TensorFlow 모델을 저장하고 불러오기 (save and restore)

2017.12.09 12:25

TensorFlow 모델을 저장하고 불러오기 (save and restore)

해당 튜토리얼에 사용한 코드는 개인 GitHub Link에서 확인 할 수 있다.

저장 복구를 위해서는 두개의 파일이 필요하다.

a) Meta graph

Tensorflow graph를 저장 하게 된다. 즉 all variables, operations, collections 등을 저장 한다. `.meta` 로 확장자를 가진다.

b) Checkpoint file

binary 파일로 weights, biases, gradients 등을 저장 한다.

`0.11` 부터는 두개의 파일로 저장된다.

- `model.ckpt.data-00000-of-00001`
- `model.ckpt.index`

`.data` 파일의 경우 training variable를 가지고 있다.

여전히 `checkpoint` 파일도 보유하고 있지만 이것은 단순히 최근 상태만을 기록하고 있다.

모델 저장 방법

`saver = tf.train.Saver()` 를 통해서 가능 하다.

```
import tensorflow as tf
w1 = tf.Variable(tf.random_normal(shape=[2]), name='w1')
w2 = tf.Variable(tf.random_normal(shape=[5]), name='w2')
saver = tf.train.Saver()
```

```
sess = tf.Session()
sess.run(tf.global_variables_initializer())
saver.save(sess, 'my_test_model')

# This will save following files in Tensorflow v >= 0.11
# my_test_model.data-00000-of-00001
# my_test_model.index
# my_test_model.meta
# checkpoint
```

만약 `1000` interations 이후에 model을 저장하고 싶다면 아래와 같이 한다.

```
saver.save(sess, "my_test_model", global_step=1000)
```

이렇게 하면 아래처럼 모델뒤에 `-1000` 이라는 숫자가 이름에 붙어서 나오게 됩니다.

```
my_test_model-1000.index
my_test_model-1000.meta
my_test_model-1000.data-00000-of-00001
checkpoint
```

모델의 구조는 같기 때문에 `.meta` 파일의 경우 1000 interation당 매번 생성할 필요는 없다.

모델 값만 저장하고 graph는 저장하지 않는 코드는 아래와 같다.

```
saver.save(sess, 'my-model', global_step=step,write_meta_graph=False)
```

만약 최근 2시간동안 4개의 모델만 저장하고 싶다면 아래와 같이 옵션을 설정한다.

```
#saves a model every 2 hours and maximum 4 latest models are saved.
saver = tf.train.Saver(max_to_keep=4, keep_checkpoint_every_n_hours=2)
```

망약 `tf.train.Saver()` 에 아무것도 지정하지 않았다면 모든 변수들을 저장하게 된다.

특정 variables/collections을 저장하고 싶다면 그것을 지정하면 된다.

`List` , `dictionary` 자료구조도 받으니 그것을 잘 활용 한다.

```
import tensorflow as tf
w1 = tf.Variable(tf.random_normal(shape=[2]), name='w1')
w2 = tf.Variable(tf.random_normal(shape=[5]), name='w2')
saver = tf.train.Saver([w1,w2])
sess = tf.Session()
sess.run(tf.global_variables_initializer())
saver.save(sess, 'my_test_model',global_step=1000)
```

모델 읽기

두 가지 일을 해야한다.

a) 네트워크 생성

`.meta` 파일을 생성 했으므로 이것을 불러오는 방식으로 network을 재생성 할 수 있다.

`.meta` 파일을 불러오기 위해서는 `tf.train.import()` 함수를 이용한다.

```
saver = tf.train.import_meta_graph('my_test_model-1000.meta')
```

이렇게 하면 현재 그래프에 이어 붙는 형식으로 동작하므로 `tf.reset_default_graph()` 를 실행해서 default graph로 초기화 해주는 것이 안전하다.

b) 파라메터 로딩

`tf.train.Saver()` 를 이용해서 파라메터를 로딩한다.

```
with tf.Session() as sess:
  new_saver = tf.train.import_meta_graph('my_test_model-1000.meta')
  new_saver.restore(sess, tf.train.latest_checkpoint('./'))
```

```
with tf.Session() as sess:
    saver = tf.train.import_meta_graph('my-model-1000.meta')
    saver.restore(sess, tf.train.latest_checkpoint('./'))
    print(sess.run('w1:0'))
##Model has been restored. Above statement will print the saved value of w1.
```

저장된 모델로 실제 작업하기

이제 위해서 다룬 내용을 토대로 종합적으로 간단한 neural net.을 생성하고 이것을 저장한다음 다시 불러오는 코드를 작성해 본다.

이런한 작업은 추후에 `transfer learning` 이나 `testing` 만 별도로 작업하기 위해서 사용 될 수 있다.

아래의 코드는 $y = (w1 + w2) \times b$ 를 구현한 내용이다.

여기서 핵심은 추후에 `variable` 과 `operation` 을 각각 불러오기 위해서 `name` 을 반드시 주어야 한다.

나중에 본인 model을 공유할 때도 이름을 잘 정해주어야 다른 사람이 가져다가 본인들 목적에 맞춰서 `fine-tuning` 해서 사용할 수 있다.

모델 생성 및 저장

```
import tensorflow as tf

# Prepare to feed input, i.e. feed_dict and placeholders
w1 = tf.placeholder(tf.float32, name="w1")
w2 = tf.placeholder(tf.float32, name="w2")
b1 = tf.Variable(2.0,dtype=tf.float32, name="bias")
feed_dict = {'w1': 4.0, 'w2': 8.0}

# Define a test operation that we will restore
w3 = w1 + w2
w4 = tf.multiply(w3, b1, name="op_to_restore")
sess = tf.Session()
sess.run(tf.global_variables_initializer())

# Create a saver object which will save all the variables
saver = tf.train.Saver()

# Run the operation by feeding input
result = sess.run(w4, {w1:feed_dict['w1'], w2:feed_dict['w2']})
print(result)
# Prints 24 which is sum of (w1+w2)*b1

# Now, save the graph
saver.save(sess, './my_test_model', global_step=1000)
```

실행결과는 `24` 이다.

모델 불러오기와서 새로운 입력값으로 처리

모델을 복구하고 `feed_dict` 을 다르게 입력하는 코드이다.

```
import tensorflow as tf

sess=tf.Session()
#First let's load meta graph and restore weights
saver = tf.train.import_meta_graph('my_test_model-1000.meta')
saver.restore(sess,tf.train.latest_checkpoint('./'))


# Now, let's access and create placeholders variables and
# create feed-dict to feed new data

graph = tf.get_default_graph()
w1 = graph.get_tensor_by_name("w1:0")
w2 = graph.get_tensor_by_name("w2:0")
feed_dict ={w1:13.0,w2:17.0}

#Now, access the op that you want to run.
op_to_restore = graph.get_tensor_by_name("op_to_restore:0")

print (sess.run(op_to_restore,feed_dict))
```

```
#This will print 60 which is calculated
#using new values of w1 and w2 and saved value of b1.
```

실행결과는 `60` 이다.

위 예제는 `Tensor` 를 이용해서 간단하게 구현하다보니 모두 `get_tensor_by_name()` 으로 가능하지만 실제로 operation과 placeholder는 각각 다르게 load해야 한다.

- placeholder 불러오기
  - `graph.get_tensor_by_name()`
- operation 불러오기
  - `graph.get_operation_by_name()`

로딩 가능한 현재 graph에서의 operation의 종류이다.

```
for op in tf.get_default_graph().get_operations():
    print(op.name)
```

모델을 불러오고 operation과 layer 추가

```
import tensorflow as tf

sess=tf.Session()
#First let's load meta graph and restore weights
saver = tf.train.import_meta_graph('my_test_model-1000.meta')
saver.restore(sess,tf.train.latest_checkpoint('./'))


# Now, let's access and create placeholders variables and
# create feed-dict to feed new data

graph = tf.get_default_graph()
w1 = graph.get_tensor_by_name("w1:0")
w2 = graph.get_tensor_by_name("w2:0")
feed_dict ={w1:13.0,w2:17.0}

#Now, access the op that you want to run.
op_to_restore = graph.get_tensor_by_name("op_to_restore:0")

#Add more to the current graph
add_on_op = tf.multiply(op_to_restore,2)

print (sess.run(add_on_op,feed_dict))
#This will print 120.
```

실행결과 `120` 이다.

참고자료

http://cv-tricks.com/tensorflow-tutorial/save-restore-tensorflow-models-quick-complete-tutorial/

1

'Data Science > TensorFlow and Scikit-Learn' 카테고리의 다른 글

윈도우 GPU tensorflow 설치 및 그래픽카드별 성능 비교 (49)	2018.03.27
학습 모델의 재사용 (Transfer Learning) (0)	2017.12.09
TensorFlow 모델을 저장하고 불러오기 (save and restore) (2)	2017.12.09
Batch 크기의 결정 방법 (0)	2017.12.07
Neural Network 적용 전에 Input data를 Normalize 해야 하는 이유 (3)	2017.12.07
TensorFlow GPU 버전 우분투 16.04에 설치 하기 (7)	2017.11.26

SECRET WRITE

텐플론잉 *2018.03.28 16:37 신고*

안녕하세요 제가 참고해서 모델 로드를 하려고 하는데요
with tf.Session() as sess:
new_saver = tf.train.import_meta_graph('my_test_model-1000.meta')
new_saver.restore(sess, tf.train.latest_checkpoint('./'))

위의 코드와 같이 작성하여 restore를 할 때
ValueError: Can't load save_path when it is None. 와 같은 에러메시지가 뜨는데요 구글링해도 만족할만한 해결방법이 안 나와서 여쭤 봅니다. 혹시 이에 관한 경험이나 아시는 부분이 있을까요?
감사합니다

Delete Reply

텐플론잉 *2018.03.28 16:52 신고*

다른 모델을 로드했더니 잘 돌아가네요.. 저장된 모델에 문제가 있었나봅니다

Delete

PREV 1 ··· 16 17 18 19 20 21 22 23 24 ··· 437 NEXT

+ Recent posts

Pycharm으로 TensorFlow 원격..

Jupyter 서버 설치 및 실행법