Urban Sound Classification with CNN on Raspberry Pi Model 3

Author: David Salvo Gutiérrez

Work Group: GTAC-iTEAM (Universitat Politècnica de València)

Feature extraction from sound

Introduction

We all got exposed to different sounds every day. Like, the sound of traffic jam, siren, music and dog bark etc. We understand sound in terms of recognition and association to a known sound, but what happens when you don't know that sound and not recognize his source?

Well that's the same starting point for a computer classifier. In that sense, how about teaching computer to classify such sounds automatically into different categories.

In this notebook we will learn techniques to classify urban sound using machine learnig. Classifying sound is pretty different from other source of data. In this notebook we will first see what features can be extracted from sound data and how easy it is to extract such features in Python using open source library called *Librosa*.

To follow this tutorial, make sure you have installed the following tools: * Tensorflow * Librosa * Numpy * Matplotlib * glob * os * keras * pandas * scikit-learn * datetime

Dataset

In this experience we are focused on delvelop a convolutional neural network model able to classify automatically the different urban sounds. To train and work afford this project, I use the urban sound dataset UbanSound8K published by the SONY project reserchers. It contains 8.732 labelled sound clips (<= 4s) from ten different classes according their Urban Soun Taxonomy publication:

- Aire Conditioner (classID = 0)
- Car Horn (classID = 1)
- Children Playing (classID = 2)
- Dog Bark (classID = 3)
- Drilling (classID = 4)
- Engine Idling (classID = 5)
- Gun Shot (classID = 6)
- Jackhammer (classID = 7)
- Siren (classID = 8)
- Street Music (classID = 9)

This dataset is available for free in this link,*UrbanSound8K*.

Whe you download the dataset, you will get a '.tar.gz' compressed file (UNIX compression distribution), from

Windows you can use prgrams like 7-zip to uncompress the file.

That file contains two different directories, one of them you can find information about audio fragments classification from a metadata 'UrbanSound8K.csv' file. The other directory contains the audio segments divided in 10 different blocks not classified by classes. Finally, audio data is distibuted as:

- slice_file_name: The name of the audio file. The name takes the following format: fsID-classID-occurrenceID-sliceID.wav
- fsID: the Freesound ID of the recording from which this excerpt (slice) is taken
- start: The start time of the slice in the original Freesound recording
- end: The end time of slice in the original Freesound recording
- salience: A (subjective) salience rating of the sound. 1 = foreground, 2 = background.
- fold: The fold number (1-10) to which this file has been allocated.
- classID: A numeric identifier of the sound class
- class: The class name

source: J. Salamon, C. Jacoby and J. P. Bello, "A Dataset and Taxonomy for Urban Sound Research", 22nd ACM International Conference on Multimedia, Orlando USA, Nov. 2014.

Project Dataset

In this rep it is not uploaded all audio dataset and features extracted to test and play the Classifier, for that should download full audio directori in this link:

https://mega.nz/#F!CZJDEAyA!kZT8d6Xl7A6sjGhU6xevSA

Audio folder contains the folders before:

```
/audio
    /fold* (1-10)
    /metadata
    /test
    /test_valencia_sound_dataset
    /train_dataset
```

- fold: contains audio files from UrbanSound8K dataset.
- metadata: contains excel file in wich one it is describe all audio dataset provide by UrbanSound8K
- test: contains numpy array extracted from individual folds, from*fold* folders, using the script *sound_featuring.py*
- test_valencia_sound_dataset: contains numpy array extracted from valencia recorded audio files, from *UrbanSoun_Valencia_dataset*
- train_dataset: contains numpy array extracted from set of*folds*. *features_no1.npy* represents all feature and label extracted from all folds exept *fold 1* using the script *sound_featuring.py*

How to use this rep?

In this rep it is upload all scripts used to implement a cnn classificator for urban sound into a raspberry pi

model 3. To implement a functional demo using this rep, you should follow the following indications.

You should take into account that the model it should be trained in the PC saved as a .h5 file and then load into the raspberry pi. To train the model first of all you must create a numpy array with the features and labels from the dataset you are going to use, you should use the script *sound_featuring.py* indicating inside it wich are .wav files to be analize.

Whe you get the features and label using *sound_featuring.py* you can train the model using *cnn.py* and indicating wich are the numpy array to use to train the model.

Finally you could test the project on the raspberry pi, using the scripts,*sound_featuring.py* to get the features from the audio segment to classify, and use *sound_classifier.py* to classify using the features extracted, between the 10 classes defined.

Project Directory

For a correct use of the repository, the directory structure must follow the following example, taking into account that the feature extraction function get labels from the .wav filename, using the number of '-' presented to get the correct label value.

/user/ml-exercise/ml-soundFeat/soundFeat/(all project files and directories)

Once you get audio directory, you should get a directory named 'soundFeat' with the following files and directories:

```
/ml-exercise
    /ml-soundFeat
        /soundFeat
            /audio
                /fold* (1-10)
                /metadata
                /test
                /test_valencia_sound_dataset
                /train_dataset
            /models
                models*.h5
            class_validation.py
            cnn.py
            functions.py
            sound_classifier.py
            sound_featuring.py
```

All project directory explanation

Audio folder specifications

In this directory there's all audio content and features extracted used to make all train and test task to develop

our urban sound classifier. The folders 'fold*' and 'metadata' are obtain from UrbanSound8K dataset, as I mentioned before. Folders 'test' 'train_dataset' and 'test_valencia_sound_dataset' are created by me specifically for this project.

'test' folder contains each feature and label extraction (tuples of feature and label) for individual 'fold' directory. Per example, file named 'feature_test1.npy' and 'labels_test1.npy' correspond to the feature and label extraction from the 'fold1' dataset using the script 'sound_featuring'.

'train_dataset' folder contains each feature and labe extraction for all group of nine folds from the UrbanSound8K dataset. Per example, file named 'features_no1.npy' and 'labels_no1.npy' correspond to the feature extraction from the all .wav audio file that belongs to the group of folders: fold2 up to fold10, i.e. all folds except the one.

'UrbanSound_Valencia_dataset' it contains urban audio files with the same characteristics like the UrbanSound8K dataset, recorded in the city of Valencia, Spain.

'test_valencia_dataset' it contains feature and label extraction from the urban sound dataset created for this project, recording urban sound of the Valencia city, in Spain.

Script files specifications

In this project you would find differents python scripts in orther to deploy a functional urban sound classifier over a Raspberry Pi Model 3. I would explain the content and utility of each python scrip used in this folder.

functions: in this file you would fine the functions used to extract features and label from an audio file (.wav extension).

sound_featuring: in this script you would fine the implementation of the extraction of the features and label from an specific set of audio files.

sound_classifier: in this script you would fine the script to implement into a Raspberry Pi to classifie new audio inputs, using a trained model, that you would fine in the script cnn in wich you could train new models.

cnn: in this script you would fine the implementation of an cnn model and the resources needed for train an test the model.

class_validation: in this script you would fine a function to test the model trained and imported with new input data getting the classification % for each class.

Stand up the project

To extract the Fetures and Label (sound_featuring)

To get the feature extraction for supervise learning or a unknown audio file, you should use the script **sound_featuring** in wich you would define the directory where it is your set or just one .wav file to get their features and use them to classifie the source.

If you see the content of the script, you would need to modify the name of the 'parent_dir' and 'sub_dir', in wich

it is save your .wav file to classifie.

```
parent_dir = '/audio'
sub_dirs= ['fold8']
```

To safe those feature extraction, just need to implement 'np.save' method:

```
# Saving Features and Labels arrays
np.save('features_test8', features)
np.save('labels_test8', labels)
```

IMPORTANT: as it is define on the function *feature_extraction* labels are being extracted from the filename taking into account the number of '-' until the label. In the function it is define as it would not possible to have more than one '-' in the completed path, because the number identifies the third, if It is [3], value after the third '-'. Such this:

```
for l, sub_dir in enumerate(sub_dirs):
        for fn in glob.glob(os.path.join(os.path.abspath(parent_dir),sub_dir,file_ext)):
            sound_clip,s = librosa.load(fn)
            label = fn.split('-')[3] (this number indentifies)
```

Path should be like this: C:\Users\user_name\ml-exercises\ml-soundFeat\soundFeat\audio\fold6

Taking into account that it is defined like we would get the third value of the .wav directory path after the third '-' character in the path, like the next example:

```
/ml-exercise
    /ml-soundFeat
        /audio
            /fold1
                /7061-6-0-0.wav (labe 6: class gun shot)
```

If in the function it is define with the number [3], we would get the label **'6'** following the example before. In other hands, if you change the value to the number [1] the folder path for audio files should be like this (without any '-' character):

C:\Users\user_name\mlexercises\mlsoundFeat\soundFeat\audio\fold6

To train the model (cnn)

If you want to train the model with other dataset or make any test you should use the script *cnn.py* in wich you woudl be able to modify and train the model with the dataset you import as a feature/label set of data.

```
 # LOAD FEATURES AND LABELS
features = np.load('audio/train_dataset/features_no6.npy')
labels = np.load('audio/train_dataset/labels_no6.npy')

# SPLIT DATASET: set into training and test
validation_size = 0.20
seed = 42
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(features, labels, te

# TRAINING
model.fit(X_train, Y_train, epochs=25, validation_data=(X_test, Y_test))

# EVALUATION
test_loss, test_acc = model.evaluate(X_test,  Y_test, verbose=2)

train_score = model.evaluate(X_train, Y_train, verbose=0)
test_score = model.evaluate(X_test, Y_test, verbose=0)

print("Training Accuracy: ", train_score[1])
print("Testing Accuracy: ", test_score[1])
```

To classify sound dataset (sound_classifier or class_validation)

CASE 1: If you want to test the model with one of the folds you dind't use on training, you just need to pick up the model trained, import it and deploy it. Using the script *sound_classifier* you just need to introduce the model.h5 saved directory and import the features and labels from the dataset provided in the folds, taking those one you dind't use in the training step. Per example, if you take the model *no10_model.h5* you would need to import *features_test10.npy* and *labels_test10.npy*.

```
 # LOAD FEATURES AND LABELS
features_test = np.load('features_test1.npy')
labels_test = np.load('labels_test1.npy')

# LOAD PRE-TRAINED MODEL
model = tf.keras.models.load_model('models/no1_model.h5')
```

CASE 2: If you want to classify a new input audio, that is not classify, i.e. It has not asigned a label, you would need to use the script *class_validation.npy* in wich you would get the classification decision importing the model trained and indicating the directory path where it is saved the audio files (.wav) to be classify.