X) + \epsilon.$$

Y = Weight * X + Bias



최소제곱법 $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$

실측(데이터)과 예상(함수) 간의 거리를 제곱하여 평균한다

[source 3-1] : linear Regression 기본

[source 3-2] : 파라메터 추가

[source 3-3] : 데이터 구조 변경(행렬) & 데이터 파일읽기

Linear Regression - 다중 인자

[source 3-2]

```
# 1) 데이터 준비
train_X = [3.3, 4.4, 5.5, 6.71, 6.93, 4.168, 9.779]
train_X2 = [2.3, 3.4, 4.5, 5.71, 5.93, 3.168, 8.779]
train_Y = [1.7, 2.76, 2.09, 3.19, 1.694, 1.573, 3.366]

# 2) 입력 값 셋팅
X = tf.placeholder("float")
X2 = tf.placeholder("float")
Y = tf.placeholder("float")

W = tf.Variable(0., name="weight")
W2 = tf.Variable(0., name="weight")
b = tf.Variable(0., name="bias")

# 2) 모델 셋업(linear)
pred = W*X + W2*X2 + b

# Cost & 최적화 설계
cost = tf.reduce_mean(tf.square(pred-Y))
optimizer = tf.train.GradientDescentOptimizer(0.01).minimize(cost)
```

Linear Regression - 다중 인자

[source 3-2]

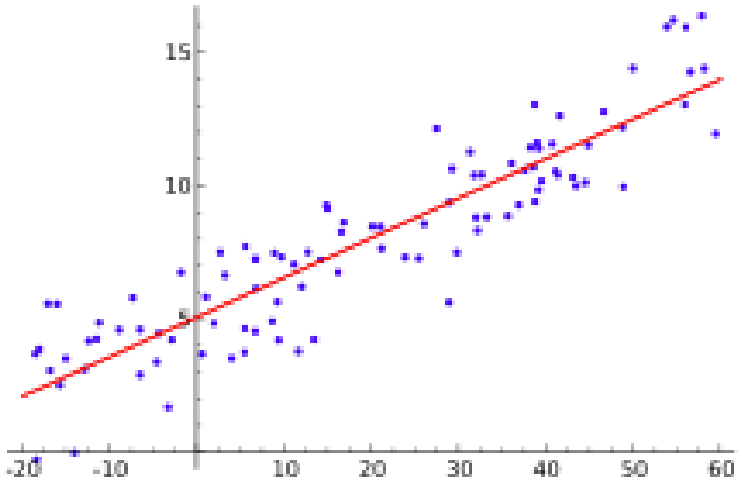
```
# 5) 세션 시작
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())

    # 6) 학습 (epoch:3000)
    for epoch in range(3000):
        sess.run(optimizer, feed_dict={X: train_X, X2: train_X2, Y:train_Y})

                                        # cost계산
        training_cost = sess.run(cost, feed_dict={X: train_X, X2: train_X2, Y:train_Y})
        training_W = sess.run(W)
                                        training_W2 = sess.run(W2)
        training_b = sess.run(b)
        print(epoch, training_cost, [training_W, training_W2, training_b])

    print("학습완료! (cost : " + str(training_cost) + ")")
```

Linear Regression - 다중 인자

[source 3-2]

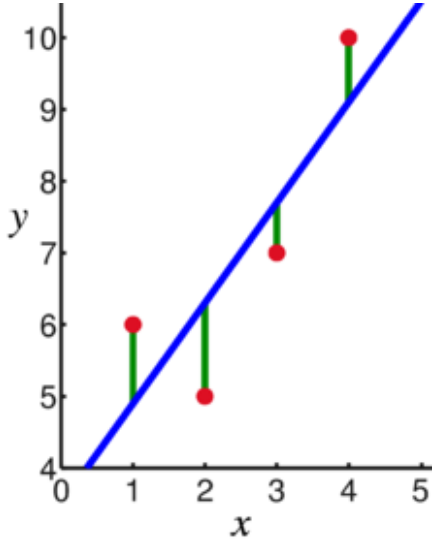
```
# 7) 테스트
test_X = [6.83, 4.668, 8.9, 7.91, 5.7, 8.7, 3.1, 2.1]
test_X2 = [5.83, 3.668, 7.9, 6.91, 4.7, 7.7, 2.1, 1.1]
test_Y = [1.84, 2.273, 3.2, 2.831, 2.92, 3.24, 1.35, 1.03]

testing_cost = sess.run(cost, feed_dict={X: test_X, X2: test_X2, Y: test_Y})
print("테스트 완료! (cost : " + str(testing_cost) + ")")

# 8) 학습과 테스트 cost비교(절대값)
print("테스트와 학습의 cost차이 : ", abs(training_cost - testing_cost))
# [화면에 찍어보는 기능 추가 가능]

# 9) 값 예측
print("x가 3.3일때 : " + str(sess.run(pred, feed_dict={X: 3.3, X2:3.6})))
```

Linear Regression



$$Y = f(X) + \epsilon.$$

Y = Weight * X + Bias



최소제곱법 $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$

실측(데이터)과 예상(함수) 간의 거리를 제곱하여 평균한다

[source 3-1] : linear Regression 기본

[source 3-2] : 파라메터 추가

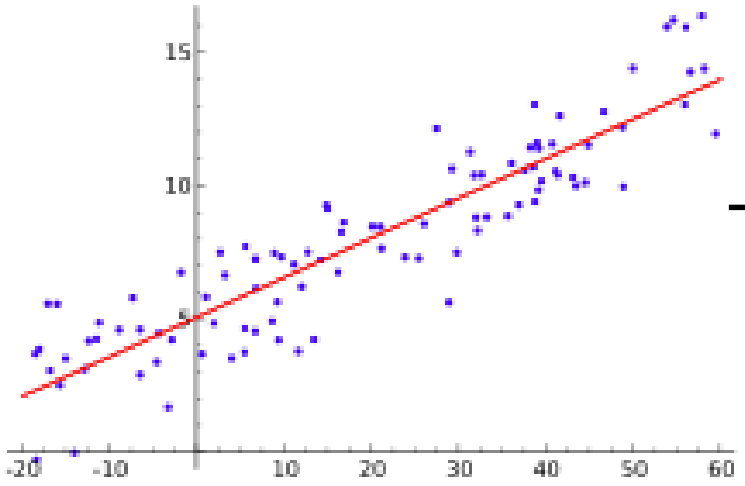
[source 3-3] : 데이터 구조 변경(행렬) & 데이터 파일읽기

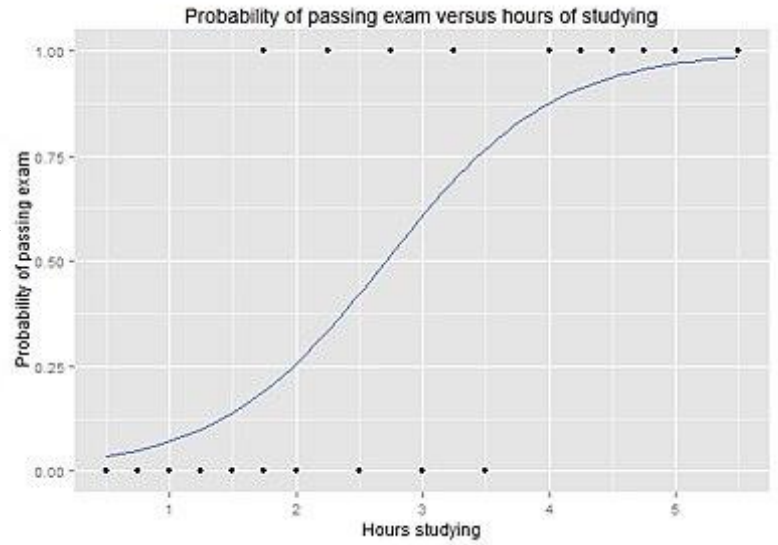
Linear Regression - matrix

[source 3-3]

```
# 1) 데이터 준비
# train_X = np.loadtxt('./datas/3_3_linear_matrix.csv', unpack=True, dtype='float')
train_X = [[3.3, 4.4, 5.5, 6.71, 6.93, 4.168, 9.779],
           [2.3, 3.4, 4.5, 5.71, 5.93, 3.168, 8.779],
           [1.,1.,1.,1.,1.,1.,1.]]
train_Y = [1.7, 2.76, 2.09, 3.19, 1.694, 1.573, 3.366]

# 2) 입력 값 셋팅
X = tf.placeholder("float")
Y = tf.placeholder("float")

W = tf.Variable(tf.zeros([1, 3], dtype=tf.float32), name="weight")

# 2) 모델 셋업(linear)
pred = tf.matmul(W, X)

# Cost & 최적화 설계
cost = tf.reduce_mean(tf.square(pred-Y))
optimizer = tf.train.GradientDescentOptimizer(0.01).minimize(cost)
```

Logistic Regression





sigmoid function (=logistic functon) $g(x) = \frac{1}{1+e^{-x}}$

Cross Entropy

$$H(p,q) = -\sum_{x} p(x)\ \log q(x).$$

$$Cost(y,h) = \begin{cases} -log(h), & \text{if } y = 1 \\ -log(1-h), & \text{if } y = 0 \end{cases}$$

$$Cost(y,h) = -y\log(h) - (1-y)\log(1-h)$$

[source 3-4] : 모델과 cost function 변경

Logistic Regression

[source 3-4]

```
# 1) 데이터 준비
train_X = [3.3, 4.4, 5.5, 6.71, 6.93, 4.168, 9.779]
train_Y = [0, 0, 1, 1, 1, 0, 1]

# 2) 입력 값 셋팅
X = tf.placeholder("float")
Y = tf.placeholder("float")

W = tf.Variable(0., name="weight")
b = tf.Variable(0., name="bias")

# 2) 모델 셋업(linear)
#pred = 1. / (1. + tf.exp(-(W*X + b)))
pred = tf.sigmoid(W*X + b)

# Cost & 최적화 설계(Cross Entropy)
cost = tf.reduce_mean(tf.reduce_sum(-Y * tf.log(pred) - (1 - Y) * tf.log(1 - pred)))
optimizer = tf.train.GradientDescentOptimizer(0.01).minimize(cost)
```

테스트 부분도 동일하게 수정

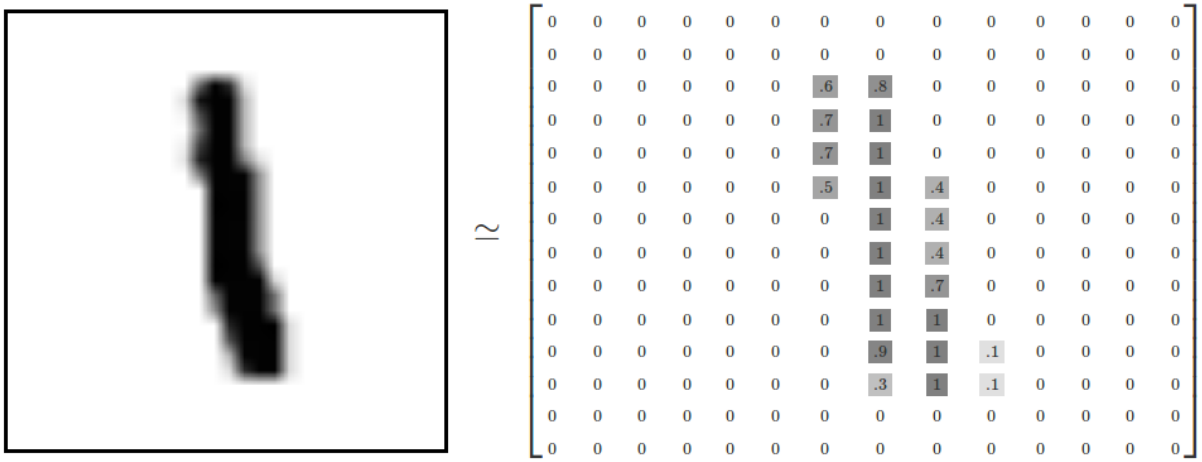
딥러닝 실습 (with MNIST)

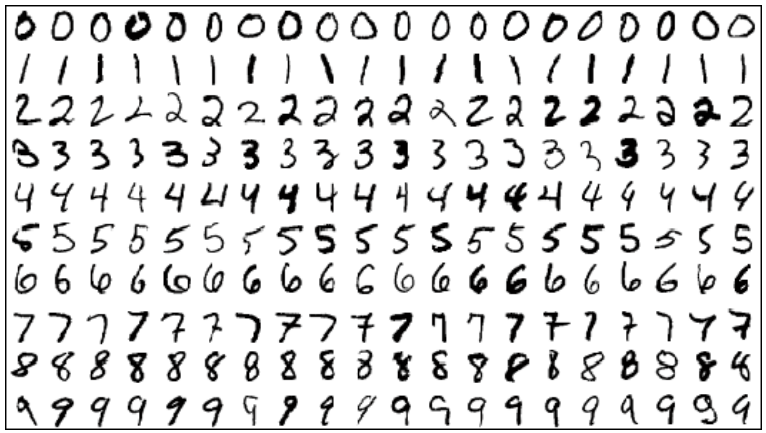
- SLP (Single-Layer Perceptron)
- MLP (Multi-Layer Perceptron)
- CNN (Convolutional Neural Network)

MNIST: 소개

필기체 숫자 인식에 주로 사용되는 데이터

http://yann.lecun.com/exdb/mnist/



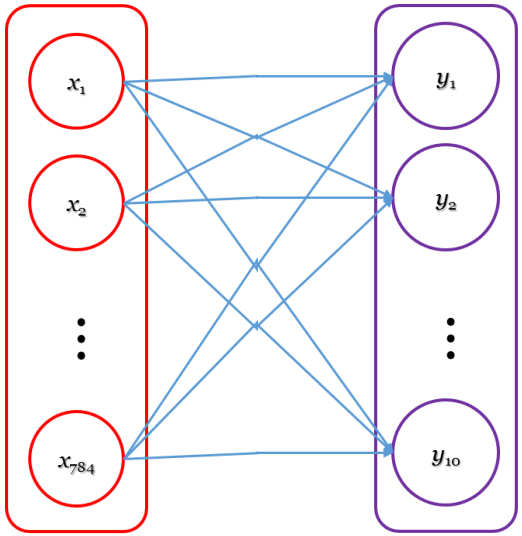


MNIST: 데이터 로딩

[source 4-1]

```
# 학습 데이터(MNIST)
(train_labels, train_images) = mnist.get_data('./datas/', 'train')
(test_labels, test_images) = mnist.get_data('./datas/', 'test')

# 데이터 사이즈 출력
print('total train data : ' + str(train_labels.size))
print('total test data : ' + str(test_labels.size))

# 임의의 수 내용 출력
print(train_images[0])
print(train_labels[0])

# 10개 수 출력(with matplot)
print('label: %s' % (train_labels[0:10]))

for i in range(10):
    plt.subplot(1, 10, i+1)
    plt.imshow(train_images[i], cmap='Greys_r')
    plt.axis('off')
plt.show()
```

SLP(SingleLayer Perceptron)

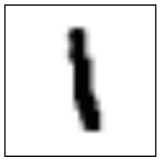


28 x 28 = 784

10

입력 층을 기반으로 하는 'SingleLayer Perceptron'
activation으로는 Softmax & cross entropy를 사용한다

기존 linear regress 예제에서 숫자 입력 데이터를
MNIST 이미지 -> 숫자(10개) 데이터로 변환



숫자 1

SLP(SingleLayer Perceptron)

1) Logistics 소스 복사
2) 정의 부분 변경

[source 4-2]

```
# 학습 데이터(MNIST)
(train_labels, train_images) = mnist.get_data('./datas/', 'train')

# X(이미지)와 Y(숫자값)의 입력값
X = tf.placeholder(tf.float32, [None, 28*28]) # 무한대 x 784(28*28) 행렬
Y = tf.placeholder(tf.int32, [None]) # 무한대 x 1 행렬

# SLP Model
pred = model(X)

# Cost Function 설계 (Cross Entropy)
cross_entropy = tf.nn.sparse_softmax_cross_entropy_with_logits(labels=Y, logits=pred)
cost = tf.reduce_mean(cross_entropy)

# Gradient descent Optimizer(학습)
optimizer = tf.train.GradientDescentOptimizer(0.01).minimize(cost)
```

SLP(SingleLayer Perceptron)

3) 모델 함수 추가

[source 4-2]

```
# SLP Model
def model(input_X):
    # 모델의 wright와 bias의 배열값을 0으로 초기화
    W = tf.Variable(tf.zeros([28*28, 10]), name="weight")
    b = tf.Variable(tf.zeros([10]), name="bias")

    # SoftMax 모델을 생성
    pred = tf.nn.softmax(tf.matmul(X, W) + b)

    return pred

# SLP Model
pred = model(X)
```

SLP(SingleLayer Perceptron)

4) 학습 부분 변경
5) 실행 & 오류 디버깅

[source 4-2]

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

    index_in_epoch = 0
    # 학습횟수(epoch:60) -> 총 60000개
    for epoch in range(1, 60):
        start = index_in_epoch
        index_in_epoch += 1000 # 배치 1000개
        end = index_in_epoch

        X_images = np.reshape(train_images[start:end], [-1, 28*28])
        Y_labels = train_labels[start:end]

        sess.run(optimizer, feed_dict={X: X_images , Y: Y_labels})
      # 로그
        training_cost = sess.run(cost, feed_dict={X: X_images , Y: Y_labels})
        print(epoch, training_cost)
    print("학습완료! (cost : " + str(training_cost) + ")")
```

SLP(SingleLayer Perceptron)

6) 검증 부분 변경

[source 4-2]

```
# 테스트 데이터로 테스트(총 10000개)
(test_labels, test_images) = mnist.get_data('./datas/', 'test')

test_X = np.reshape(test_images, [-1, 28*28])
test_Y = test_labels

testing_cost = sess.run(cost, feed_dict={X: test_X, Y: test_Y})
print("테스트 완료! (cost : " + str(testing_cost) + ")")

# 학습과 테스트 cost비교(절대값)
print("테스트와 학습의 cost차이 : ", abs(training_cost - testing_cost))
print("학습완료! (cost : " + str(training_cost) + ")")
```

SLP(SingleLayer Perceptron)

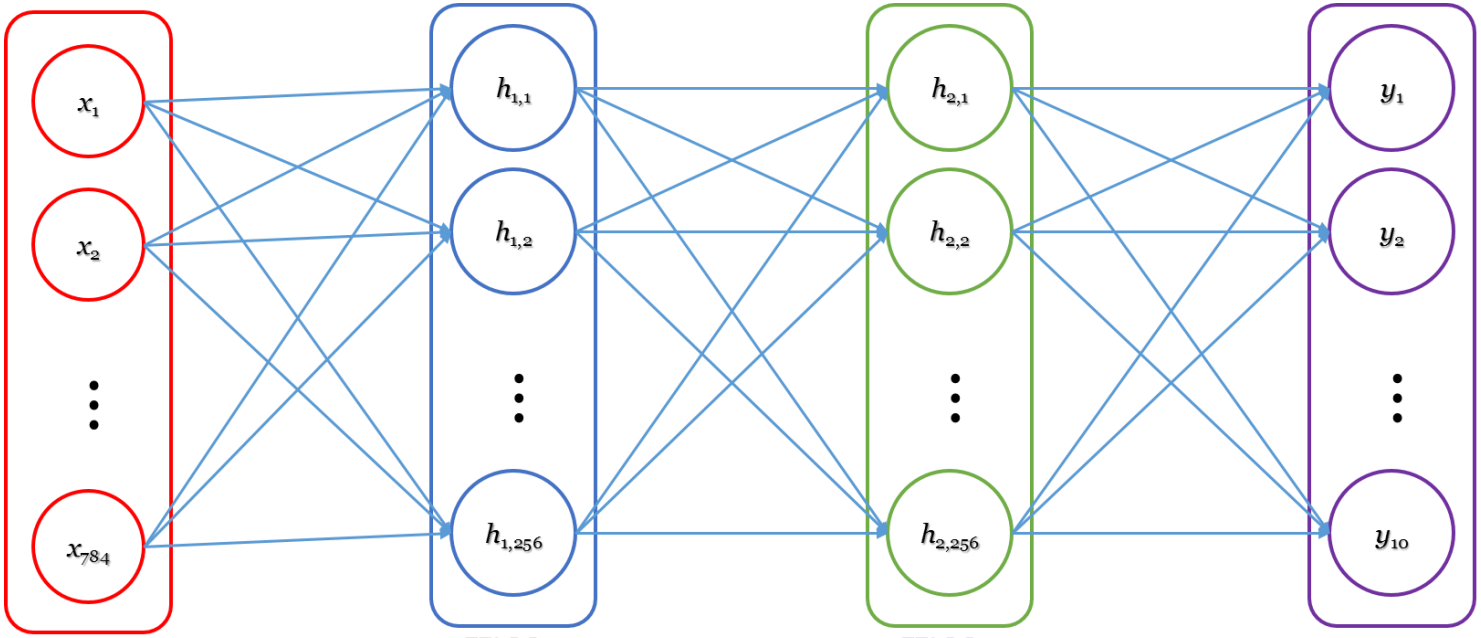
7) 값 예측 변경

[source 4-2]

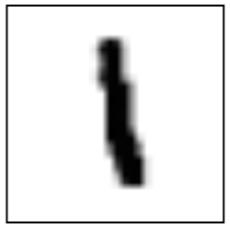
```
# 값 예측 (10개)

for i in range(20):

    x_test = np.reshape(train_images[i], [-1, 28*28])

    arr_data = sess.run(pred, feed_dict={X: x_test})



    pred_val = tf.argmax(arr_data, 1)

    real_val = train_labels[i]



    print("예측값: " + str(pred_val.eval()) + " / 실제값" + str(real_val) + " => " ₩

          + str(tf.equal(pred_val, real_val).eval()))
```

train cost 결과:
SLP(softmax)
=> cost : 1.688616

MLP(MultiLayer Perceptron)



28 x 28 = 784 128 64 10



숫자 1

MLP(MultiLayer Perceptron)

1) 모델 함수 변경
2) activation 함수 변경(sigmoid -> tanh)

[source 4-3]

```
# MLP Model
def model(input_X):
    # 히든 레이어 1
    W1 = tf.Variable(tf.truncated_normal([28*28, 128], stddev=0.1))
    b1 = tf.Variable(tf.zeros([128]))
    h1 = tf.nn.sigmoid(tf.matmul(input_X, W1) + b1)

    # 출력(fully connected) 레이어 (10개 출력)
    class_num = 10
    W_fc = tf.Variable(tf.truncated_normal([128, class_num], stddev=0.1))
    b_fc = tf.Variable(tf.zeros([class_num]))
    pred = tf.matmul(h1, W_fc) + b_fc

    return pred

pred = model(X)
```

train cost 결과:

MLP(sigmoid) Layer 1개
=> cost : 2.017988

MLP(tanh) Layer 1개
=> cost : 1.9295076

MLP(MultiLayer Perceptron)

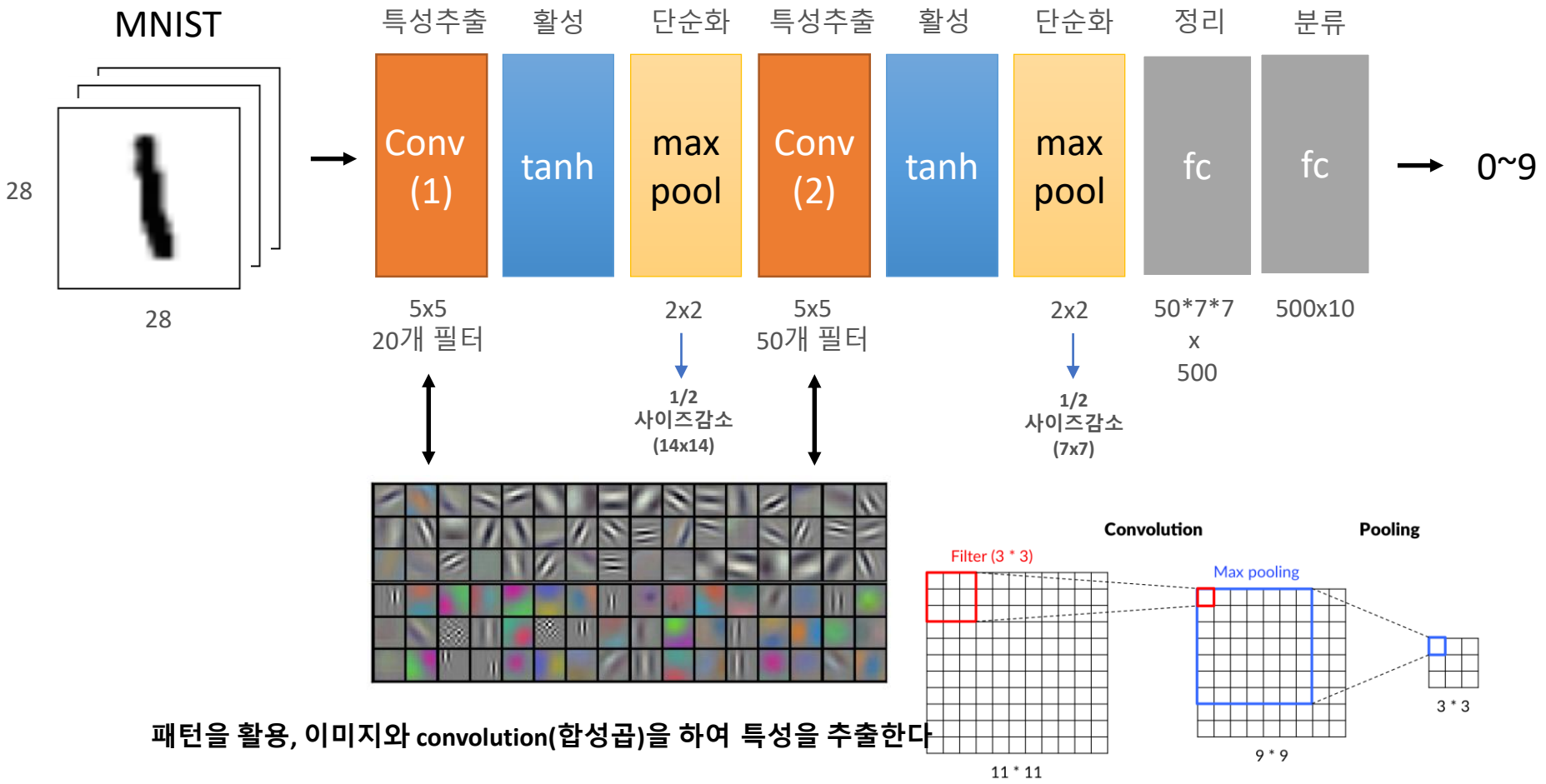
[source 4-3]

3) 모델에 '히든 레이어 2' 추가

```
# MLP Model
def model(input_X):
    # 히든 레이어 1
    [...]
    # 히든 레이어2
    W2 = tf.Variable(tf.truncated_normal([128, 64], stddev=0.1))
    b2 = tf.Variable(tf.zeros([64]))
    h2 = tf.nn.tanh(tf.matmul(h1, W2) + b2)

    # 출력(fully connected) 레이어 (10개 출력)
    class_num = 10
    W_fc = tf.Variable(tf.truncated_normal([64, class_num], stddev=0.1))
    b_fc = tf.Variable(tf.zeros([class_num]))
    pred = tf.matmul(h2, W_fc) + b_fc
    return pred

pred = model(X)
```

train cost 결과:
MLP(tanh) Layer 2개
=> cost : 1.7680303

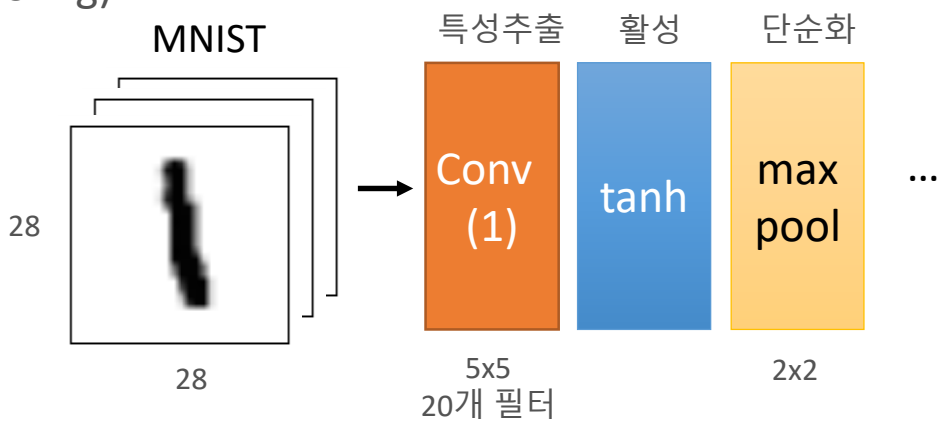
CNN (Convolution Neural Network)



패턴을 활용, 이미지와 convolution(합성곱)을 하여 특성을 추출한다

CNN (Convolution Neural Network)

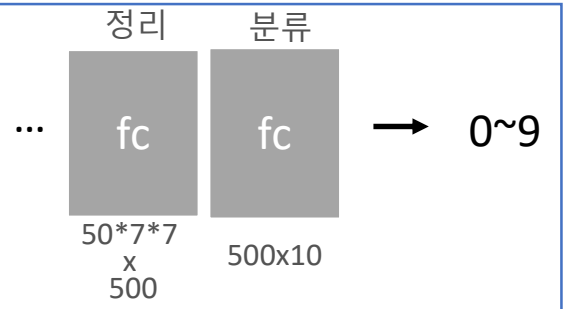
1) 모델 함수 변경 (Conv, Activate, Pooling)




```
def model(input_X):
    x_reshape = tf.reshape(X, [-1, 28, 28, 1])

    # Conv 레이어1
    W_conv1 = tf.Variable(tf.truncated_normal([5, 5, 1, 20], stddev=0.1))
    b_conv1 = tf.Variable(tf.zeros([20]))
    h_conv1 = tf.nn.conv2d(x_reshape, W_conv1, strides=[1, 1, 1, 1], padding='SAME') + b_conv1
    fmap_conv1 = tf.nn.tanh(h_conv1) # -> Feature(Activation) Map 생성

    # Pooling(Max) 레이어1
    h_pool1 = tf.nn.max_pool(fmap_conv1, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
    .
    .
```

CNN (Convolution Neural Network)

1) 모델 함수 변경 (Convolution Layer 1 설정)

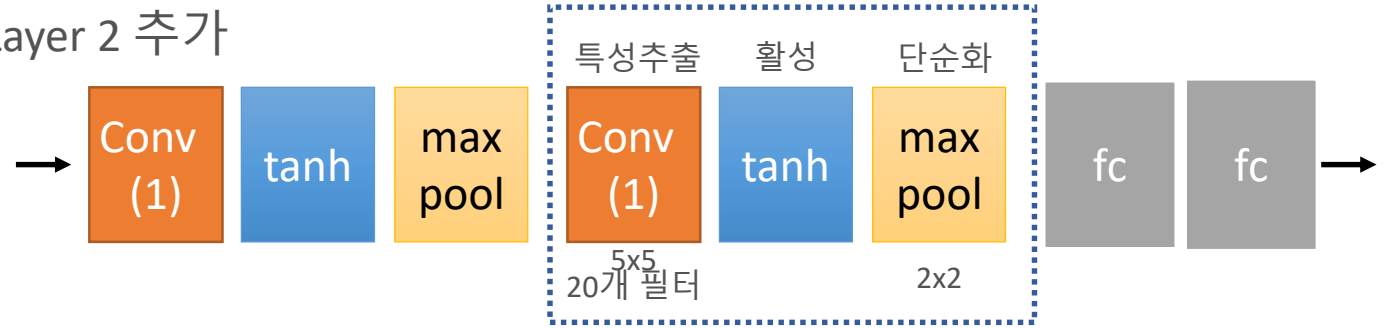



```
def model(input_X):
    ...
    # fully-connected 레이어 1
    W_fc1 = tf.Variable(tf.truncated_normal([20 * 14 * 14, 500], stddev=0.1))
    b_fc1 = tf.Variable(tf.zeros([500]))
    h_pool1_flat = tf.reshape(h_pool1, [-1, 20 * 14 * 14])
    h_fc1 = tf.nn.tanh(tf.matmul(h_pool1_flat, W_fc1) + b_fc1)

    # 출력(fully connected) 레이어 2 (10개 출력)
    class_num = 10
    W_fc2 = tf.Variable(tf.truncated_normal([500, class_num], stddev=0.1))
    b_fc2 = tf.Variable(tf.zeros([class_num]))
    pred = tf.matmul(h_fc1, W_fc2) + b_fc2

    return pred

pred = model(X)
```

train cost 결과:
CNN(tanh) Layer 1개
=> cost : 0.92411

CNN (Convolution Neural Network)

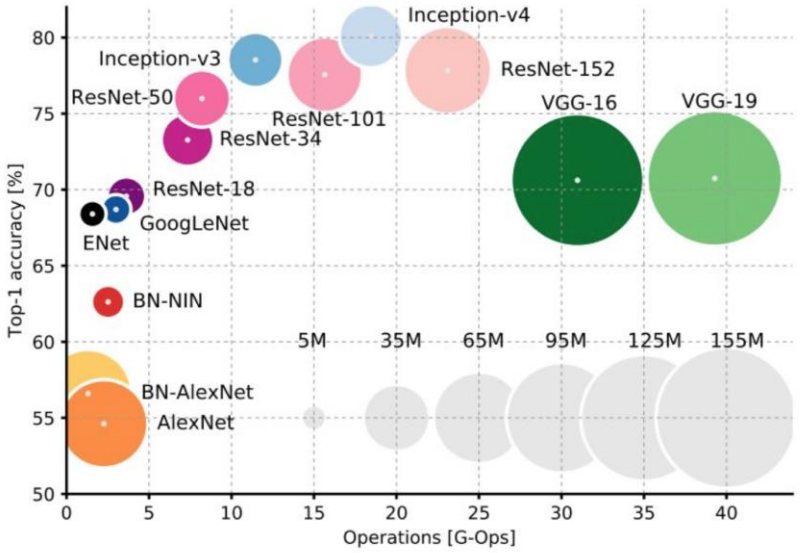
2) Convolution Layer 2 추가




```
...
# Conv 레이어2
W_conv2 = tf.Variable(tf.truncated_normal([5, 5, 20, 50], stddev=0.1))
b_conv2 = tf.Variable(tf.zeros([50]))
h_conv2 = tf.nn.conv2d(h_pool1, W_conv2, strides=[1, 1, 1, 1], padding='SAME') + b_conv2
fmap_conv2 = tf.nn.tanh(h_conv2) # -> Feature(Activation) Map 생성

# Pooling(Max) 레이어2
h_pool2 = tf.nn.max_pool(fmap_conv2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')

# fully-connected 레이어 1
W_fc1 = tf.Variable(tf.truncated_normal([50 * 7 * 7, 500], stddev=0.1))
b_fc1 = tf.Variable(tf.zeros([500]))
h_pool2_flat = tf.reshape(h_pool2, [-1, 50 * 7 * 7])
h_fc1 = tf.nn.tanh(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)
...
```

train cost 결과:
CNN(tanh) Layer 1개
=> cost : 0.5423556

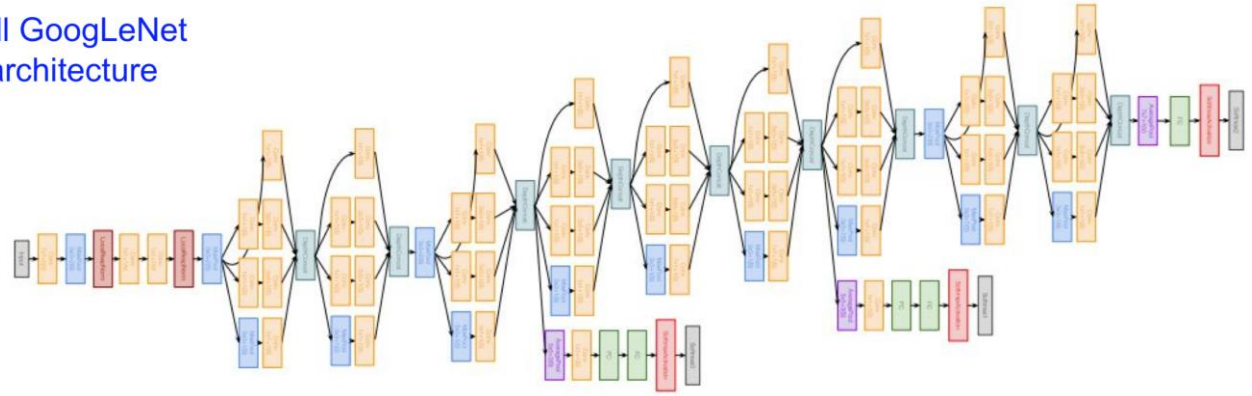
CNN (Convolution Neural Network)



CNN 알고리즘 종류

-> 레이어 구성에 따라 성능의 차이가 많이 남

Full GoogLeNet architecture



CNN (Convolution Neural Network)

https://cs.stanford.edu/people/karpathy/convnetjs/

ConvNetJS

Deep Learning in your browser

Intro | Deep Learning Resources | Getting Started | Documentation

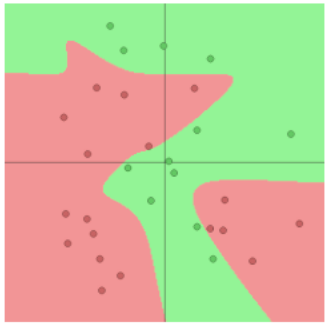
ConvNetJS is a Javascript library for training Deep Learning models (Neural Networks) entirely in your browser. Open a tab and you're training. No software requirements, no compilers, no installations, no GPUs, no sweat.

Browser Demos

Classify **MNIST** digits with a Convolutional Neural Network

Classify **CIFAR-10** with Convolutional Neural Network

Interactively classify toy 2-D data with a Neural Network

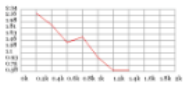


Training Stats

pause

Forward time per example: 5ms
Backprop time per example: 6ms
Classification loss: 0.57736
L2 Weight decay loss: 0.00128
Training accuracy: 0.82
Validation accuracy: 0.68
Examples seen: 1561

Loss:



clear graph

Learning rate: 0.01 change
Momentum: 0.9 change
Batch size: 20 change
Weight decay: 0.001 change
save network snapshot as JSON
init network from JSON snapshot

Instantiate a Network and Trainer

```
layer_defs = [];
layer_defs.push({type:'input', out_sx:24, out_sy:24, out_depth:1});
layer_defs.push({type:'conv', sx:5, filters:8, stride:1, pad:2, activation:'relu'});
```

change netwi

Network Visualization

input (24x24x1)
max activation: 0.99607, min: 0
max gradient: 0.01863, min: -0.01242
Activations:
Activation Gradients:

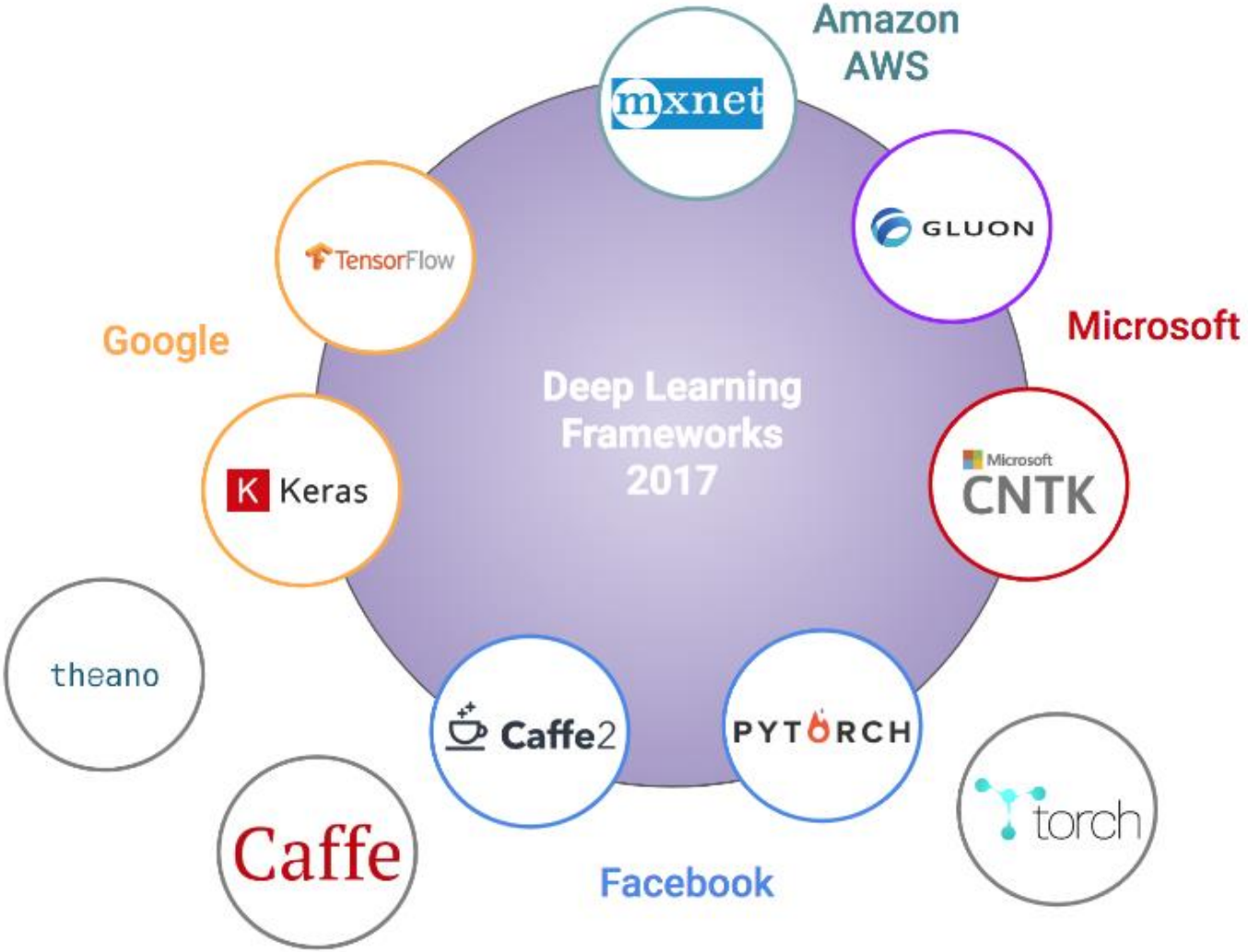
conv (24x24x8)
filter size 5x5x1, stride 1
max activation: 2.17689, min: -1.9214
max gradient: 0.00452, min: -0.00588
parameters: 8x5x5x1+8 = 208
Activations:
Activation Gradients:
Weights:
Weight Gradients:

relu (24x24x8)
max activation: 2.17689, min: 0
max gradient: 0.00452, min: -0.00588
Activations:
Activation Gradients:

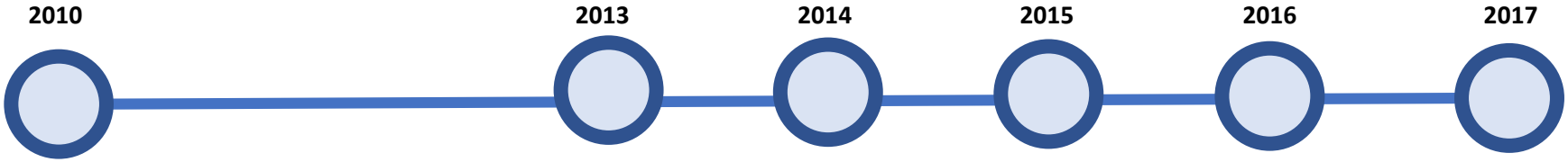
pool (12x12x8)
pooling size 2x2, stride 2
max activation: 2.17689,
Activations:
Activation Gradients:

딥러닝 프레임워크

딥러닝 프레임워크 Landscape



딥러닝 프레임워크 Timeline



(Nov.) theano

(Dec.) Caffe

(Jun.) DL4J DEEPLEARNING4J

(Jul.) torch

(Mar.) K

(Jun.) Chainer

(Oct.) mxnet

(Nov.) TensorFlow

(Jan.) Microsoft CNTK
On GitHub
(Debut: Apr. '2015)

(Jan.) PYTORCH

(Apr.) Caffe2

(Oct.) GLUON

https://github.com/Theano/Theano/blob/master/HISTORY.txt
https://www.embedded-vision.com/industry-analysis/technical-articles/caffe-deep-learning-framework-interview-core-developers
https://www.preferred-networks.jp/en/news/8531
https://www.microsoft.com/en-us/research/blog/microsoft-computational-network-toolkit-offers-most-efficient-distributed-deep-learning-computational-performance/github/#sm.0000qz6ljpzl3fnrxby1iwpo5v5rf
https://www.wired.com/2014/06/skymind-deep-learning/
https://twitter.com/fchollet/status/581627598741999616
https://github.com/dmlc/mxnet/issues?q=is%3Aissue+is%3Aopen+sort%3Acreated-asc
https://www.wired.com/2015/11/google-open-sources-its-artificial-intelligence-engine/
https://github.com/Theano/Theano/blob/master/HISTORY.txt
https://www.facebook.com/yann.lecun/posts/10152142434782143
https://techcrunch.com/2017/04/18/facebook-open-sources-caffe2-its-flexible-deep-learning-framework-of-choice/

Theano

http://deeplearning.net/software/theano/index.html

- 개발 및 유지보수 주체
  - Created by
    - James Bergstra, Frederic Bastien, etc. (http://www.iro.umontreal.ca/~lisa/pointeurs/theano_scipy2010.pdf_
  - Maintained by
    - LISA lab @ Université de Montréal (Now MILA)
- 릴리즈
  - Nov ‘2010
- 적용 사례
  - Keras
  - Lasagne
  - Blocks
- Motivation
  - There’s any.

Theano

http://deeplearning.net/software/theano/index.html

- 특징

F/W	주체	플랫폼	모바일	언어	인터페이스	OpenMP	CUDA	OpenCL	멀티GPU	분산
Theano	Université de Montréal	Linux, Mac, Windows	-	Python	Python	Y	Y	-	Y	-

  - 장점
    - low-level을 제어할 수 있는 API
    - 추상화된 그래프 모델 지원
    - 빠르고 유연함
    - RNN 다양한 지원
  - 단점
    - low-level API의 복잡성
    - 에러메시지 도움 안됨
    - 사전 학습모델 취약

Caffe

http://caffe.berkeleyvision.org/

- 개발 및 유지보수 주체
  - Created by
    - Yangqing Jia (http://daggerfs.com/)
      - UC Berkerey 컴퓨터 과학 Ph.D. / 지도 교수(Trevor Darrell, BAIR 책임자)
      - 구글 브레인 TensorFlow 프로젝트 참여
      - 페이스북 리서치 사이언티트
    - Evan Shellhamer (http://imaginarynumber.net/)
      - UC Berkerey 컴퓨터 과학 Ph.D. / 지도 교수(Trevor Darrell, BAIR 책임자)
  - Maintained by
    - BAIR(Berkeley Artificial Intelligence Research, http://bair.berkeley.edu/)
- 릴리즈
  - '2013: DeCAF (https://arxiv.org/abs/1310.1531)
  - Dec. '2013: Caffe v0
- 적용 사례
  - Facebook, Adobe, Microsoft, Samsung, Flickr, Tesla, Yelp, Pinterest, etc.
- Motivation
  - '2012 ILSVRC에서 발표한 AlexNet을 재현
  - DNN 정의/훈련/배포하기 위한 범용 F/W 구현

Caffe

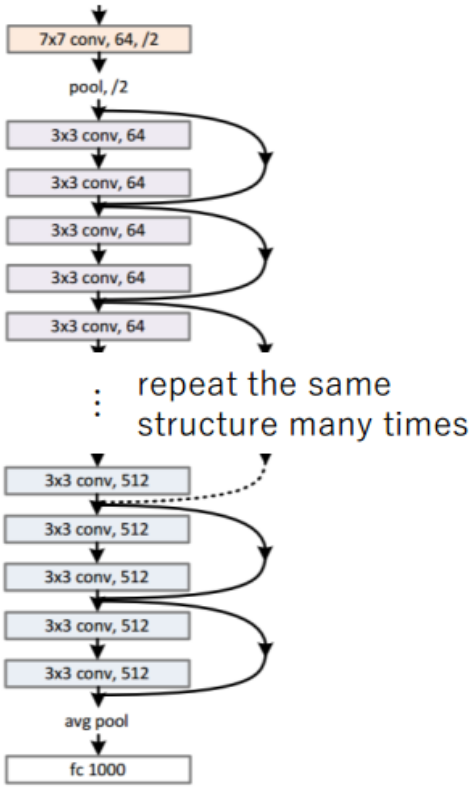
http://caffe.berkeleyvision.org/

Caffe2(http://caffe2.ai/) 출시
- By Facebook
- Android 지원, iOS 지원(예정)
- 분산 처리 지원

- 특징

F/W	주체	플랫폼	모바일	언어	인터페이스	OpenMP	CUDA	OpenCL	멀티GPU	분산
Caffe	BAIR	Linux, Mac	-	C++	Python, MATAB	Y	Y	-	Y	-

- 장점
  - 이미지 처리에 특화
  - 프로그래밍하는 대신 설정 파일로 학습 방법을 정의
  - Caffe Model Zoo를 통한 다양한 Pre-trained Model 제공
  - 이미지 기반 참조 모델의 de facto standard
- 단점
  - 이미지 이외의 텍스트, 사운드 등의 데이터 처리에는 부적합
  - 유연하지 못한 API
    - 새로운 기능 추가의 경우 C++/CUDA로 직접 구현 필요
  - 문서화가 잘 안되어 있음



DL4J

https://deeplearning4j.org/

- 개발 및 유지보수 주체
  - Created by
    - Adam Gibson @Skymind (CTO)
    - Chris Nicholson @Skymind (CEO)
  - Maintained by
    - Skymind (https://skymind.ai/)

skymind

- 릴리즈
  - Jun. ‘2014
- 적용 사례
  - 은행 Fraud Detection 연구 파트너쉽 with Nextremer in Japan (https://skymind.ai/press/nextremer)
- Motivation
  - 가장 많은 프로그래머를 보유하는 Java 기반의 딥러닝 프레임워크 개발
  - 추론엔진에 대해 엔터프라이즈 서비스급 안정성을 보장

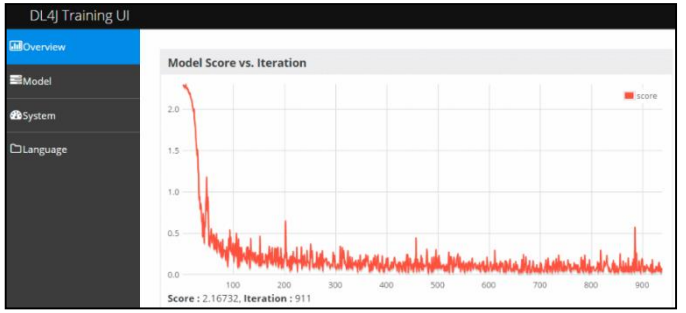
DL4J

https://deeplearning4j.org/

- 특징

F/W	주체	플랫폼	모바일	언어	인터페이스	OpenMP	CUDA	OpenCL	멀티GPU	분산
DL4J	SkyMind	Cross-platform (JVM)	Android	Java	Java, Scala, Python	Y	Y	-	Y	Y (Spark)

  - 장점
    - Java를 기반으로 한 쉬운 이식성 및 엔터프라이즈 시스템 수준의 안전성 제공
    - Spark 기반의 분산 처리 지원
    - 문서화가 잘 되어 있음
    - 학습 디버깅을 위한 시각화 도구 DL4J UI 제공
    - 기업 대상 기술 컨설팅 제공
  - 단점
    - Java 언어로 인한 학습 및 테스트 과정의 번거로움
    - 사용자 커뮤니티 상대적 취약
    - 부족한 예제



Torch

http://torch.ch/

- 개발 및 유지보수 주체
  - Created & Maintained by
    - Ronan Collobert: Research Scientist @ Facebook
    - Clément Farabet: Senior Software Engineer @ Twitter
    - Koray Kavukcuoglu: Research Scientist @ Google DeepMind
    - Soumith Chinatala: Research Engineer @ Facebook
- 릴리즈
  - Jul. ‘2014
- 적용 사례
  - Facebook, Google, Twitter, Element Inc., etc.
- Motivation
  - Unlike Caffe, for research rather than mass market
  - Unlike Theano, easy to use based on imperative model rather than symbolic model

Torch

http://torch.ch/

- 특징

F/W	주체	플랫폼	모바일	언어	인터페이스	OpenMP	CUDA	OpenCL	멀티GPU	분산
Torch	Ronan, Clément, Koray, Soumith	Linux, Mac, Windows	Android, iOS	C, Lua	Lua	Y	Y	Y	Y	Not officially

  - 장점
    - 알고리즘 모듈화가 잘 되어 있어 사용이 용이
    - 다양한 데이터 전처리 및 시각화 유틸리티 제공
    - 간단한 Lua 프로그래밍 구문
    - Imperative 프로그래밍 모델 기반의 직관적인 API
    - OpenCL 지원
    - 모바일 지원
  - 단점
    - 파이썬 인터페이스 없음(PyTorch 별도 존재)
    - 문서화가 잘 안되어 있음
    - 사용자 커뮤니티 취약
    - 심볼릭 모델 미제공
    - 상용 어플레케이션이 아니라 연구용으로 적합

Keras

https://keras.io/

- 개발 및 유지보수 주체
  - Created & Maintained by
    - Francois Chollet @Google
- 릴리즈
  - Mar. ‘2015
- 적용 사례
  - TensorFlow (http://www.fast.ai/2017/01/03/keras)
- Motivation
  - Provide a high-level interface based on deep learning framework like Theano, TensorFlow
  - Easy to use
  - 최소화, 단순화, 모듈화
  - 다양한 딥러닝 프레임워크와의 쉬운 연동

Keras

https://keras.io/

- 특징

F/W	주체	플랫폼	모바일	언어	인터페이스	OpenMP	CUDA	OpenCL	멀티GPU	분산
Keras	François Chollet	Linux, Mac, Windows	-	Python	Python	Y(Theano) N(TF)	Y	-	Y	-

  - 장점
    - 직관적인 API 인터페이스
    - Caffe, Torch, TensorFlow 등 다양한 딥러닝 프레임워크 모델 import 기능 제공
    - 문서화가 잘되어 있음
  - 단점
    - 기반 Theano 프레임워크에서 문제가 발생시 debugging이 어려움

Chainer

http://docs.chainer.org/en/latest/index.html

- 개발 및 유지보수 주체
  - Created & Maintained by
    - Preferred Networks, Inc.(https://www.preferred-networks.jp/ja/)
- 릴리즈
  - Jun. ‘2015
- 적용 사례
  - Toyota motors, Panasonic (https://www.wsj.com/articles/japan-seeks-tech-revival-with-artificial-intelligence-1448911981)
  - FANUC (http://www.fanucamerica.com/FanucAmerica-news/Press-releases/PressReleaseDetails.aspx?id=79)
- Motivation
  - Define-by-Run 아키텍처
    - 실행 시점에 네트워크 그래프가 정의됨
    - 복잡한 네트워크 정의를 보다 유연하게 지원할 수 있게 함

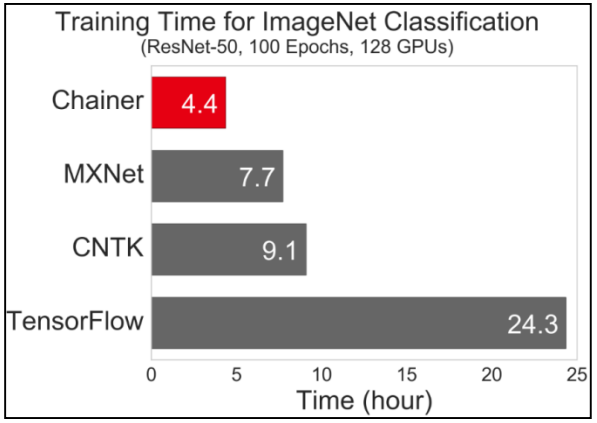
Chainer

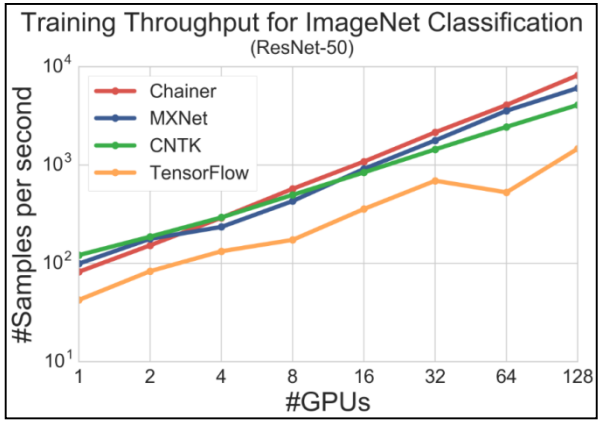
http://docs.chainer.org/en/latest/index.html

- 특징

F/W	주체	플랫폼	모바일	언어	인터페이스	OpenMP	CUDA	OpenCL	멀티GPU	분산
Chainer	Preferred Networks	Linux	-	Python	Python	-	Y	-	Y	Y

  - 장점
    - 빠른 속도





출처: Performance of Distributed Deep Learning using ChainerMN
http://chainer.org/general/2017/02/08/Performance-of-Distributed-Deep-Learning-Using-ChainerMN.html

MXNet

http://mxnet.io/

- 개발 및 유지보수 주체
  - Created by
    - CMU (http://www.cs.cmu.edu/~muli/file/mxnet-learning-sys.pdf)
  - Maintained by
    - DMLC(Distributed Machine Learning Community)
      - CMU, NYU, NVIDIA, Baidu, Amazon, Microsoft etc.
- 릴리즈
  - Oct. ‘2015
- 적용 사례
  - AWS (https://www.infoq.com/news/2016/11/amazon-mxnet-deep-learning)
- Motivation
  - Support for Mixed Programming Model: Imperative & Symbolic
  - Support for Portability: Desktops, Clusters, Mobiles, etc.
  - Support for Multiple Languages: C++, R, Python, Matlab, Javascript, etc.

MXNet

http://mxnet.io/

- 특징

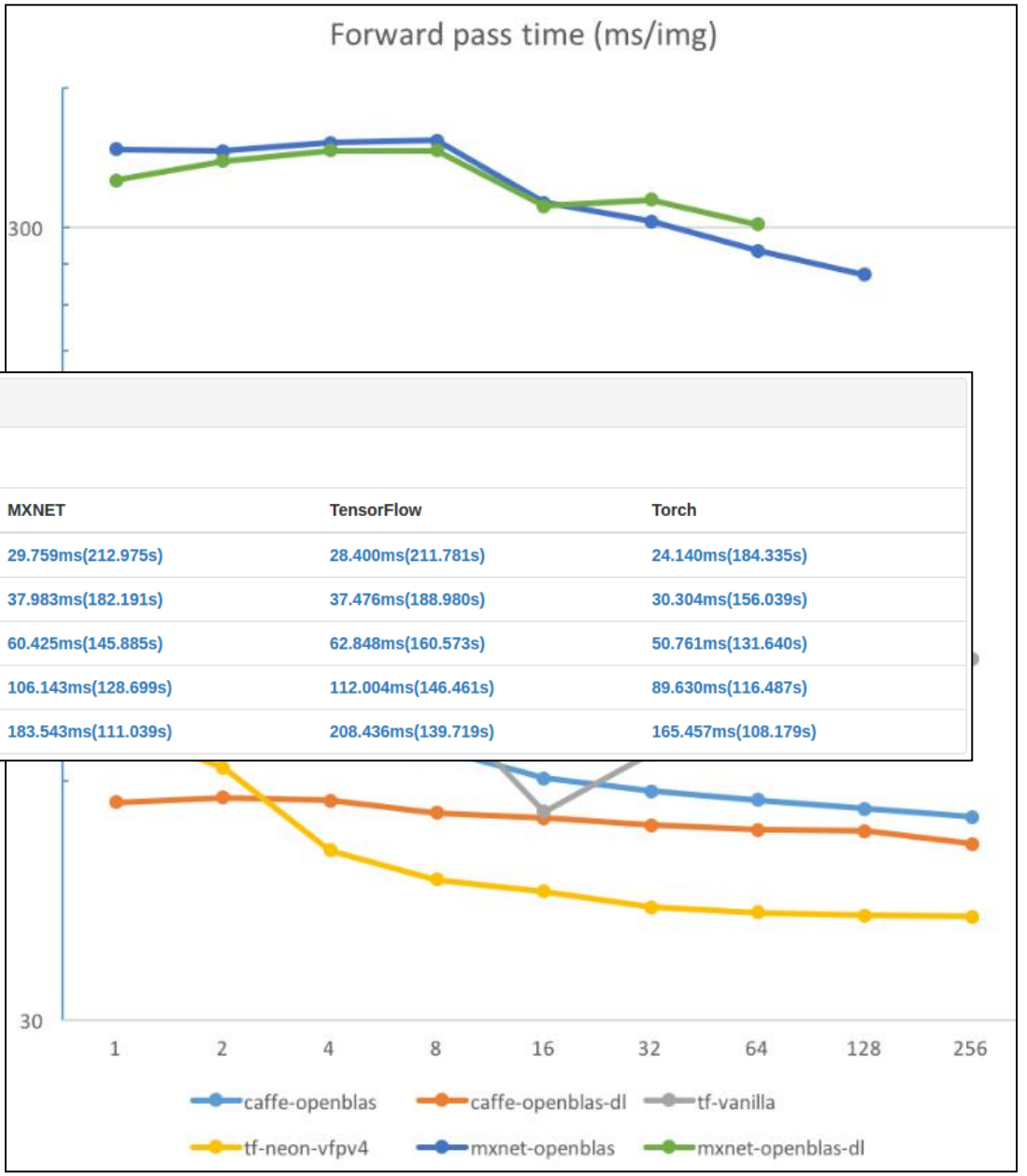
F/W	주체	플랫폼	모바일	언어	인터페이스	OpenMP	CUDA	OpenCL	멀티GPU	분산
MXNet	DMLC	Linux, Mac, Windows, Javascript	Android, iOS	C++	C++, Python, Julia, MATLAB, JavaScript, Go, R, Scala, Perl	Y	Y	-	Y	Y (key-value)

- 장점
  - 다양한 프로그래밍 인터페이스 제공
  - 모바일 지원
  - 빠르게 발전
  - low-level / high-level API 모두 제공
  - Imperative / Graph 프로그래밍 모델 모두 지원

MXNet

http://mxnet.io/

- 단점
  - 다소 처리 속도 느림



fcn5 on K80

Tesla K80, CUDA: 8.0 CUDNN: v5.1 CUDA_DRIVER: 367.48 Network: fcn5

BatchSize	Caffe	CNTK	MXNET	TensorFlow	Torch
342	24.762ms(195.255s)	25.001ms(239.932s)	29.759ms(212.975s)	28.400ms(211.781s)	24.140ms(184.335s)
512	32.088ms(172.273s)	31.246ms(212.595s)	37.983ms(182.191s)	37.476ms(188.980s)	30.304ms(156.039s)
1024	55.074ms(150.900s)	52.691ms(190.474s)	60.425ms(145.885s)	62.848ms(160.573s)	50.761ms(131.640s)
2048	98.418ms(139.026s)	94.671ms(178.316s)	106.143ms(128.699s)	112.004ms(146.461s)	89.630ms(116.487s)
4096	238.965ms(168.775s)	181.415ms(173.852s)	183.543ms(111.039s)	208.436ms(139.719s)	165.457ms(108.179s)

출처: Benchmarking State-of-the-Art Deep Learning Software Tools
http://dlbench.comp.hkbu.edu.hk/?v=v7

출처: How to run deep neural networks on weak hardware
https://www.linkedin.com/pulse/how-run-deep-neural-networks-weak-hardware-dmytro-prylipko

TensorFlow

https://www.tensorflow.org/

- 개발 및 유지보수 주체
  - Created & Maintained by
    - Google Brain
- 릴리즈
  - Nov. ‘2015
- 적용 사례
  - Google
    - Search Signals (https://www.bloomberg.com/news/articles/2015-10-26/google-turning-its-lucrative-web-search-over-to-ai-machines)
    - Email auto-responder (https://research.googleblog.com/2015/11/computer-respond-to-this-email.html)
    - Photo Search (https://techcrunch.com/2015/11/09/google-open-sources-the-machine-learning-tech-behind-google-photos-search-smart-reply-and-more/#.t38yrr8:fUIZ)
- Motivation
  - It’s Google

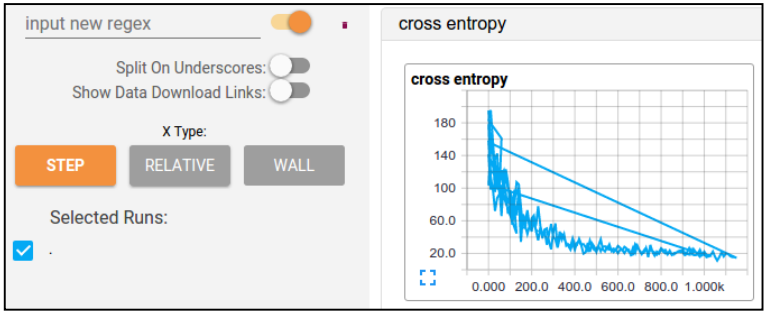
TensorFlow

https://www.tensorflow.org/

- 특징

F/W	주체	플랫폼	모바일	언어	인터페이스	OpenMP	CUDA	OpenCL	멀티GPU	분산
TensorFlow	Google	Linux, Mac, Windows	Android, iOS	C++, Python	Python, C/C++, Java, Go	N	Y	-	Y	Y (gRPC)

- 장점
  - 추상화된 그래프 모델
  - 학습 디버깅을 위한 시각화 도구 TensorBoard 제공

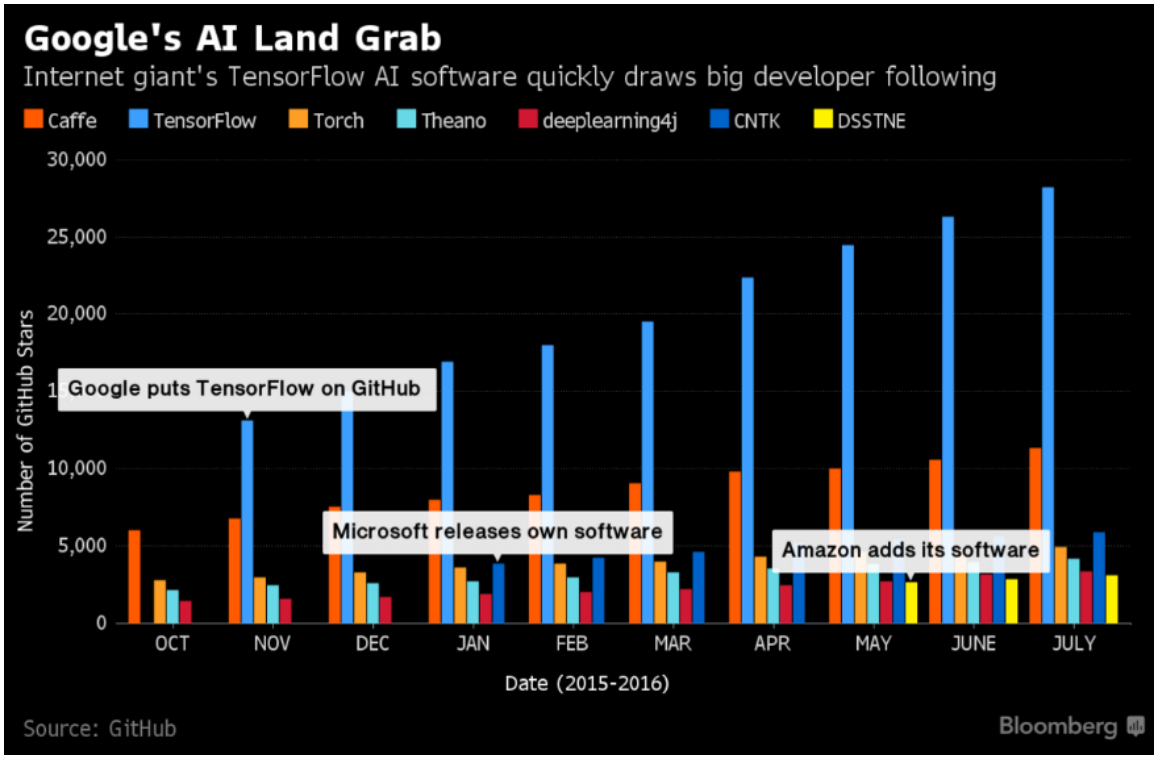


  - 모바일 지원
  - low-level / high-level API 모두 제공

TensorFlow

https://www.tensorflow.org/

- 장점
  - 방대한 사용자 커뮤니티



출처: Machine Learning Frameworks Comparison
https://blog.paperspace.com/which-ml-framework-should-i-use

TensorFlow

https://www.tensorflow.org/

- 단점
  - Define-and-Run 모델 / 런타임에 그래프 변경 안됨
  - Torch에 비해 느림

AlexNet (One Weird Trick paper) - Input 128x3x224x224

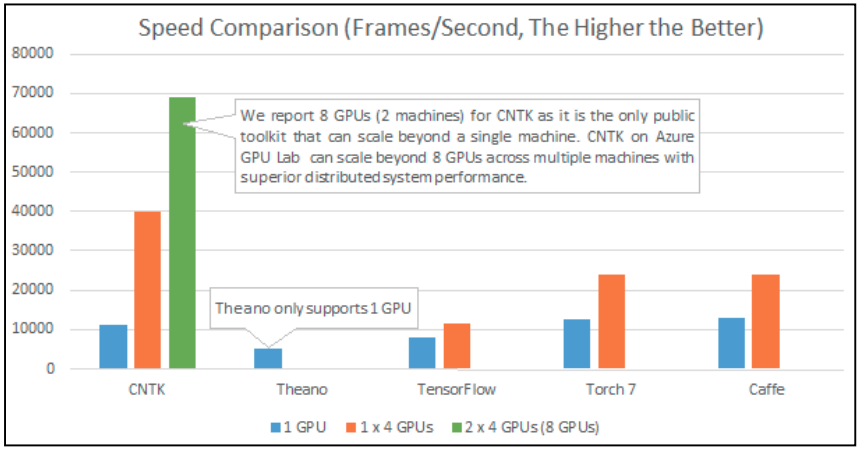
Library	Class	Time (ms)	forward (ms)	backward (ms)
CuDNN[R4]-fp16 (Torch)	cudnn.SpatialConvolution	71	25	46
Nervana-neon-fp16	ConvLayer	78	25	52
CuDNN[R4]-fp32 (Torch)	cudnn.SpatialConvolution	81	27	53
TensorFlow	conv2d	81	26	55
Nervana-neon-fp32	ConvLayer	87	28	58
fbfft (Torch)	fbnn.SpatialConvolution	104	31	72
Chainer	Convolution2D	177	40	136
cudaconvnet2*	ConvLayer	177	42	135
CuDNN[R2] *	cudnn.SpatialConvolution	231	70	161
Caffe (native)	ConvolutionLayer	324	121	203
Torch-7 (native)	SpatialConvolutionMM	342	132	210
CL-nn (Torch)	SpatialConvolutionMM	963	388	574

출처: soumith/convnet-benchmarks
https://github.com/soumith/convnet-benchmarks

CNTK

https://www.microsoft.com/en-us/research/product/cognitive-toolkit/

- 개발 및 유지보수 주체
  - Created & Maintained by
    - Microsoft Research
- 릴리즈
  - Jan. ‘2016
- 적용 사례
  - Microsoft’s speech recognition engine
  - Skype’s Translator
- Motivation
  - Efficient performance on distributed environments



https://www.microsoft.com/en-us/research/blog/microsoft-computational-network-toolkit-offers-most-efficient-distributed-deep-learning-computational-performance/

CNTK

https://www.microsoft.com/en-us/research/product/cognitive-toolkit/

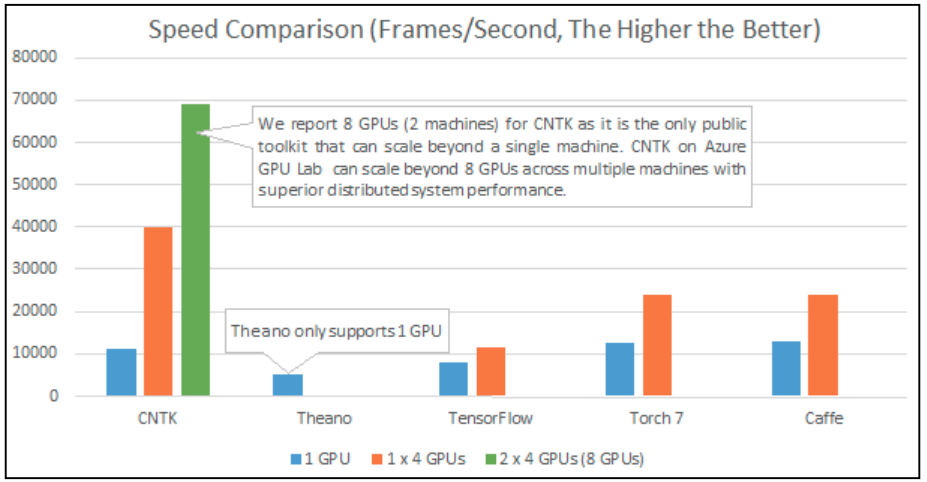
- 특징

F/W	주체	플랫폼	모바일	언어	인터페이스	OpenMP	CUDA	OpenCL	멀티GPU	분산
CNTK	Microsoft	Linux, Windows	-	C++	Python, C++	Y	Y	-	Y	Y (MPI)

- 장점
  - 처리 성능의 linear scaling
  - 범용 ML 프레임워크

- 단점
  - 협소한 사용자 커뮤니티
  - NVIDIA only (No OpenCL)

[2015. 7]



출처: Microsoft Computational Network Toolkit offers most efficient distributed deep learning computational performance
https://www.microsoft.com/en-us/research/blog/microsoft-computational-network-toolkit-offers-most-efficient-distributed-deep-learning-computational-performance/

딥러닝 프레임워크: 주요 특성

F/W	주체	플랫폼	모바일	언어	인터페이스	OpenMP	CUDA	OpenCL	멀티GPU	분산
Caffe	BAIR	Linux, Mac	-	C++	Python, MATLAB	Y	Y	-	Y	-
Chainer	Preferred Networks	Linux	-	Python	Python	-	Y	-	Y	Y
CNTK	Microsoft	Linux, Windows	-	C++	Python, C++	Y	Y	-	Y	Y (MPI)
DL4J	SkyMind	Cross-platform (JVM)	Android	Java	Java, Scala, Python	Y	Y	-	Y	Y (Spark)
Keras	François Chollet	Linux, Mac, Windows	-	Python	Python	Y(Theano) N(TF)	Y	-	Y	
MXNet	DMLC	Linux, Mac, Windows, Javascript	Android, iOS	C++	C++, Python, Julia, MATLAB, JavaScript, Go, R, Scala, Perl	Y	Y	-	Y	Y (key-value)
TensorFlow	Google	Linux, Mac, Windows	Android, iOS	C++, Python	Python, C/C++, Java, Go	N	Y	-	Y	Y (gRPC)
Theano	Université de Montréal	Linux, Mac, Windows	-	Python	Python	Y	Y	-	Y	
Torch	Ronan, Clément, Koray, Soumith	Linux, Mac, Windows	Android, iOS	C, Lua	Lua	Y	Y	Y	Y	Not officially

출처: Comparison of deep learning software https://en.wikipedia.org/wiki/Comparison_of_deep_learning_software#cite_note-29

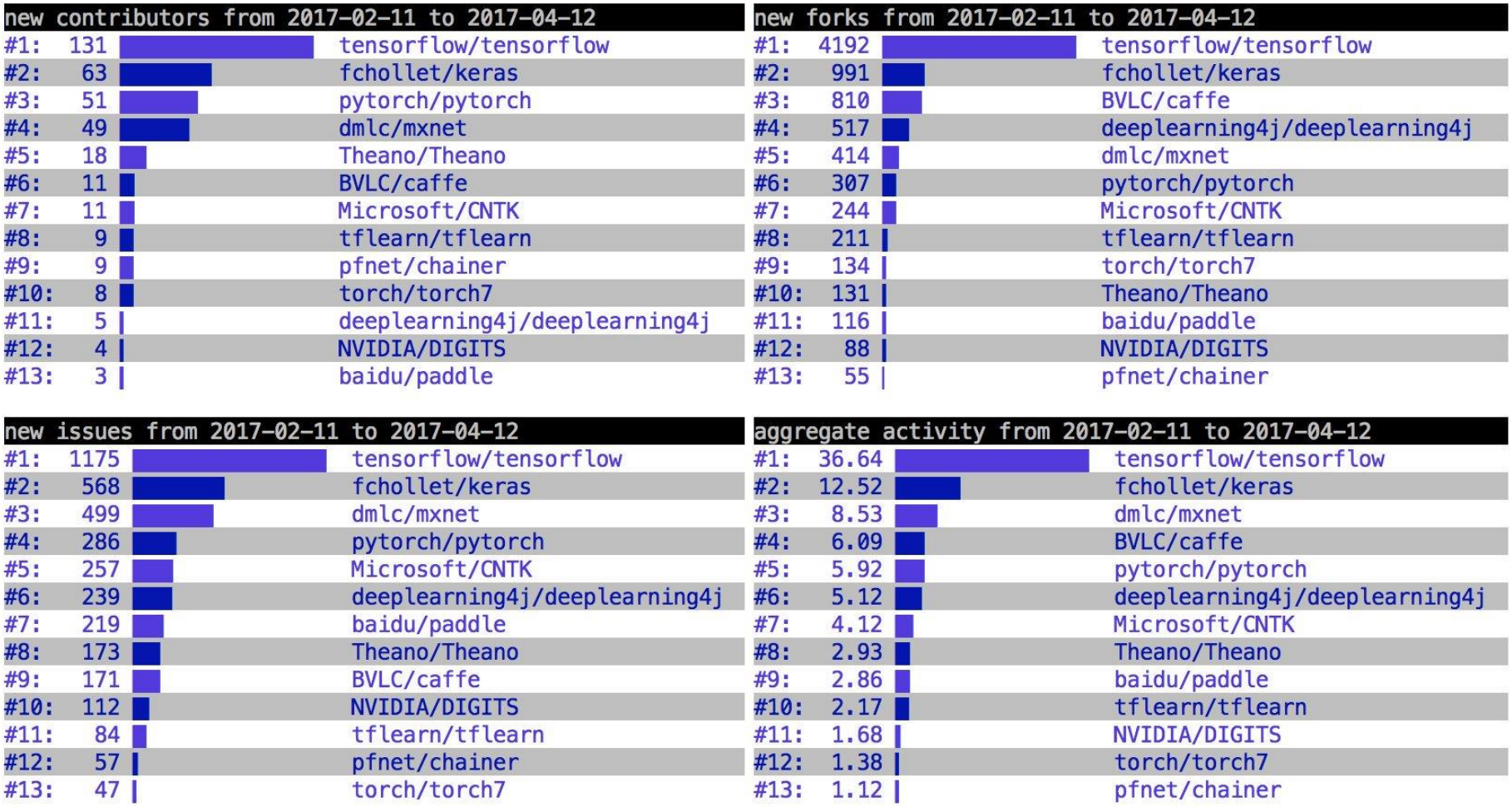
딥러닝 프레임워크: 주요 특성

	Languages	Tutorials and training materials	CNN modeling capability	RNN modeling capability	Architecture: easy-to-use and modular front end	Speed	Multiple GPU support	Keras compatible
Theano	Python, C++	++	++	++	+	++	+	+
Tensor-Flow	Python	+++	+++	++	+++	++	++	+
Torch	Lua, Python (new)	+	+++	++	++	+++	++	
Caffe	C++	+	++		+	+	+	
MXNet	R, Python, Julia, Scala	++	++	+	++	++	+++	
Neon	Python	+	++	+	+	++	+	
CNTK	C++	+	+	+++	+	++	+	

출처: Getting Started with Dep Learning
https://svds.com/getting-started-deep-learning/

딥러닝 프레임워크: Popularity

Deep learning libraries: GitHub activity from February 11 to April 12, 2017



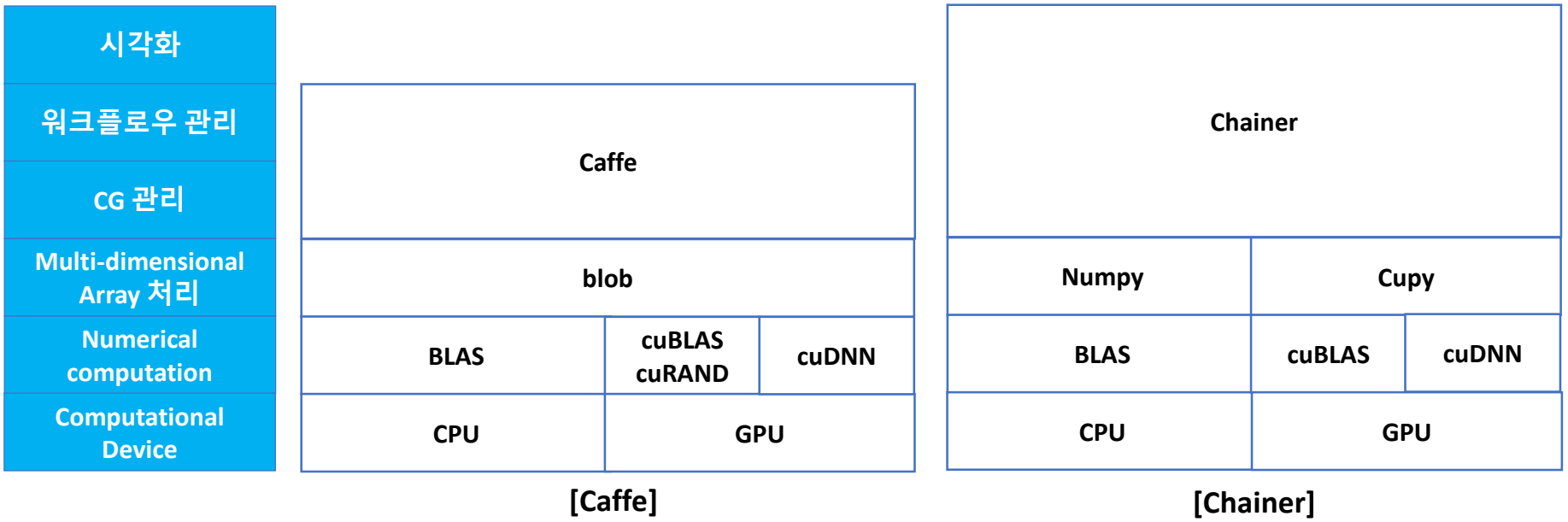
딥러닝 프레임워크: Tech. Stack

name	functions	example
Graphical visualization		DIGITS, TensorBoard
Machine learning workflow management	Dataset prep, Save/Load Training loop	Keras, TF slim
Computational graph(CG) management	Build/Optimize CGs Forward/Back prop	Theano, TensorFlow Torch.nn
Multi-dimensional array processing	High-level array manipulation	NumPy, CuPy Eigen, Torch (core)
Numerical computation	Matrix operation Convolution	BLAS(OpenBLAS, MKL), cuBLAS, cuDNN, MKL DNN
Computational device		CPU, GPU, TPU, FPGA

출처: DLIF: Common design of neural network implementations
https://www.dropbox.com/s/qfz34ba3ftuli6b/AAAI2017-2-0203.pdf

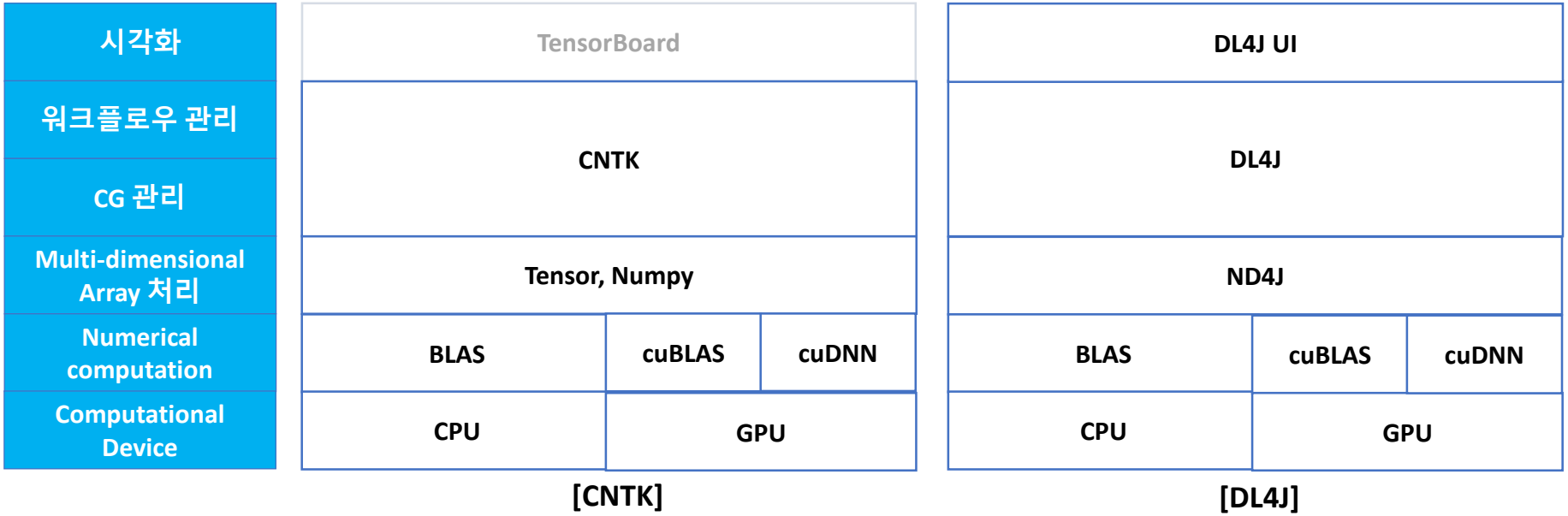
딥러닝 프레임워크: Tech. Stack

출처: DLIF: Common design of neural network implementations
https://www.dropbox.com/s/qfz34ba3ftuli6b/AAAI2017-2-0203.pdf



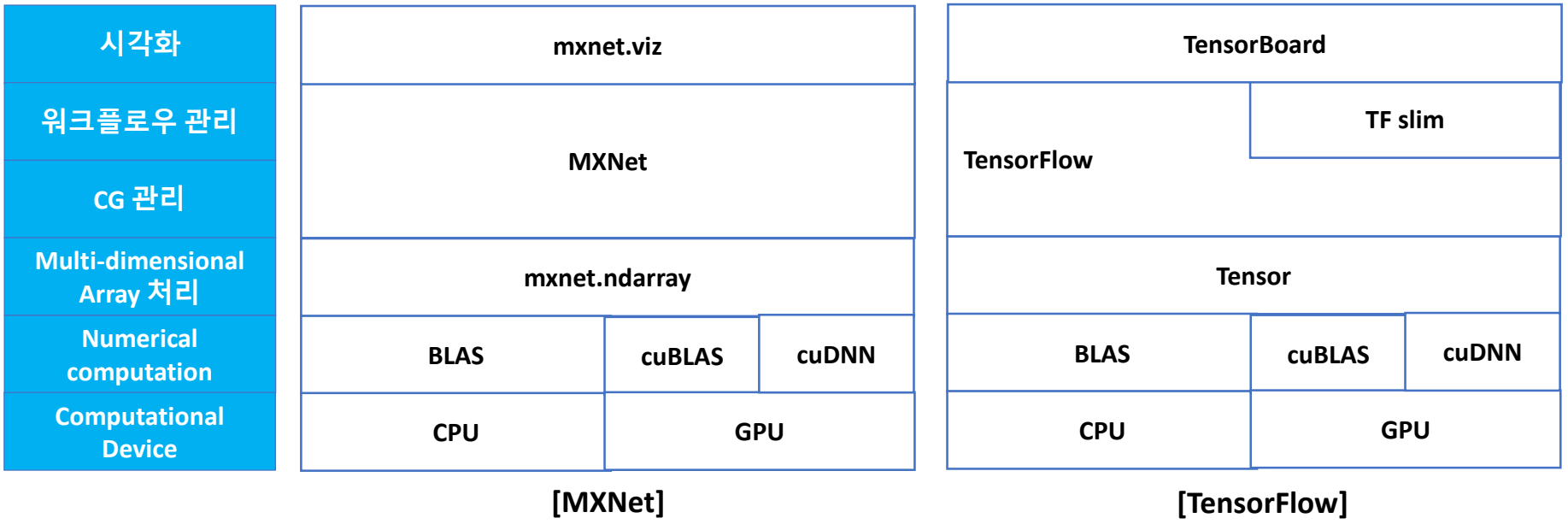
딥러닝 프레임워크: Tech. Stack

출처: DLIF: Common design of neural network implementations
https://www.dropbox.com/s/qfz34ba3ftuli6b/AAAI2017-2-0203.pdf



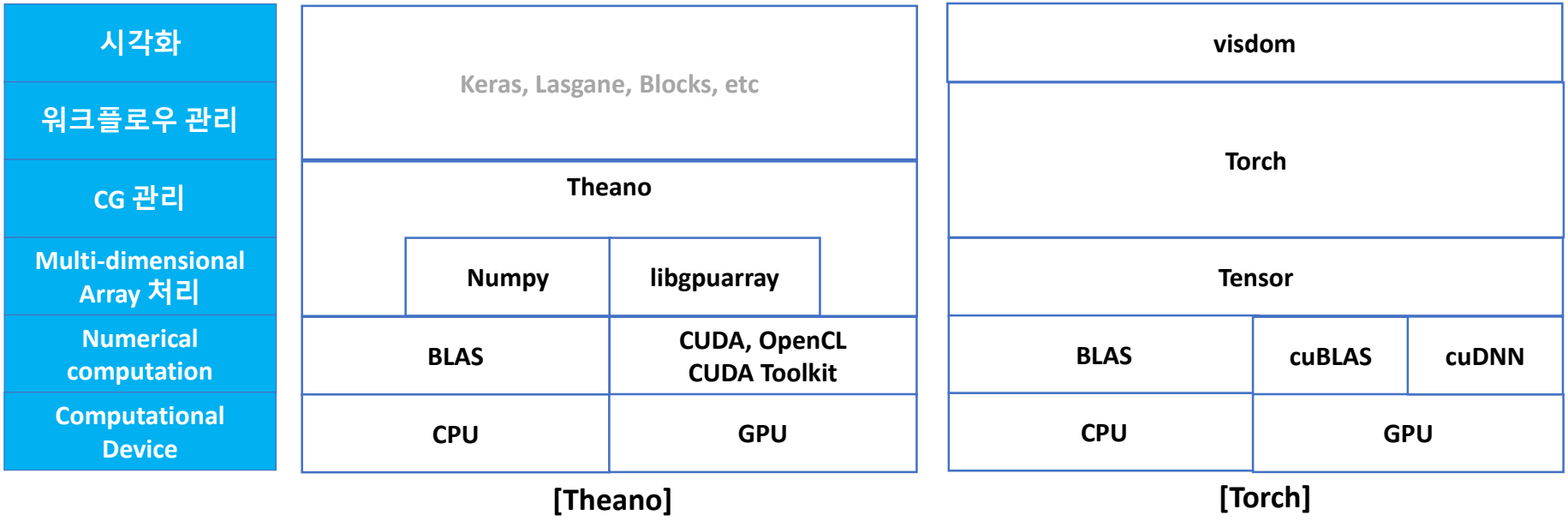
딥러닝 프레임워크: Tech. Stack

출처: DLIF: Common design of neural network implementations
https://www.dropbox.com/s/qfz34ba3ftuli6b/AAAI2017-2-0203.pdf



딥러닝 프레임워크: Tech. Stack

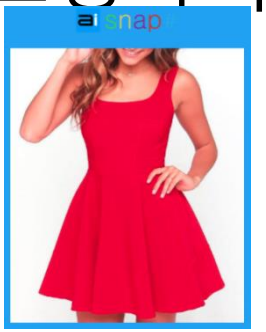
출처: DLIF: Common design of neural network implementations
https://www.dropbox.com/s/qfz34ba3ftuli6b/AAAI2017-2-0203.pdf

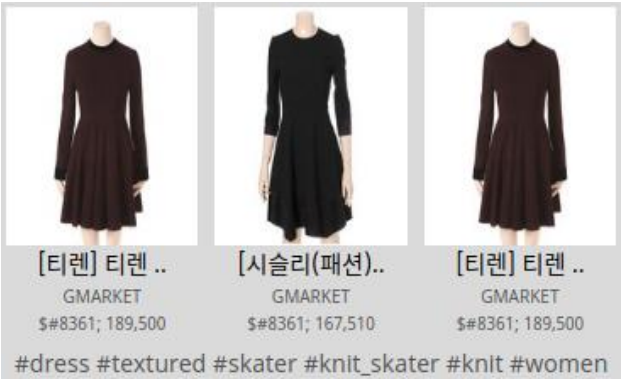


패션 아이템 및 음식 이미지 분석

ai snap#

- 마음에 드는 상품 사진으로 이미지 검색하고 쇼핑하기





패션 도메인 카테고리 & 속성

- 학습 및 추론을 위한 패션 도메인 카테고리와 속성 정의

[대분류] [중분류]

- 상의
  - 티셔츠
  - 블레이저
  - …
- 하의
  - 청바지
  - 스커트
  - …
- 전신
  - 코트
  - 드레스
  - …

[속성#1: 형태]

- boxy
- **fit**
- **slim**
- laced
- polo
- …

[속성#: 스타일]

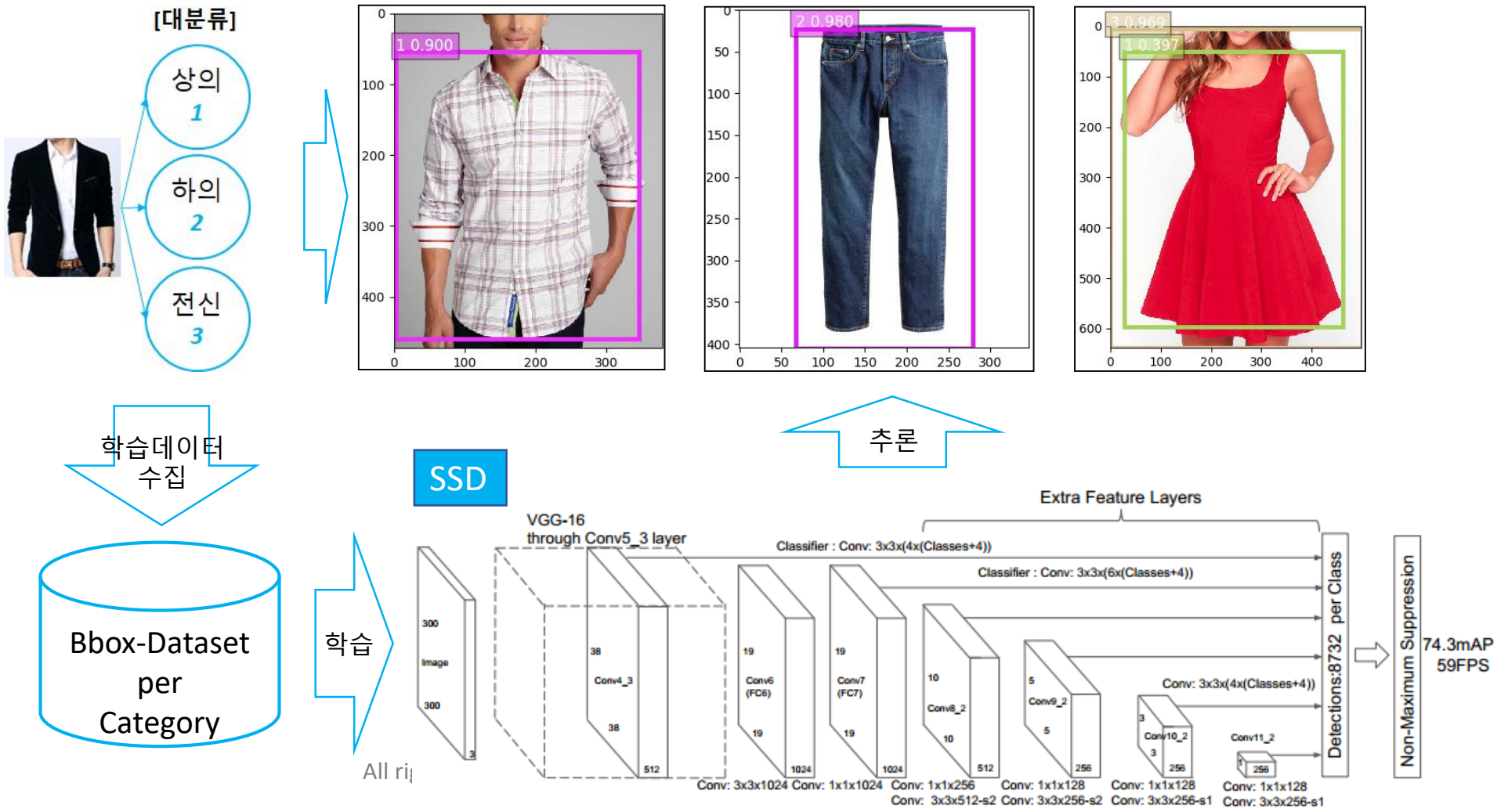
- basic
- athletic
- chic
- **classic**
- loveley
- ..

[속성#: 요소] …

- belted
- collar
- v-neck
- zipper
- **button**
- …

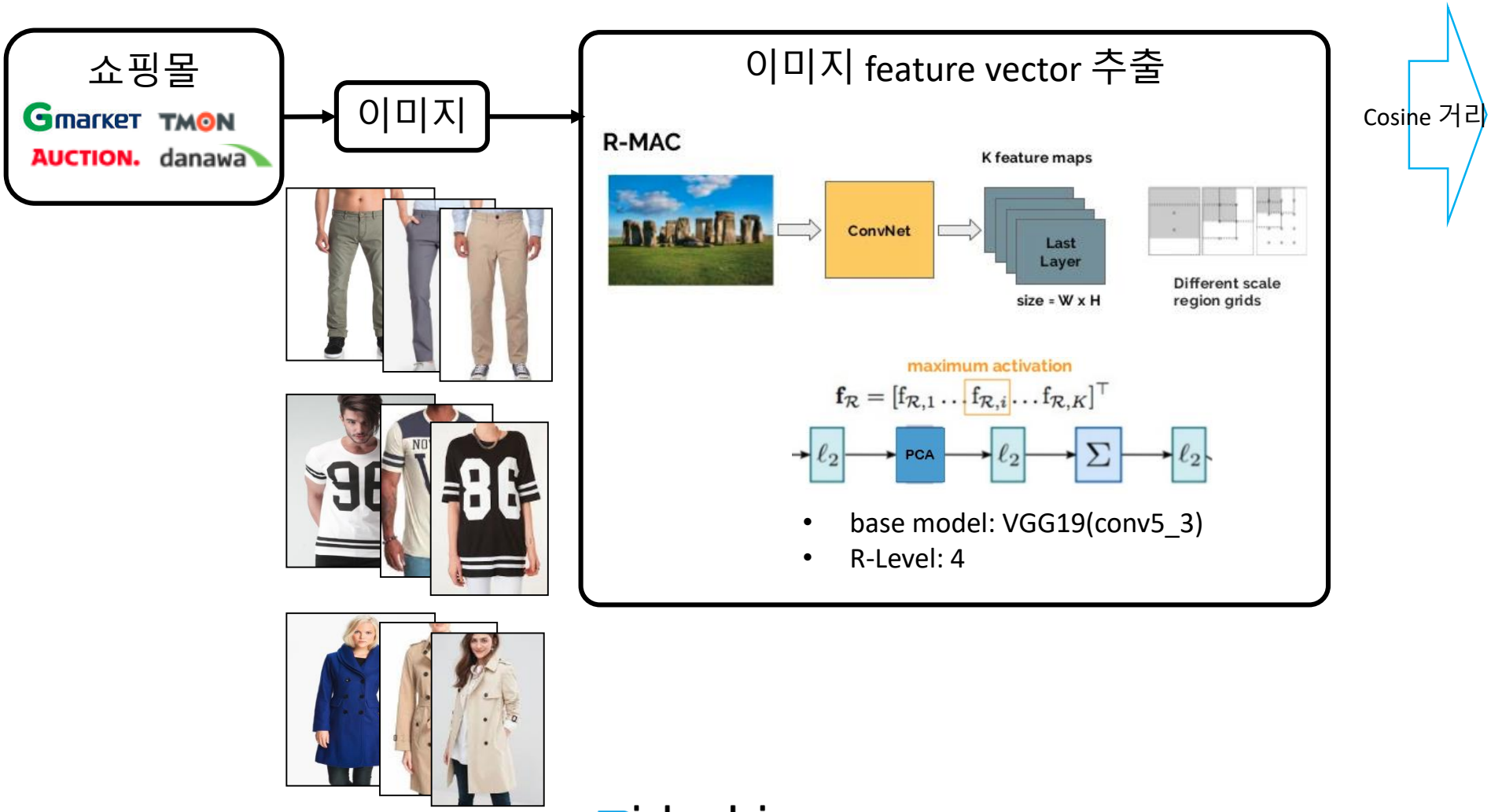
관심 영역(Region of Interest) 탐지

- 이미지 검색을 위한 영역을 탐지



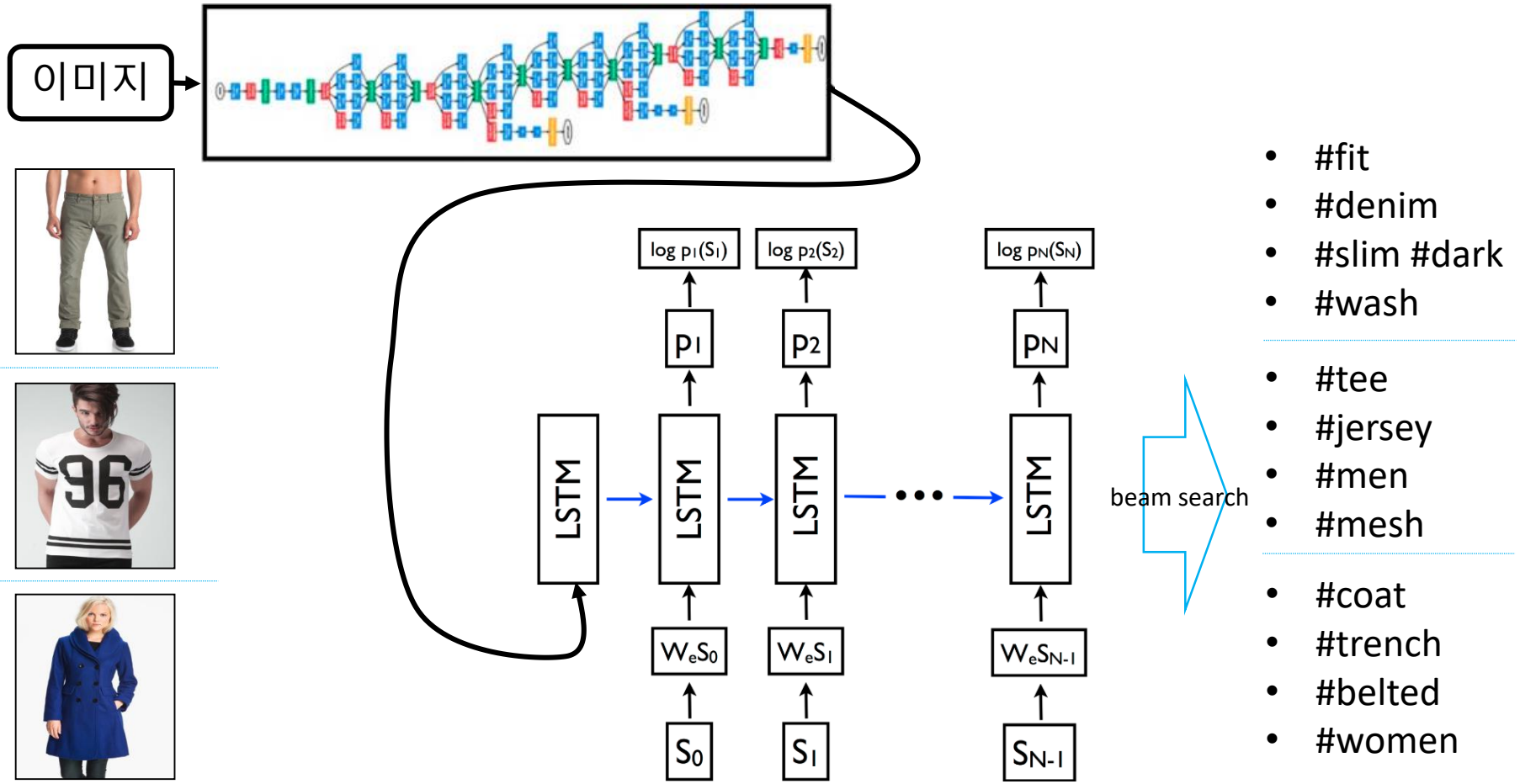
이미지 유사도 검색

- 상품 이미지

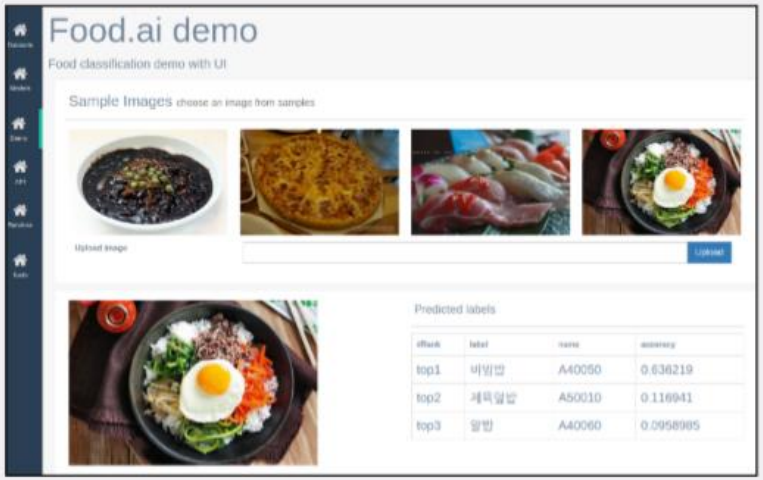


텍스트 기반 속성 식별

- CNN-RNN 모델 기반 이미지 속성 식별



음식 이미지 분석



음식명	1회 제공량 (g)	열량 (kcal)	탄수화물 (g)	단백질 (g)	지방 (g)	당류 (g)	나트륨 (g)
영양밥	174	321.41	63.61	6.3	5.06	-	101.7
비빔밥	500	704.62	114.83	19.92	18.62	25.66	1377.1
자장면	650	796.5	133.63	19.77	20.32	7.79	2391.58