Tensorflow: restoring a graph and model then running evaluation on a single image

I think it would be immensely helpful to the Tensorflow community if there was a well-documented solution to the crucial task of testing a single new image against the model created by the convnet in the CIFAR-10 tutorial.

I may be wrong, but this critical step that makes the trained model usable in practice seems to be lacking. There is a "missing link" in that tutorial—a script that would directly load a single image (as array or binary), compare it against the trained model, and return a classification.

Prior answers give partial solutions that explain the overall approach, but none of which I've been able to implement successfully. Other bits and pieces can be found here and there, but unfortunately haven't added up to a working solution. Kindly consider the research I've done, before tagging this as duplicate or already answered.

Tensorflow: How to restore a previously saved model (python)

Restoring TensorFlow model

Unable to restore models in tensorflow v0.8

https://gist.github.com/nikitakit/6ef3b72be67b86cb7868

The most popular answer is the first, in which @RyanSepassi and @YaroslavBulatov describe the problem and an approach: one needs to "manually construct a graph with identical node names, and use Saver to load the weights into it". Although both answers are helpful, it is not apparent how one would go about plugging this into the CIFAR-10 project.

A fully functional solution would be highly desirable so we could port it to other single image classification problems. There are several questions on SO in this regard that ask for this, but still no full answer (for example Load checkpoint and evaluate single image with tensorflow DNN).

I hope we can converge on a working script that everyone could use.

The below script is not yet functional, and I'd be happy to hear from you on how this can be improved to provide a solution for single-image classification using the CIFAR-10 TF tutorial trained model.

Assume all variables, file names etc. are untouched from the original tutorial.

New file: **cifar10_eval_single.py**

```
import cv2
import tensorflow as tf

FLAGS = tf.app.flags.FLAGS

tf.app.flags.DEFINE_string('eval_dir', './input/eval',
                           """Directory where to write event logs.""")
tf.app.flags.DEFINE_string('checkpoint_dir', './input/train',
                           """Directory where to read model checkpoints.""")

def get_single_img():
    file_path = './input/data/single/test_image.tif'
    pixels = cv2.imread(file_path, 0)
    return pixels

def eval_single_img():

    # below code adapted from @RyanSepassi, however not functional
    # among other errors, saver throws an error that there are no
    # variables to save
    with tf.Graph().as_default():
        # Get image.
        image = get_single_img()

        # Build a Graph.
        # TODO

        # Create dummy variables.
        x = tf.placeholder(tf.float32)
        w = tf.Variable(tf.zeros([1, 1], dtype=tf.float32))
        b = tf.Variable(tf.ones([1, 1], dtype=tf.float32))
        y_hat = tf.add(b, tf.matmul(x, w))

        saver = tf.train.Saver()

        with tf.Session() as sess:
            sess.run(tf.initialize_all_variables())
            ckpt = tf.train.get_checkpoint_state(FLAGS.checkpoint_dir)

            if ckpt and ckpt.model_checkpoint_path:
                saver.restore(sess, ckpt.model_checkpoint_path)
                print('Checkpoint found')
            else:
                print('No checkpoint found')

            # Run the model to get predictions
            predictions = sess.run(y_hat, feed_dict={x: image})
            print(predictions)

def main(argv=None):
    if tf.gfile.Exists(FLAGS.eval_dir):
        tf.gfile.DeleteRecursively(FLAGS.eval_dir)
    tf.gfile.MakeDirs(FLAGS.eval_dir)
    eval_single_img()

if __name__ == '__main__':
    tf.app.run()
```

python python-3.x machine-learning tensorflow

asked Jun 6 '16 at 3:14

pepe
3,517 15 75 137

3 Answers

Here's how I ran a single image at a time. I'll admit it seems a bit hacky with the reuse of getting the scope.

This is a helper function

```
def restore_vars(saver, sess, chkpt_dir):
""" Restore saved net, global score and step, and epsilons OR
create checkpoint directory for later storage. """
sess.run(tf.initialize_all_variables())

checkpoint_dir = chkpt_dir

if not os.path.exists(checkpoint_dir):
    try:
        os.makedirs(checkpoint_dir)
    except OSError:
        pass

path = tf.train.get_checkpoint_state(checkpoint_dir)
#print("path1 = ",path)
#path = tf.train.latest_checkpoint(checkpoint_dir)
print(checkpoint_dir,"path = ",path)
if path is None:
    return False
else:
    saver.restore(sess, path.model_checkpoint_path)
    return True
```

Here is the main part of the code that runs a single image at a time within the for loop.

```
to_restore = True
with tf.Session() as sess:

    for i in range(test_img_set):

            # Gets the image
            images = get_image(i)
            images = np.asarray(images,dtype=np.float32)
            images = tf.convert_to_tensor(images/255.0)
            # resize image to whatever you're model takes in
            images = tf.image.resize_images(images,256,256)
            images = tf.reshape(images,(1,256,256,3))
            images = tf.cast(images, tf.float32)

            saver = tf.train.Saver(max_to_keep=5, keep_checkpoint_every_n_hours=1)
            #print("infer")
            with tf.variable_scope(tf.get_variable_scope()) as scope:
                if to_restore:
                    logits = inference(images)
                else:
                    scope.reuse_variables()
                    logits = inference(images)


            if to_restore:
                restored = restore_vars(saver, sess,FLAGS.train_dir)
                print("restored ",restored)
                to_restore = False

            logit_val = sess.run(logits)
            print(logit_val)
```

Here is an alternative implementation to the above using place holders it's a bit cleaner in my opinion. but I'll leave the above example for historical reasons.

```
imgs_place = tf.placeholder(tf.float32, shape=[my_img_shape_put_here])
images = tf.reshape(imgs_place,(1,256,256,3))

saver = tf.train.Saver(max_to_keep=5, keep_checkpoint_every_n_hours=1)

#print("infer")
logits = inference(images)

restored = restore_vars(saver, sess,FLAGS.train_dir)
print("restored ",restored)

with tf.Session() as sess:
    for i in range(test_img_set):
        logit_val = sess.run(logits,feed_dict={imgs_place=i})
        print(logit_val)
```

edited Jul 28 '16 at 19:50

answered Jun 9 '16 at 23:57

Steven
1,752 1 4 21

got it working with this

```
softmax = gn.inference(image)
saver = tf.train.Saver()
ckpt = tf.train.get_checkpoint_state(FLAGS.checkpoint_dir)
with tf.Session() as sess:
  saver.restore(sess, ckpt.model_checkpoint_path)
  softmaxval = sess.run(softmax)
  print(softmaxval)
```

output

```
[[  6.73550041e-03   4.44930716e-04   9.92570221e-01   1.00681427e-06
    3.05406687e-08   2.38927707e-04   1.89839399e-12   9.36238484e-06
    1.51646684e-09   3.38977535e-09]]
```

restores a model and does inference on a single mnist-style image

https://github.com/noahhsmith/starid/blob/master/learning-based/lb.py

restores a model and does inference on a batch of mnist-style images from a tfrecords file

https://github.com/noahhsmith/starid/blob/master/learning-based/predict.py

edited Dec 31 '16 at 9:58

answered Dec 27 '16 at 19:33

Noah Smith
28 1 7

I don't have working code for you I'm afraid, but here's how we often tackle this problem in production:

- Save out the GraphDef to disk, using something like write_graph.
- Use freeze_graph to load the GraphDef and checkpoints, and save out a GraphDef with the Variables converted into Constants.
- Load the GraphDef in something like label_image or classify_image.

For your example this is overkill, but I would at least suggest serializing the graph in the original example as a GraphDef, and then loading it in your script (so you don't have to duplicate the code generating the graph). With the same graph created, you should be able to populate it from a SaverDef, and the freeze_graph script may help as an example.

answered Jun 7 '16 at 1:38

Pete Warden
1,157 2 4