tflite_conversion

November 18, 2020

[5]:
```
import os
import sys
import yaml

import numpy as np
from sklearn.metrics.pairwise import cosine_similarity
import tensorflow as tf
```

[6]:
```
tf.__version__
```

[6]:
```
'2.3.1'
```

[7]:
```
ROOT = '/home/thomas/Dir/ccny/ccny-masters-thesis'
```

[8]:
```
sys.path.append(os.path.join(ROOT, 'tensorflow'))

from dataset import GE2EDatasetLoader
```

1 Load `tf.keras` Model

[9]:
```
embedding_models_path = os.path.join(ROOT, 'tensorflow/frozen_models/embedding')
model_confs_path = os.path.join(ROOT, 'tensorflow/frozen_models/confs')
```

[10]:
```
models = os.listdir(embedding_models_path)
models
```

[10]:
```
['1605081214',
 '1605371725',
 '1605623350',
 '1605672336',
 '1605647312',
 '1605518238']
```

[11]:
```
model_idx = 3

model_to_convert = models[model_idx]
```

```
model_to_convert
```

[11]:
```
'1605672336'
```

[12]:
```
confs = os.listdir(os.path.join(model_confs_path, model_to_convert))
confs
```

[12]:
```
['conv_1d.yml']
```

[13]:
```
with open(f'{model_confs_path}/{model_to_convert}/{confs[0]}', 'r') as c:
    conf = yaml.safe_load(c)

conf['train']['network']
```

[13]:
```
{'optimizer': {'type': 'SGD', 'lr': 0.01, 'clipnorm': 3.0},
 'dropout': 0.1,
 'layers': [{'conv1d': {'filters': 8, 'kernel_size': 10}},
  {'conv1d': {'filters': 8, 'kernel_size': 3}},
  'flatten',
  {'embedding': {'nodes': 64}},
  {'similarity_matrix': {'embedding_length': 64}}],
 'callbacks': {'lr_scheduler': {'cutoff_epoch': 25, 'decay': 'exponential'},
  'csv_logger': {'dir': 'training_logs'},
  'checkpoint': {'dir': 'model_checkpoints'}}}
```

[14]:
```
model_path = os.path.join(embedding_models_path, model_to_convert)
```

[15]:
```
embedding_model = tf.keras.models.load_model(model_path)
embedding_model.summary()
```

```
WARNING:tensorflow:No training configuration found in save file, so the model
was *not* compiled. Compile it manually.
```

```
[WARNING] {tensorflow} 2020-11-18 07:04:37,060 No training configuration found
in save file, so the model was *not* compiled. Compile it manually.
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 31, 8)             9688
_________________________________________________________________
batch_normalization (BatchNo (None, 31, 8)             32
_________________________________________________________________
dropout (Dropout)            (None, 31, 8)             0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 29, 8)             200
_________________________________________________________________
```

```
batch_normalization_1 (Batch (None, 29, 8)             32
_________________________________________________________________
dropout_1 (Dropout)          (None, 29, 8)             0
_________________________________________________________________
flatten (Flatten)            (None, 232)               0
_________________________________________________________________
dense (Dense)                (None, 64)                14912
=================================================================
Total params: 24,864
Trainable params: 24,832
Non-trainable params: 32
_________________________________________________________________
```

[16]:
```
train, test = GE2EDatasetLoader(root_dir=os.path.join(ROOT, 'feature-data')).
 ↪get_datasets()
```

2 Convert to tf.lite model

[42]:
```
converter = tf.lite.TFLiteConverter.from_saved_model(model_path)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
tflite_model = converter.convert()
```

[43]:
```
TFLITE_MODEL_DIR = os.path.join(ROOT, f'tensorflow/frozen_models/tiny/
 ↪{model_to_convert}')
TFLITE_MODEL_PATH = os.path.join(ROOT, TFLITE_MODEL_DIR, 'model.tflite')
!mkdir -p $TFLITE_MODEL_DIR
```

[44]:
```
with open(TFLITE_MODEL_PATH, 'wb') as out:
    out.write(tflite_model)
```

[45]:
```
!du -sh $TFLITE_MODEL_DIR/*
```

```
32K     /home/thomas/Dir/ccny/ccny-masters-
thesis/tensorflow/frozen_models/tiny/1605672336/model.tflite
```

3 Making predictions

[46]:
```
feats, targets = next(iter(train))
```

[47]:
```
feats.shape # [batch_size x spectrogram height x spectrogram width]
```

[47]:
```
TensorShape([64, 40, 121])
```

[48]:
```
targets[:8].numpy() # 8 utterances / per speaker / per batch
```

```
[48]: array([2067, 2067, 2067, 2067, 2067, 2067, 2067, 2067])
```

```
[49]: def get_output(feat, input_details, output_details):
          if feat.ndim == 2:
              feat = np.expand_dims(feat, axis = 0)
          feat = feat.astype(input_details["dtype"])
          interpreter.set_tensor(input_details["index"], feat)
          interpreter.invoke()
          output = interpreter.get_tensor(output_details["index"])[0]
          return output
```

```
[50]: interpreter = tf.lite.Interpreter(model_path=str(TFLITE_MODEL_PATH))
      interpreter.allocate_tensors()
```

```
[51]: input_details = interpreter.get_input_details()[0]
      input_details
```

```
[51]: {'name': 'conv1d_input',
       'index': 0,
       'shape': array([  1,  40, 121], dtype=int32),
       'shape_signature': array([ -1,  40, 121], dtype=int32),
       'dtype': numpy.float32,
       'quantization': (0.0, 0),
       'quantization_parameters': {'scales': array([], dtype=float32),
        'zero_points': array([], dtype=int32),
        'quantized_dimension': 0},
       'sparsity_parameters': {}}
```

```
[52]: output_details = interpreter.get_output_details()[0]
      output_details
```

```
[52]: {'name': 'Identity',
       'index': 27,
       'shape': array([ 1, 64], dtype=int32),
       'shape_signature': array([-1, 64], dtype=int32),
       'dtype': numpy.float32,
       'quantization': (0.0, 0),
       'quantization_parameters': {'scales': array([], dtype=float32),
        'zero_points': array([], dtype=int32),
        'quantized_dimension': 0},
       'sparsity_parameters': {}}
```

```
[53]: # d-vector

      # enroll with 4 utterances
      n_enrollment = 4
```

```
d_vector = np.zeros((output_details['shape'][1],))

print(f'enrolling speaker_id: {np.unique(targets[:n_enrollment])}')
for i in range(n_enrollment):
    d_vector += get_output(feats[i], input_details, output_details)

d_vector /= n_enrollment
d_vector.shape
```

```
enrolling speaker_id: [2067]
```

[53]:
```
(64,)
```

[56]:
```
# test against another example from same speaker
speaker_idx = n_enrollment + 1

pred = get_output(feats[speaker_idx], input_details, output_details)
print(f'speaker_id: {targets[speaker_idx]}')
cosine_similarity(d_vector.reshape(1,-1), pred.reshape(1,-1))
```

```
speaker_id: 2067
```

[56]:
```
array([[0.81528801]])
```

[57]:
```
# test against utterance from different speaker
speaker_idx = n_enrollment + 5

pred = get_output(feats[speaker_idx], input_details, output_details)
print(f'speaker_id: {targets[speaker_idx]}')
cosine_similarity(d_vector.reshape(1,-1), pred.reshape(1,-1))
```

```
speaker_id: 2867
```

[57]:
```
array([[0.30849305]])
```

4 Convert to C++ array

The real kicker here is being able to port our `tflite` model onto our target microcontroller (in this case an Arduino Nano 33 BLE). With the model represented on our microcontroller, we can do online speaker verification at the edge in a low power + compute setting.

[58]:
```
# https://github.com/tensorflow/tensorflow/tree/master/tensorflow/lite/micro/
↪examples/micro_speech
```

[59]:
```

```

```
TFLITE_MODEL_MICRO_PATH = os.path.join(ROOT, 'MicrocontrollerRecognizer/model.
↪h')
TFLITE_MODEL_MICRO_PATH
```

[59]:
```
'/home/thomas/Dir/ccny/ccny-masters-thesis/MicrocontrollerRecognizer/model.h'
```

[60]:
```
# https://colab.research.google.com/github/tensorflow/tensorflow/blob/master/
↪tensorflow/lite/micro/examples/hello_world/train/train_hello_world_model.
↪ipynb#scrollTo=_UQblnrLd_ET
```

[61]:
```
# Update variable names
!xxd -i {TFLITE_MODEL_PATH} > {TFLITE_MODEL_MICRO_PATH}
REPLACE_TEXT = TFLITE_MODEL_PATH.replace('/', '_').replace('.', '_').
↪replace('-', '_')
!sed -i 's/'{REPLACE_TEXT}'/speaker_model/g' {TFLITE_MODEL_MICRO_PATH}
```

[62]:
```
!head {TFLITE_MODEL_MICRO_PATH}
```

```
unsigned char speaker_model[] = {
  0x28, 0x00, 0x00, 0x00, 0x54, 0x46, 0x4c, 0x33, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x12, 0x00,
  0x1c, 0x00, 0x04, 0x00, 0x08, 0x00, 0x0c, 0x00, 0x10, 0x00, 0x14, 0x00,
  0x00, 0x00, 0x18, 0x00, 0x12, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00,
  0x3c, 0x7b, 0x00, 0x00, 0x80, 0x67, 0x00, 0x00, 0x68, 0x67, 0x00, 0x00,
  0x3c, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
  0x0c, 0x00, 0x00, 0x00, 0x08, 0x00, 0x0c, 0x00, 0x04, 0x00, 0x08, 0x00,
  0x08, 0x00, 0x00, 0x00, 0x08, 0x00, 0x00, 0x00, 0x1d, 0x00, 0x00, 0x00,
  0x13, 0x00, 0x00, 0x00, 0x6d, 0x69, 0x6e, 0x5f, 0x72, 0x75, 0x6e, 0x74,
```

[ ]:
```

```