

CNN implementation

정보시스템공학과

18211501

안규황



CNN - Tensorflow

Python TensorFlow 튜토리얼 x Homework - Google 프로그래밍 x Convolutional Neural Networks x Keras tutorial - build a convolutional neural network x adventures-in-ml-code/ x Students - Google 드라이브 x

adventuresinmachinelearning.com/convolutional-neural-networks-tutorial-tensorflow/

Bookmarks 대학원 학과사무실_IT융합실 CryptoCraft Lab Blockchain관련 NSR_SHA-3 NSR_암호전문인력인 논문작성관련 Youtube git adventuresinML/adv TensorFlow 시작하기

ADVENTURES IN MACHINE LEARNING

LEARN AND EXPLORE MACHINE LEARNING

ABOUT CONTACT

Convolutional Neural Networks Tutorial in TensorFlow

April 24, 2017 Andy Convolutional Neural Networks, Deep learning, TensorFlow 12

Feature maps

f.maps

f.maps

Convolutions

Subsampling

Subsampling

Full convolutional neural network

In the two previous tutorial posts, [an introduction to neural networks](#) and [an introduction to](#)

POPULAR TUTORIALS

- Neural Networks Tutorial – A Pathway to Deep Learning
- Python TensorFlow Tutorial – Build a Neural Network
- Convolutional Neural Networks Tutorial in TensorFlow
- Keras tutorial – build a convolutional neural network in 11 lines
- Word2Vec word embedding tutorial in Python and TensorFlow

CATEGORIES

- Amazon AWS
- CNTK
- Convolutional Neural Networks
- Deep learning
- gensim
- GPUs

CNN - Tensorflow

This convolutional neural networks tutorial will introduce these networks by building them in TensorFlow. If you're not familiar with TensorFlow, I'd suggest checking out my previously mentioned tutorial, which is a gentle introduction. Otherwise, you're welcome to wing it. Another option is to build the convolutional neural network in Keras, which is more syntactically stream-lined – you can see how to do this my brief Keras tutorial.

CNN - Tensorflow

Python TensorFlow Tutorial

Homework - Google

Convolutional Neural Ne

Keras tutorial - build a c

adventures-in-ml-code/c


Students - Google 드라

adventuresinmachinelearning.com/python-tensorflow-tutorial/

Bookmarks 대학원 학과사무실_IT융합공 CryptoCraft Lab Blockchain관련 NSR_SHA-3 NSR_암호전문인력인 논문작성관련 Youtube git adventuresinML/ad TensorFlow 시작하

Python TensorFlow Tutorial – Build a Neural Network

April 8, 2017 Andy Deep learning, Neural networks, TensorFlow 19



The TensorFlow logo

Google's TensorFlow has been a hot topic in deep learning recently. The open source software, designed to allow efficient computation of data flow graphs, is especially suited to deep learning tasks. It is designed to be executed on single or multiple CPUs and GPUs, making it a good option for complex deep learning tasks. In it's most recent incarnation – version 1.0 – it can even be run on certain mobile operating systems. This introductory tutorial to TensorFlow will give an overview of some of the basic concepts of TensorFlow in Python. These will be a good stepping stone to building more complex deep learning networks, such as **Convolution Neural Networks**, **natural language models** and **Recurrent**

POPULAR TUTORIALS

- Neural Networks Tutorial – A Pathway to Deep Learning
- Python TensorFlow Tutorial – Build a Neural Network
- Convolutional Neural Networks Tutorial in TensorFlow
- Keras tutorial – build a convolutional neural network in 11 lines
- Word2Vec word embedding tutorial in Python and TensorFlow

CATEGORIES

- Amazon AWS
- CNTK
- Convolutional Neural Networks
- Deep learning
- gensim
- GPUs
- Keras
- LSTMs
- Neural networks
- NLP
- Optimisation

206 Shares

f 182

p 3

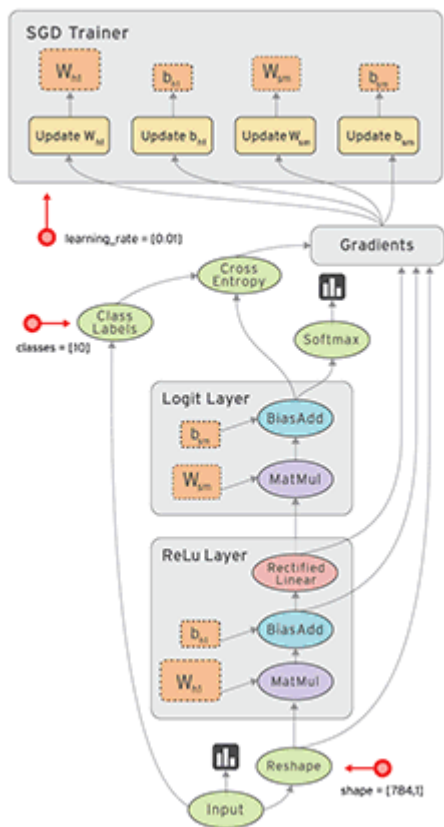
t

G+

in

HANSUNG Since 1945 한성대학교

CNN - Tensorflow



- Tensorflow 란?
 - 기계학습과 딥러닝을 위해 구글에서 만든 Opensource library
- 데이터 플로우 그래프
 - 데이터 플로우 그래프는 수학 계산과 데이터의 흐름을 노드(Node)와 엣지(Edge)를 사용한 방향 그래프(Directed Graph)로 표현
- 노드는 수학적 계산, 데이터 입/출력, 그리고 데이터의 읽기/저장 등의 작업을 수행
- 엣지는 노드들 간 데이터의 입출력 관계를 표현

Simple Tensorflow Example

```
$ python
```

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

Simple Tensorflow Example

```
import tensorflow as tf

# 변수를 0으로 초기화
state = tf.Variable(0, name="counter")

# state에 1을 더할 오퍼레이션 생성
one = tf.constant(1)
new_value = tf.add(state, one)
update = tf.assign(state, new_value)

# 그래프는 처음에 변수를 초기화해야 합니다. 아래 함수를 통해 init 오퍼레이션을 만듭니다.
init_op = tf.initialize_all_variables()

# 그래프를 띄우고 오퍼레이션들을 실행
with tf.Session() as sess:
    # 초기화 오퍼레이션 실행
    sess.run(init_op)
    # state의 초기 값을 출력
    print(sess.run(state))
    # state를 갱신하는 오퍼레이션을 실행하고, state를 출력
    for _ in range(3):
        sess.run(update)
        print(sess.run(state))
```

텐서(Tensor)

내부적으로 모든 데이터는 텐서를 통해 표현됩니다. 텐서는 일종의 다차원 배열인데, 그래프 내의 오퍼레이션 간에는 텐서만이 전달됩니다.

변수(Variables)

변수는 그래프의 실행시, 패러미터를 저장하고 갱신하는데 사용됩니다. 메모리 상에서 텐서를 저장하는 버퍼 역할을 합니다.

오퍼레이션(Operation)

그래프 상의 노드는 오퍼레이션(줄임말 *op*)으로 불립니다. 오퍼레이션은 하나 이상의 *텐서*를 받을 수 있습니다. 오퍼레이션은 계산을 수행하고, 결과를 하나 이상의 텐서로 반환할 수 있습니다.

세션(Session)

그래프를 실행하기 위해서는 세션 객체가 필요합니다. 세션은 오퍼레이션의 실행 환경을 캡슐화한 것입니다.

결과

0
1
2
3

MNIST



MNIST

```
# MNIST 데이터 가져옴
```

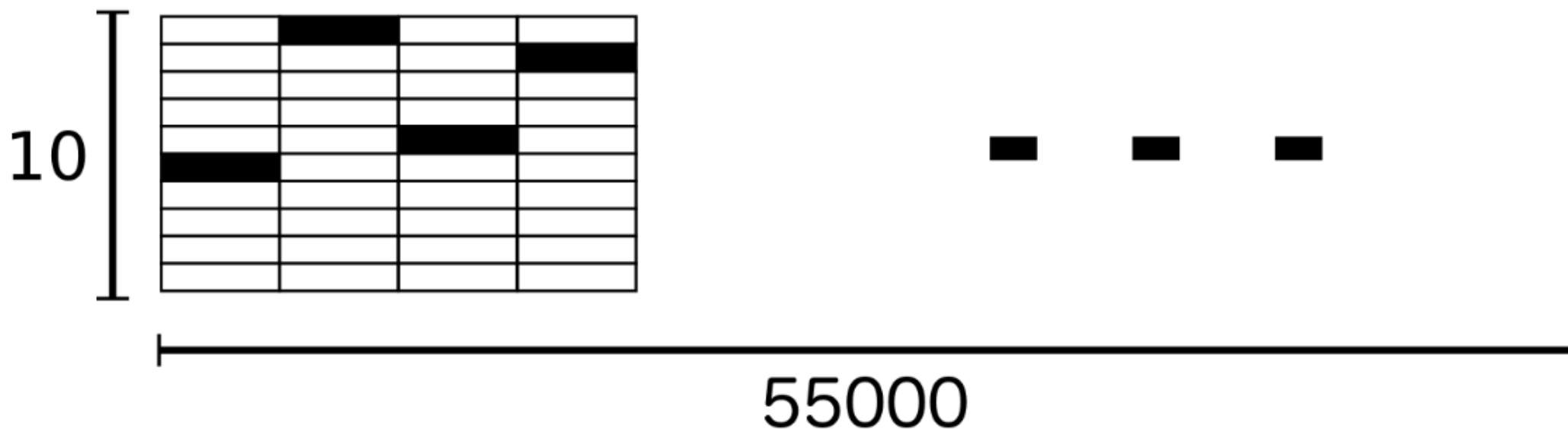
```
import input_data
```

```
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
```

```
mnist = input_data.read_data_sets("./samples/MNIST_data/", one_hot=True)
```

MNIST

mnist.train.ys



MNIST에는 55,000개의 학습용 이미지 + 10,000개의 테스트 이미지 + 5,000개의 검증 이미지가 있습니다. 각 이미지는 28x28 크기를 가집니다. 이것을 펼치면 784 차원의 벡터가 됩니다.

CNN – Tensorflow – Epoch 1

```
# 이미지 데이터 플레이스홀더  
x = tf.placeholder("float", [None, 784])  
# 웨이트와 바이어스 변수  
W = tf.Variable(tf.zeros([784,10]))  
b = tf.Variable(tf.zeros([10]))  
# 모델 구현  
y = tf.nn.softmax(tf.matmul(x,W) + b)
```

CNN – Tensorflow – Epoch 1

모든 변수 초기화

```
init = tf.initialize_all_variables()
```

```
sess = tf.Session()
```

```
sess.run(init)
```

CNN – Tensorflow – Epoch 1

```
# 임의로 100개 샘플링
```

```
for i in range(1000):
```

```
    batch_xs, batch_ys = mnist.train.next_batch(100)
```

```
    sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})
```

```
# 정답율
```

```
correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))
```

```
accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
```

```
print (sess.run(accuracy, feed_dict={x: mnist.test.images, y_: mnist.test.labels}))
```

CNN – Tensorflow – Epoch 1

```
In [22]: runfile('C:/Users/kyu/untitled.py', wdir='C:/Users/kyu')  
Extracting ./samples/MNIST_data/train-images-idx3-ubyte.gz  
Extracting ./samples/MNIST_data/train-labels-idx1-ubyte.gz  
Extracting ./samples/MNIST_data/t10k-images-idx3-ubyte.gz  
Extracting ./samples/MNIST_data/t10k-labels-idx1-ubyte.gz  
0.919
```

CNN – Tensorflow – Epoch 10

```
Epoch: 1 cost = 0.739 test accuracy: 0.911  
Epoch: 2 cost = 0.169 test accuracy: 0.960  
Epoch: 3 cost = 0.100 test accuracy: 0.978  
Epoch: 4 cost = 0.074 test accuracy: 0.979  
Epoch: 5 cost = 0.057 test accuracy: 0.984  
Epoch: 6 cost = 0.047 test accuracy: 0.984  
Epoch: 7 cost = 0.040 test accuracy: 0.986  
Epoch: 8 cost = 0.034 test accuracy: 0.986  
Epoch: 9 cost = 0.029 test accuracy: 0.989  
Epoch: 10 cost = 0.025 test accuracy: 0.990
```

```
Training complete!  
0.9897
```

CNN - Keras

```
from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.layers import Dense, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.models import Sequential
import matplotlib.pyplot as plt
```

```
#CNN structure variable
```

```
batch_size = 128
```

```
num_classes = 10
```

```
epochs = 7
```

```
# input image dimensions
```

```
img_x, img_y = 28, 28 //MNIST 각 이미지 크기가 28x28 이기 때문에
```


CNN - Keras

열차와 테스트 세트로 이미 나뉘어있는 MNIST 데이터 세트를로드합니다.

```
(x_train, y_train), (x_test, y_test) = mnist.load_data ()
```

데이터를 4D 텐서로 바꾼다 - (sample_number, x_img_size, y_img_size, num_channels)

왜냐하면 MNIST가 그레이 스케일이기 때문에 우리는 하나의 채널만을 가지고 있습니다 - RGB 컬러 이미지는 3

```
x_train = x_train.reshape (x_train.shape [ 0 ], img_x, img_y, 1 )
```

```
x_test = x_test.reshape (x_test.shape [ 0 ], img_x, img_y, 1 )
```

```
input_shape = (img_x, img_y, 1 )
```

CNN - Keras

```
# convert the data to the right type // MNIST to CNN input  
x_train = x_train.astype('float32')  
x_test = x_test.astype('float32')  
x_train /= 255  
x_test /= 255  
print('x_train shape:', x_train.shape)  
print(x_train.shape[0], 'train samples')  
print(x_test.shape[0], 'test samples')
```

CNN - Keras

```
#CNN modeling (add layer)
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5), strides=(1, 1),
                activation='relu',
                input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Conv2D(64, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(1000, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))

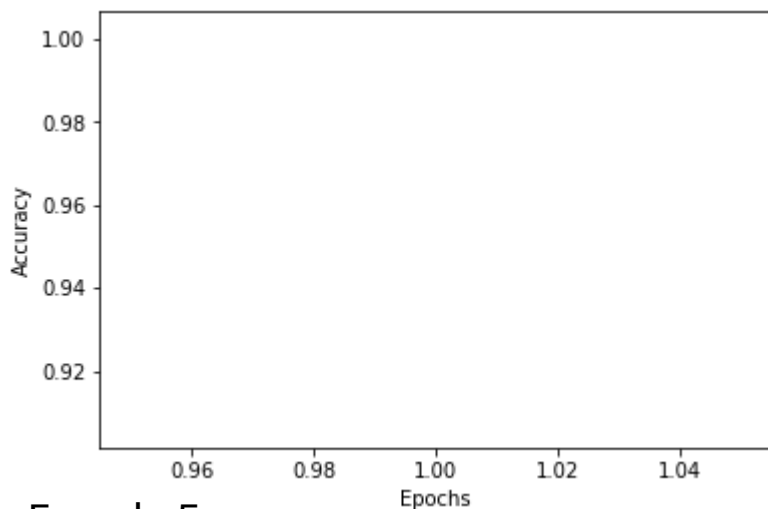
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adam(),
              metrics=['accuracy'])
```

CNN - Keras

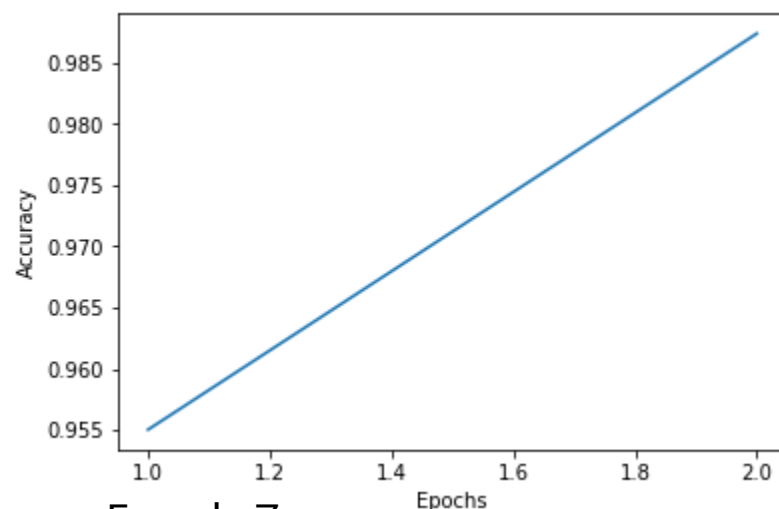
```
#training set
model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test),
          callbacks=[history])
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
plt.plot(range(1, 8), history.acc)
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.show()
```

CNN - Keras

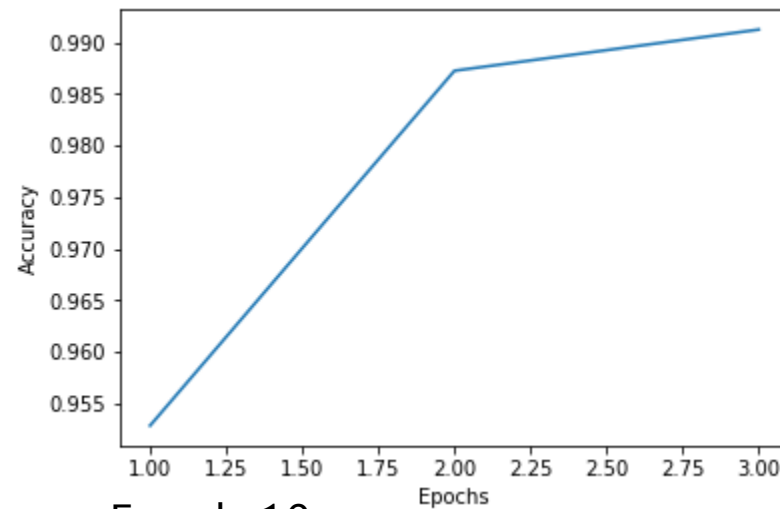
Epoch 1



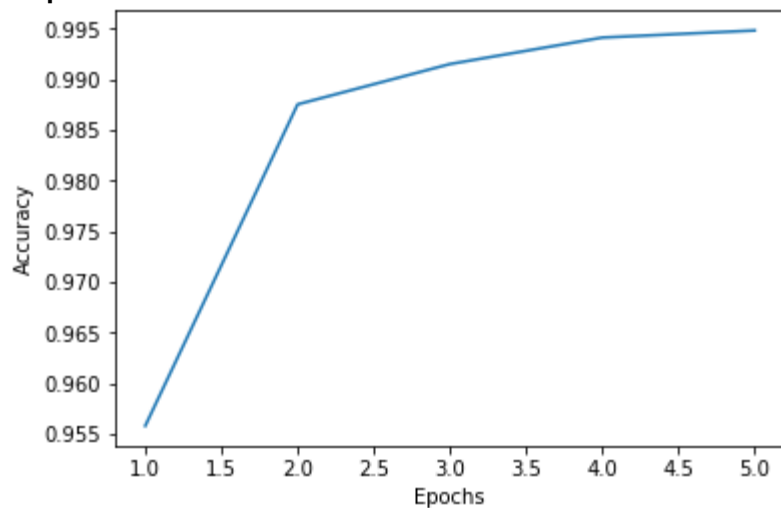
Epoch 2



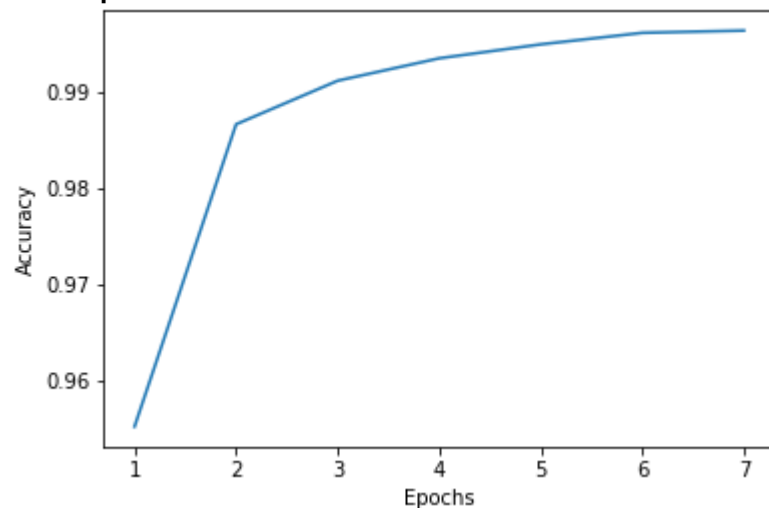
Epoch 3



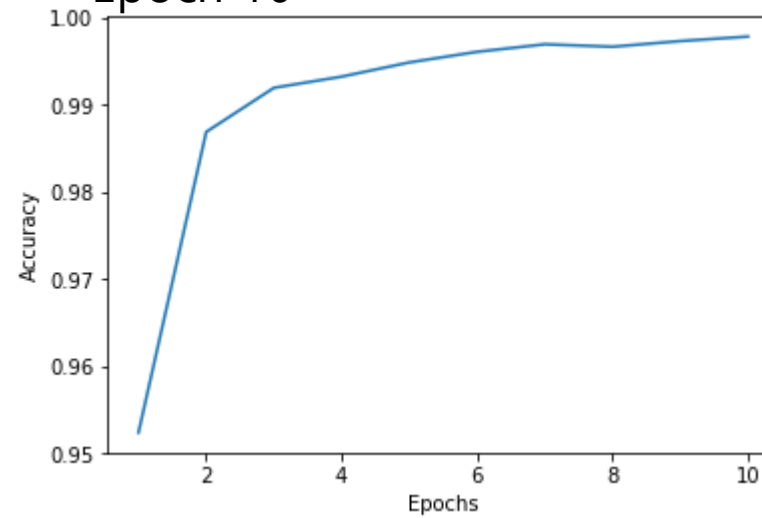
Epoch 5



Epoch 7



Epoch 10



Tensorflow vs keras

케라스 사용자에서의 의미

텐서플로우-케라스는 텐서플로우 코어 위에서 만들어졌기 때문에

- 순수 텐서플로우 기능과 케라스 기능이 쉽게 섞여지고 매칭됩니다.
- 케라스 사용자는 다음의 텐서플로우의 기능을 사용할 수 있습니다.
 - Distributed training
 - Cloud ML
 - Hyperparameter tuning
 - TF-Serving

텐서플로우 사용자에서의 의미

- 모델 정의할 때 케라스의 고차원 API를 사용할 수 있습니다.
- 텐서플로우 코어와 tf.keras와의 깊은 호환성 때문에 유연성에 대한 손실이 없습니다.
- Experiment, Cloud ML, TF-Serving와 같은 당신의 TF workflow를 기존의 케라스 코드에 쉽게 적용가능합니다.

감사합니다.
