FCN5

June 24, 2020

0.0.1 -- *coding: utf-8* --

MusicTaggerCNN model for Keras.

0.0.2 Reference:

- Automatic tagging using deep convolutional neural networks
- Music-auto_tagging-keras

[1]:
```
from __future__ import print_function
from __future__ import absolute_import

import keras
from keras import backend as K
from keras.layers import Input, Dense
from keras.models import Model
from keras.layers import Dense, Dropout, Flatten
from keras.layers.convolutional import Convolution2D
from keras.layers.convolutional import MaxPooling2D, ZeroPadding2D
from keras.layers.normalization import BatchNormalization
from keras.layers.advanced_activations import ELU
from keras.utils.data_utils import get_file
from keras.layers import Input, Dense
```

```
Using Theano backend.
WARNING (theano.tensor.blas): Using NumPy C-API based implementation for BLAS
functions.
```

[2]:
```
TH_WEIGHTS_PATH = 'https://github.com/keunwoochoi/music-auto_tagging-keras/blob/
 ↪master/data/music_tagger_cnn_weights_theano.h5'
TF_WEIGHTS_PATH = 'https://github.com/keunwoochoi/music-auto_tagging-keras/blob/
 ↪master/data/music_tagger_cnn_weights_tensorflow.h5'
```

1 MusicTaggerCNN

Instantiate the MusicTaggerCNN architecture, optionally loading weights pre-trained on Million Song Dataset. Note that when using TensorFlow, for best performance you should set `image_dim_ordering="tf"` in your Keras config at ~/.keras/keras.json.

The model and the weights are compatible with both TensorFlow and Theano. The dimension ordering convention used by the model is the one specified in your Keras config file.

For preparing mel-spectrogram input, see audio_conv_utils.py

in applications. You will need to install Librosa to use it.

Arguments

```
weights: one of `None` (random initialization) or "msd" (pre-training on ImageNet).

input_tensor: optional Keras tensor (i.e. output of `layers.Input()`) to use as image input fo

include_top: whether to include the 1 fully-connected layer (output layer) at the top of the n
```

Returns

A Keras model instance.

[3]:
```
def MusicTaggerCNN(weights='msd', input_tensor=None,
                   include_top=True):
    '''Instantiate the MusicTaggerCNN architecture,
    optionally loading weights pre-trained
    on Million Song Dataset. Note that when using TensorFlow,
    for best performance you should set
    `image_dim_ordering="tf"` in your Keras config
    at ~/.keras/keras.json.

    The model and the weights are compatible with both
    TensorFlow and Theano. The dimension ordering
    convention used by the model is the one
    specified in your Keras config file.

    For preparing mel-spectrogram input, see
    `audio_conv_utils.py` in [applications](https://github.com/fchollet/keras/
↪tree/master/keras/applications).
    You will need to install [Librosa](http://librosa.github.io/librosa/)
    to use it.

    # Arguments
        weights: one of `None` (random initialization)
            or "msd" (pre-training on ImageNet).
        input_tensor: optional Keras tensor (i.e. output of `layers.Input()`)
            to use as image input for the model.
        include_top: whether to include the 1 fully-connected
            layer (output layer) at the top of the network.
            If False, the network outputs 256-dim features.
```

```
    # Returns
        A Keras model instance.
    '''
    if weights not in {'msd', None}:
        raise ValueError('The `weights` argument should be either '
                         '`None` (random initialization) or `msd` '
                         '(pre-training on Million Song Dataset).')

    # Determine proper input shape
    if keras.backend.image_data_format() == 'channels_first':
        input_shape = (1, 96, 1366)
    else:
        input_shape = (96, 1366, 1)

    if input_tensor is None:
        melgram_input = Input(shape=input_shape)
    else:
        if not K.is_keras_tensor(input_tensor):
            melgram_input = Input(tensor=input_tensor, shape=input_shape)
        else:
            melgram_input = input_tensor

    # Determine input axis
    if keras.backend.image_data_format() == 'channels_first':
        channel_axis = 1
        freq_axis = 2
        time_axis = 3
    else:
        channel_axis = 3
        freq_axis = 1
        time_axis = 2

    # Input block
    x = BatchNormalization(axis=freq_axis, name='bn_0_freq')(melgram_input)

    # Conv block 1
    x = Convolution2D(64, 3, 3, border_mode='same', name='conv1')(x)
    x = BatchNormalization(axis=channel_axis, mode=0, name='bn1')(x)
    x = ELU()(x)
    x = MaxPooling2D(pool_size=(2, 4), name='pool1')(x)

    # Conv block 2
    x = Convolution2D(128, 3, 3, border_mode='same', name='conv2')(x)
    x = BatchNormalization(axis=channel_axis, mode=0, name='bn2')(x)
    x = ELU()(x)
    x = MaxPooling2D(pool_size=(2, 4), name='pool2')(x)
```

```
    # Conv block 3
    x = Convolution2D(128, 3, 3, border_mode='same', name='conv3')(x)
    x = BatchNormalization(axis=channel_axis, mode=0, name='bn3')(x)
    x = ELU()(x)
    x = MaxPooling2D(pool_size=(2, 4), name='pool3')(x)

    # Conv block 4
    x = Convolution2D(128, 3, 3, border_mode='same', name='conv4')(x)
    x = BatchNormalization(axis=channel_axis, mode=0, name='bn4')(x)
    x = ELU()(x)
    x = MaxPooling2D(pool_size=(3, 5), name='pool4')(x)

    # Conv block 5
    x = Convolution2D(64, 3, 3, border_mode='same', name='conv5')(x)
    x = BatchNormalization(axis=channel_axis, mode=0, name='bn5')(x)
    x = ELU()(x)
    x = MaxPooling2D(pool_size=(4, 4), name='pool5')(x)

    # Output
    x = Flatten()(x)
    if include_top:
        x = Dense(10, activation='sigmoid', name='output')(x)

    # Create model
    model = Model(melgram_input, x)
    if weights is None:
        return model
    else:
        # Load input
        if keras.backend.image_data_format() == 'channels_last':
            raise RuntimeError("Please set keras.backend.image_data_format() ==␣
↪'channels_first'."
                               "You can set it at ~/.keras/keras.json")
        model.load_weights('data/music_tagger_cnn_weights_%s.h5' % K._BACKEND,
                           by_name=True)
        return model
```

[4]:
```
from IPython.display import Image
Image(filename='tf_th_keras_v2.png')
```

[4]:

Keras is ignoring the image_dim_ordering setting, because with **Keras v2** (v2.0.4 being the latest at the time of this writing) the name of the parameter has been changed.

Now you set it using the **image_data_format** parameter.

image_data_format can be set to **"channels_last"** or **"channels_first"**, which corresponds to the TensorFlow or Theano dimension orders respectively.

i.e. When using TensorFlow, you now set your **keras.json** like this,

```
{
  "image_data_format": "channels_last",
  "epsilon": 1e-07,
  "floatx": "float32",
  "backend": "tensorflow"
}
```

For Theano, you need to set it like this,

```
{
  "image_data_format": "channels_first",
  "epsilon": 1e-07,
  "floatx": "float32",
  "backend": "theano"
}
```

[5]:
```
import keras

if keras.backend.image_data_format() == 'channels_last':
    print("here backend is tensorflow" , keras.backend.image_data_format())
elif keras.backend.image_data_format() == 'channels_first':
    print("here backend is theano" , keras.backend.image_data_format())

#print(K.image_dim_ordering())
```

```
here backend is theano channels_first
```

TRY TO MAKE A MODEL USER-FRIENDLY IN NEAR FUTURE.

[6]:
```
model = MusicTaggerCNN(weights=None)
```

```
/home/user/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:66:
UserWarning: Update your `Conv2D` call to the Keras 2 API: `Conv2D(64, (3, 3),
name="conv1", padding="same")`
/home/user/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:67:
UserWarning: Update your `BatchNormalization` call to the Keras 2 API:
`BatchNormalization(axis=1, name="bn1")`
/home/user/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:72:
UserWarning: Update your `Conv2D` call to the Keras 2 API: `Conv2D(128, (3, 3),
```

```
name="conv2", padding="same")`
/home/user/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:73:
UserWarning: Update your `BatchNormalization` call to the Keras 2 API:
`BatchNormalization(axis=1, name="bn2")`
/home/user/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:78:
UserWarning: Update your `Conv2D` call to the Keras 2 API: `Conv2D(128, (3, 3),
name="conv3", padding="same")`
/home/user/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:79:
UserWarning: Update your `BatchNormalization` call to the Keras 2 API:
`BatchNormalization(axis=1, name="bn3")`
/home/user/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:84:
UserWarning: Update your `Conv2D` call to the Keras 2 API: `Conv2D(128, (3, 3),
name="conv4", padding="same")`
/home/user/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:85:
UserWarning: Update your `BatchNormalization` call to the Keras 2 API:
`BatchNormalization(axis=1, name="bn4")`
/home/user/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:90:
UserWarning: Update your `Conv2D` call to the Keras 2 API: `Conv2D(64, (3, 3),
name="conv5", padding="same")`
/home/user/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:91:
UserWarning: Update your `BatchNormalization` call to the Keras 2 API:
`BatchNormalization(axis=1, name="bn5")`
```

[7]:
```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 1, 96, 1366)       0
_________________________________________________________________
bn_0_freq (BatchNormalizatio (None, 1, 96, 1366)       384
_________________________________________________________________
conv1 (Conv2D)               (None, 64, 96, 1366)      640
_________________________________________________________________
bn1 (BatchNormalization)     (None, 64, 96, 1366)      256
_________________________________________________________________
elu_1 (ELU)                  (None, 64, 96, 1366)      0
_________________________________________________________________
pool1 (MaxPooling2D)         (None, 64, 48, 341)       0
_________________________________________________________________
conv2 (Conv2D)               (None, 128, 48, 341)      73856
_________________________________________________________________
bn2 (BatchNormalization)     (None, 128, 48, 341)      512
_________________________________________________________________
elu_2 (ELU)                  (None, 128, 48, 341)      0
_________________________________________________________________
pool2 (MaxPooling2D)         (None, 128, 24, 85)       0
_________________________________________________________________
```

```
conv3 (Conv2D)               (None, 128, 24, 85)       147584
_________________________________________________________________
bn3 (BatchNormalization)     (None, 128, 24, 85)       512
_________________________________________________________________
elu_3 (ELU)                  (None, 128, 24, 85)       0
_________________________________________________________________
pool3 (MaxPooling2D)         (None, 128, 12, 21)       0
_________________________________________________________________
conv4 (Conv2D)               (None, 128, 12, 21)       147584
_________________________________________________________________
bn4 (BatchNormalization)     (None, 128, 12, 21)       512
_________________________________________________________________
elu_4 (ELU)                  (None, 128, 12, 21)       0
_________________________________________________________________
pool4 (MaxPooling2D)         (None, 128, 4, 4)         0
_________________________________________________________________
conv5 (Conv2D)               (None, 64, 4, 4)          73792
_________________________________________________________________
bn5 (BatchNormalization)     (None, 64, 4, 4)          256
_________________________________________________________________
elu_5 (ELU)                  (None, 64, 4, 4)          0
_________________________________________________________________
pool5 (MaxPooling2D)         (None, 64, 1, 1)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 64)                0
_________________________________________________________________
output (Dense)               (None, 10)                650
=================================================================
Total params: 446,538
Trainable params: 445,322
Non-trainable params: 1,216
_________________________________________________________________
```

1.0.1 Compiling the model

[2]:
```
#compile model using accuracy to measure model performance
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

```
␣
↪---------------------------------------------------------------------------

NameError                                 Traceback (most recent call␣
↪last)

<ipython-input-2-5b0921aaafdd> in <module>
```

```
      1 #compile model using accuracy to measure model performance
----> 2 model.compile(optimizer='adam',
      3               loss='categorical_crossentropy',
      4               metrics=['accuracy'])


NameError: name 'model' is not defined
```

1.0.2 Load data

[9]:
```
import numpy as np
```

[10]:
```
concat_x = np.load('concat_x.npy')
concat_y = np.load('concat_y.npy')
```

[11]:
```
print(len(concat_x),'   ',(len(concat_y)) )
```

```
1000     1000
```

[12]:
```
train_x = concat_x[0:800]
train_y = concat_y[0:800]
```

[13]:
```
test_x = concat_x[800:1000]
test_y = concat_y[800:1000]
```

```
valid_x = concat_x[950:1000] valid_y = concat_y[950:1000]
```

1.0.3 Training the model

```
#train the model model.fit(X_train, y_train, validation_data=(X_test, y_test),
epochs=1000)
```

[14]:
```
model.fit(train_x, train_y, validation_data=(test_x, test_y), epochs=12 )
```

```
Train on 800 samples, validate on 200 samples
Epoch 1/12
800/800 [==============================] - 1147s 1s/step - loss: 1.9874 - acc:
0.1725 - val_loss: 4.6778 - val_acc: 0.0850
Epoch 2/12
800/800 [==============================] - 1173s 1s/step - loss: 1.6163 - acc:
0.4138 - val_loss: 3.3504 - val_acc: 0.1100
Epoch 3/12
800/800 [==============================] - 1193s 1s/step - loss: 1.3535 - acc:
0.5587 - val_loss: 2.7192 - val_acc: 0.1100
Epoch 4/12
800/800 [==============================] - 1328s 2s/step - loss: 1.1566 - acc:
0.6375 - val_loss: 2.4091 - val_acc: 0.1750
Epoch 5/12
```

```
800/800 [==============================] - 1142s 1s/step - loss: 1.0232 - acc:
0.6763 - val_loss: 2.3538 - val_acc: 0.2950
Epoch 6/12
800/800 [==============================] - 1179s 1s/step - loss: 0.8556 - acc:
0.7375 - val_loss: 2.4140 - val_acc: 0.2550
Epoch 7/12
800/800 [==============================] - 1111s 1s/step - loss: 0.7374 - acc:
0.7725 - val_loss: 2.3458 - val_acc: 0.2600
Epoch 8/12
800/800 [==============================] - 1074s 1s/step - loss: 0.6354 - acc:
0.8100 - val_loss: 2.1673 - val_acc: 0.2900
Epoch 9/12
800/800 [==============================] - 1137s 1s/step - loss: 0.5068 - acc:
0.8475 - val_loss: 1.9019 - val_acc: 0.4050
Epoch 10/12
800/800 [==============================] - 1162s 1s/step - loss: 0.4698 - acc:
0.8762 - val_loss: 1.9972 - val_acc: 0.3900
Epoch 11/12
800/800 [==============================] - 1143s 1s/step - loss: 0.3633 - acc:
0.9113 - val_loss: 1.7700 - val_acc: 0.4350
Epoch 12/12
800/800 [==============================] - 1119s 1s/step - loss: 0.3009 - acc:
0.9413 - val_loss: 2.1803 - val_acc: 0.3350
```

[14]:
```
<keras.callbacks.History at 0x7ff343327c90>
```

[15]:
```
from keras.models import load_model

model.save('fcn.h5')  # creates a HDF5 file 'my_model.h5'
del model  # deletes the existing model

# returns a compiled model
# identical to the previous one
model = load_model('fcn.h5')
```

[ ]:
```

```