Deploying Keras Models via Google Cloud ML

I am looking to use Google Cloud ML to host my Keras models so that I can call the API and make some predictions. I am running into some issues from the Keras side of things.

So far I have been able to build a model using TensorFlow and deploy it on CloudML. In order for this to work I had to make some changes to my basic TF code. The changes are documented here: https://cloud.google.com/ml/docs/how-tos/preparing-models#code_changes

I have also been able to train a similar model using Keras. I can even save the model in the same export and export.meta format as I would get with TF.

```
from keras import backend as K

saver = tf.train.Saver()
session = K.get_session()
saver.save(session, 'export')
```

The part I am missing is how do I add the placeholders for input and output into the graph I build on Keras?

tensorflow google-cloud-platform keras google-cloud-ml

edited Jan 31 '17 at 17:18
Kilian Stinson
2,014 18 28

asked Jan 31 '17 at 13:52
Matthew Jackson
36 1 5

named the input and output of my graph with

```
signature = predict_signature_def(inputs={'NAME_YOUR_INPUT': new_Model.input},
                                  outputs={'NAME_YOUR_OUTPUT': new_Model.output})
```

You can see the full exporting example for an already trained keras model 'model.h5' below.

```
import keras.backend as K
import tensorflow as tf
from keras.models import load_model, Sequential
from tensorflow.python.saved_model import builder as saved_model_builder
from tensorflow.python.saved_model import tag_constants, signature_constants
from tensorflow.python.saved_model.signature_def_utils_impl import
predict_signature_def

# reset session
K.clear_session()
sess = tf.Session()
K.set_session(sess)

# disable loading of learning nodes
K.set_learning_phase(0)

# load model
model = load_model('model.h5')
config = model.get_config()
weights = model.get_weights()
new_Model = Sequential.from_config(config)
new_Model.set_weights(weights)

# export saved model
export_path = 'YOUR_EXPORT_PATH' + '/export'
builder = saved_model_builder.SavedModelBuilder(export_path)

signature = predict_signature_def(inputs={'NAME_YOUR_INPUT': new_Model.input},
                                  outputs={'NAME_YOUR_OUTPUT': new_Model.output})

with K.get_session() as sess:
    builder.add_meta_graph_and_variables(sess=sess,
                                         tags=[tag_constants.SERVING],
                                         signature_def_map={

signature_constants.DEFAULT_SERVING_SIGNATURE_DEF_KEY: signature})
```

edited Nov 2 '17 at 10:58

answered May 28 '17 at 22:01

Lausbert
143 1 2 11

1 The full implementation link is broken. I think this might be where it's supposed to be now, but I can't even find any Python code there. – aldel Nov 1 '17 at 19:27

fixed, thanks for your hint :) – Lausbert Nov 2 '17 at 10:59

I found out that in order to use keras on google cloud one has to install it with a setup.py script and put it on the same place folder where you run the gcloud command:

```
├── setup.py
└── trainer
    ├── __init__.py
    ├── cloudml-gpu.yaml
    ├── example5-keras.py
```

And in the setup.py you put content such as:

```
from setuptools import setup, find_packages

setup(name='example5',
  version='0.1',
  packages=find_packages(),
  description='example to run keras on gcloud ml-engine',
  author='Fuyang Liu',
  author_email='fuyang.liu@example.com',
  license='MIT',
  install_requires=[
      'keras',
      'h5py'
  ],
  zip_safe=False)
```

Then you can start your job running on gcloud such as:

```
gcloud ml-engine jobs submit training $JOB_NAME \
  --job-dir gs://$BUCKET_NAME/$JOB_NAME \
  --runtime-version 1.0 \
  --module-name trainer.example5-keras \
  --package-path ./trainer \
  --region $REGION \
  --config=trainer/cloudml-gpu.yaml \
  -- \
  --train-file gs://tf-learn-simple-sentiment/sentiment_set.pickle
```

To use GPU then add a file such as `cloudml-gpu.yaml` in your module with the following content:

```
trainingInput:
  scaleTier: CUSTOM
  # standard_gpu provides 1 GPU. Change to complex_model_m_gpu for 4
GPUs
  masterType: standard_gpu
  runtimeVersion: "1.0"
```

edited Apr 6 '17 at 11:29

answered Apr 5 '17 at 23:13

Fuyang Liu
181 8

These are great instructions for running training, but the original question is about serving. Do you mind posting a new question about how to train Keras models on CloudML and self-answering with this valuable information? – rhaertel80 Apr 6 '17 at 14:33

Oh, that totally makes sense. I will do that :) – Fuyang Liu Apr 10 '17 at 12:10

I don't know much about Keras. I consulted with some experts, and the following should work:

```
from keras import backend as k

# Build the model first
model = ...
```

```
outputs = dict(zip((layer.name for layer in model.output_layers),
                   (t.name for t in model.outputs)))
tf.add_to_collection('outputs', json.dumps(outputs))

# Fit/train the model
model.fit(...)

# Export the model
saver = tf.train.Saver()
session = K.get_session()
saver.save(session, 'export')
```

Some important points:

- You have to call tf.add_to_collection after you create the model but before you ever call K.get_session(), fit etc.,
- You should be sure set the name of input and output layers when you add them to the graph because you'll need to refer to them when you send prediction requests.

answered Feb 3 '17 at 22:34

rhaertel80
2,959 1 6 21

I also have a problem when making the prediction due to the keras learning phase, I train the model (learning phase = 1) then before I save I do K.set_learning_phase(0) but I still get an error: ERROR: (gcloud.beta.ml.local.predict) ERROR:root:Exception during running the graph: You must feed a value for placeholder tensor 'keras_learning_phase' with dtype bool [[Node: keras_learning_phase = Placeholder[dtype=DT_BOOL, shape=[], _device="/job:localhost/replica:0/task:0/cpu:0"]()]] – Matthew Jackson Feb 4 '17 at 10:37

You could include {"keras_learning_phase": 0, ...} as part of the inputs to each instance. Does that work? – rhaertel80 Feb 5 '17 at 10:07

Do you mean in the inputs from your code above? If so yes, it didn't work. The prediction complained the input tensor for keras_learning_phase had the wrong shape due to the first dimension ie there could be multiple inputs. Not sure how to just pass in a scalar. – Matthew Jackson Feb 5 '17 at 12:12

I added the input as `inputs['keras_learning_phase'] = K.learning_phase().name` and included `{'keras_learning_phase': 0}` in my input data for my cloud ml prediction. I get the error:

help. matthew-jackson Mar 30 '17 at 16:49