Help finding output node for creating frozen graph #63

Closed **ablacklama** opened this issue on 20 May 2018 · 10 comments

ablacklama commented on 20 May 2018

i'm trying to create a frozen graph of a model but the script requires the name of the output node. So i guess this is a request for help and a feature request. Do you know what the output node's name is? I couldn't find it in the code and i'm not familiar with tf.contrib.slim. And as a feature it would be nice to be able to export our model as a frozen inference graph:

import os, argparse

import tensorflow as tf

def freeze_graph(model_dir, output_node_names):
"""Extract the sub graph defined by the output nodes and convert
all its variables into constant
Args:
model_dir: the root folder containing the checkpoint state file
output_node_names: a string, containing all the output node's names,
comma separated
"""
if not tf.gfile.Exists(model_dir):
raise AssertionError(
"Export directory doesn't exists. Please specify an export "
"directory: %s" % model_dir)

```
if not output_node_names:
    print("You need to supply the name of a node to --output_node_names.")
    return -1

# We retrieve our checkpoint fullpath
checkpoint = tf.train.get_checkpoint_state(model_dir)
input_checkpoint = checkpoint.model_checkpoint_path

# We precise the file fullname of our freezed graph
absolute_model_dir = "/".join(input_checkpoint.split('/')[:-1])
output_graph = absolute_model_dir + "/frozen_model.pb"

# We clear devices to allow TensorFlow to control on which device it will load operations
clear_devices = True

# We start a session using a temporary fresh Graph
with tf.Session(graph=tf.Graph()) as sess:
    # We import the meta graph in the current default Graph
    saver = tf.train.import_meta_graph(input_checkpoint + '.meta',
clear_devices=clear_devices)
```

```
        # We use a built in TF helper to export variables to constants
        output_graph_def = tf.graph_util.convert_variables_to_constants(
            sess, # The session is used to retrieve the weights
            tf.get_default_graph().as_graph_def(), # The graph_def is used to retrieve the nodes
            output_node_names.split(",") # The output node names are used to select the usefull
    nodes
        )

        # Finally we serialize and dump the output graph to the filesystem
        with tf.gfile.GFile(output_graph, "wb") as f:
            f.write(output_graph_def.SerializeToString())
        print("%d ops in the final graph." % len(output_graph_def.node))

    return output_graph_def


freeze_graph('checkpoints', 'outputnode???')
'
```

3 +😀

GeorgeSeif commented on 20 May 2018

Hey there **@ablacklama**

I've never personally exported an inference graph in TF. What do you mean by output node?

+😀

ablacklama commented on 20 May 2018

Oh it's just the name of the output op that you'd run to get the models output. Since if we're using the model for inference we don't need the loss part of the graph

+😀

GeorgeSeif commented on 21 May 2018

Ah okay this is I guess a small feature missing from the repo or a mini design flaw. Basically, with these models, the last layer is supposed to be a Softmax activation. But, the way we apply it is in the `softmax_cross_entropy_with_logits()` function for training. If you wanted to export a graph for testing, you'd have to have a softmax on the end of the network, but without the whole `cross_entropy_with_logits` part.

to the end and export the graph. It may also help to give each model layer a name within the model files (i.e where the networks are defined in the `models/` folder).

2 +😀

ablacklama commented on 22 May 2018

here's an easy fix for anyone wondering in the future. in main.py line 189. This may break something if you don't want to just use the training function though...

losses = None
if args.class_balancing:
print("Computing class weights for", args.dataset, "...")
class_weights = utils.compute_class_weights(labels_dir=args.dataset + "/train_labels",
label_values=label_values)
unweighted_loss = None
if args.loss_func == "cross_entropy":
unweighted_loss = tf.nn.softmax_cross_entropy_with_logits(logits=network, labels=net_output)
network = tf.nn.softmax(network, name="softmax_output")
elif args.loss_func == "lovasz":
unweighted_loss = utils.lovasz_softmax(probas=network, labels=net_output)
losses = unweighted_loss * class_weights
else:
if args.loss_func == "cross_entropy":
losses = tf.nn.softmax_cross_entropy_with_logits(logits=network, labels=net_output)
network = tf.nn.softmax(network, name="softmax_output")

1 +😀

GeorgeSeif closed this on 11 Jun 2018

ukiras123 commented on 7 Feb 2019

@ablacklama Were you able to create the frozen graph?

+😀

ablacklama commented on 7 Feb 2019

i did, but it was a year ago so don't ask me how...

NPetsky commented on 14 Feb 2019

I froze the graph with adding a softmax layer like **@ablacklama** did, so I did not import the meta graph for freezing but reconstructed the graph with the new layer at the end, loaded the weights and then froze it

+😀

cena001plus commented on 12 Jul 2019 • edited

@GeorgeSeif hello, i have creating froen graph of pb and tflite , but the predict result is wrong ,so i test ckpt model created by train.py , the ruslut is wrong , but the ckpt model used in train.py is run good, i can't found any dierent between train.py and predict.py. the predict.py code blow .

```
import os,time,cv2, sys, math
import tensorflow as tf
import argparse
import numpy as np

from utils import utils, helpers
from builders import model_builder

parser = argparse.ArgumentParser()
parser.add_argument('--image', type=str, default='./mydata/val/18.png', required=False,
help='The image you want to predict on. ')
parser.add_argument('--checkpoint_path', type=str, default="./checkpoints/0015/model.ckpt",
required=False, help='The path to the latest checkpoint weights for your model.')
parser.add_argument('--crop_height', type=int, default=128, help='Height of cropped input
image to network')
parser.add_argument('--crop_width', type=int, default=128, help='Width of cropped input image
to network')
parser.add_argument('--model', type=str, default="DeepLabV3_plus", required=False, help='The
model you are using')
parser.add_argument('--dataset', type=str, default="mydata", required=False, help='The dataset
you are using')
args = parser.parse_args()

class_names_list, label_values = helpers.get_label_info(os.path.join(args.dataset,
"class_dict.csv"))

num_classes = len(label_values)

print("\n***** Begin prediction *****")
print("Dataset -->", args.dataset)
print("Model -->", args.model)
print("checkpoint_path -->", args.checkpoint_path)
print("Crop Height -->", args.crop_height)
print("Crop Width -->", args.crop_width)
print("Num Classes -->", num_classes)
print("Image -->", args.image)
```

```
config.gpu_options.allow_growth = True
sess=tf.Session(config=config)

net_input = tf.placeholder(tf.float32,shape=[1,128,128,3])
net_output = tf.placeholder(tf.float32,shape=[1,128,128,num_classes])

network, _ = model_builder.build_model(args.model, net_input=net_input,
                                       num_classes=num_classes,
                                       crop_width=args.crop_width,
                                       crop_height=args.crop_height,
                                       is_training=False)

sess.run(tf.global_variables_initializer())

print('Loading model checkpoint weights')
saver=tf.train.Saver(max_to_keep=1000)
saver.restore(sess, args.checkpoint_path)

print("Testing image " + args.image)

loaded_image = utils.load_image(args.image)
resized_image =cv2.resize(loaded_image, (args.crop_width, args.crop_height))
input_image = np.expand_dims(np.float32(resized_image[:args.crop_height,
:args.crop_width]),axis=0)/255.0

st = time.time()
output_image = sess.run(network,feed_dict={net_input:input_image})

run_time = time.time()-st

output_image = np.array(output_image[0,:,:,:])
output_image = helpers.reverse_one_hot(output_image)

out_vis_image = helpers.colour_code_segmentation(output_image, label_values)
file_name = utils.filepath_to_name(args.image)
cv2.imwrite("%s_predpredpred.png"%(file_name),cv2.cvtColor(np.uint8(out_vis_image),
cv2.COLOR_RGB2BGR))

print("Wrote image " + "%s_predpred.png"%(file_name))
```

+😀

cena001plus commented on 13 Jul 2019

> **@GeorgeSeif** hello, i have creating froen graph of pb and tflite , but the predict result is wrong ,so i test ckpt model created by train.py , the ruslut is wrong , but the ckpt model used in train.py is run good, i can't found any dierent between train.py and predict.py. the predict.py code blow .
>
> ```
> import os,time,cv2, sys, math
> import tensorflow as tf
> import argparse
> import numpy as np
> ```

```
parser = argparse.ArgumentParser()
parser.add_argument('--image', type=str, default='./mydata/val/18.png', required=False,
help='The image you want to predict on. ')
parser.add_argument('--checkpoint_path', type=str,
default="./checkpoints/0015/model.ckpt", required=False, help='The path to the latest
checkpoint weights for your model.')
parser.add_argument('--crop_height', type=int, default=128, help='Height of cropped input
image to network')
parser.add_argument('--crop_width', type=int, default=128, help='Width of cropped input
image to network')
parser.add_argument('--model', type=str, default="DeepLabV3_plus", required=False,
help='The model you are using')
parser.add_argument('--dataset', type=str, default="mydata", required=False, help='The
dataset you are using')
args = parser.parse_args()

class_names_list, label_values = helpers.get_label_info(os.path.join(args.dataset,
"class_dict.csv"))

num_classes = len(label_values)

print("\n***** Begin prediction *****")
print("Dataset -->", args.dataset)
print("Model -->", args.model)
print("checkpoint_path -->", args.checkpoint_path)
print("Crop Height -->", args.crop_height)
print("Crop Width -->", args.crop_width)
print("Num Classes -->", num_classes)
print("Image -->", args.image)

# Initializing network
config = tf.ConfigProto()
config.gpu_options.allow_growth = True
sess=tf.Session(config=config)

net_input = tf.placeholder(tf.float32,shape=[1,128,128,3])
net_output = tf.placeholder(tf.float32,shape=[1,128,128,num_classes])

network, _ = model_builder.build_model(args.model, net_input=net_input,
                                       num_classes=num_classes,
                                       crop_width=args.crop_width,
                                       crop_height=args.crop_height,
                                       is_training=False)

sess.run(tf.global_variables_initializer())

print('Loading model checkpoint weights')
saver=tf.train.Saver(max_to_keep=1000)
saver.restore(sess, args.checkpoint_path)

print("Testing image " + args.image)

loaded_image = utils.load_image(args.image)
resized_image =cv2.resize(loaded_image, (args.crop_width, args.crop_height))
input_image = np.expand_dims(np.float32(resized_image[:args.crop_height,
:args.crop_width]),axis=0)/255.0
```

```
    output_image = np.array(output_image[0,:,:,:])
    output_image = helpers.reverse_one_hot(output_image)

    out_vis_image = helpers.colour_code_segmentation(output_image, label_values)
    file_name = utils.filepath_to_name(args.image)
    cv2.imwrite("%s_predpredpred.png"%(file_name),cv2.cvtColor(np.uint8(out_vis_image),
    cv2.COLOR_RGB2BGR))

    print("Wrote image " + "%s_predpred.png"%(file_name))
```

is_training =false change to true, sloved.

+😀

cena001plus commented on 13 Jul 2019 • edited

@GeorgeSeif hello , the pb model is run good , but pb convert to tflite model, the tflite model 's output is dierent from pb model, i don't know where is wrong , i will preciate it if you can give sme question, the convert code is blow:

``

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-


import tensorflow as tf
from tensorflow import lite
from tensorflow import keras

#--在colab运行成功
from tensorflow import lite

converter =lite.TFLiteConverter.from_frozen_graph('Seg.pb', input_arrays=["input"],
output_arrays=["soft_output"],input_shapes={"input":[1,3,256,256]})
converter.post_training_quantize=True
tflite_model = converter.convert()
open("Seg.tflite","wb").write(tflite_model)
```

+😀

Assignees

No one assigned

Labels

Projects

None yet

Milestone

No milestone

5 participants