Urban Sound Classification with CNN on Raspberry Pi Model 3 and Windows 10

Author: David Salvo Gutiérrez

Co-Author: Daniel Sanz Montrull

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

Environment Project Set Up

Windows 10 installations

We will train and test our model on the PC first. We used windows 10 for Dell XPS 15 Lapto, which we used to train and test de classification model. To set-up the windows 10 lapto we must install Python 3.6 version environment and its libraries, for this project. To make things simple, we used Anaconda as a Python Framework to work with this environment.

1. Install Anaconda 3.8 (https://www.anaconda.com/products/individual)
2. Then as it is indicated on the official Tensorflow page, we must install Microsoft Visual C++ on windows. To take into account different SO distributions for the installation, follow the instructions indicated in the page: https://www.tensorflow.org/install/pip#windows
3. Once it was installed Visual C++, we have to create a virtual environment as it is indicated on the following step.
4. Create a virtual environment into Anaconda Prompt for Tensorflow.
5. (open anaconda prompt, user example)
    1. C:\Users\sglvladi> conda create -n tensorflow pip python=3.7 # select python version, tensorflow is the virtual env name.
    2. C:\Users\sglvladi> conda activate tensorflow
    3. (tensorflow) C:\Users\sglvladi>
6. Install all dependecies needed for our project into the virtual environment.
    1. (tensorflow) C:\Users\sglvladi> pip install --ignore-installed --upgrade tensorflow
    2. (tensorflow) C:\Users\sglvladi> pip install pandas
    3. (tensorflow) C:\Users\sglvladi> pip install keras
    4. (tensorflow) C:\Users\sglvladi> pip install jupyterlab
    5. (tensorflow) C:\Users\sglvladi> pip install numpy
    6. (tensorflow) C:\Users\sglvladi> pip install librosa
    7. (tensorflow) C:\Users\sglvladi> deactivate tensorflow #(virtual env exit)

Windows 10 Set-Up

Once we install all software environment needed for this project, just choose your operation IDE. We could use Jupyter Lab to interactively program our Classification Model our we can use Common User IDE as Visual Studio Code. In our case we use both of them, at the beginning it would be very useful to use Jupyter Lab in order to understand correctly the code below, and the we use VSC to fastly organize and program all project scripts needed.

If we want to use Jupyter Lab, just activate previous virtual env.

1. (open anaconda prompt): C:\Users\sglvladi> activate tensorflow
    1. (tensorflow) C:\Users\sglvladi> jupyterlab

The Jupyter Lab interface would be opened. Otherwise, just open VSC and select python interpreter (tensorflow) for VSC Open Terminal. Then you can compile and execute python scripts from the same VSC. To initialize the project, the docker image is displayed in which we have all the network services that have been implemented in this network. To do this we will follow the steps indicated below.

2. First download and install docker on the PC. Once downloaded, the service is initialized and from the terminal without having to activate the anaconda environment, the services of the image found in the \docker folder are displayed.
    1. C:\Users\sglvladi\docker> docker-compose -p FIWARE up -d

Una vez inicializados los servicios se comprueba que están en funcionamiento y escuchando en los puertos de red que se habian indicado en la image.

3. C:\Users\sglvladi\docker> docker ps -a (It is not necessary to indicate with -f the name of the .yaml because by default it implements the one called docker-compose)

If the image has been correctly instantiated and the services deployed, we should obtain the following result after executing the previous command:

First Header	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
78cf2e4812b4	smartsdk/quantumleap	"/bin/sh -c 'python …"	12 days ago	Up 34 seconds (healthy)	0.0.0.0:8668->8668/tcp	quantumleap
28c98df320c5	fiware/orion:2.4.0	"/usr/bin/contextBro…"	12 days ago	Up 34 seconds (healthy)	0.0.0.0:1026->1026/tcp	orion
022b1dd94129	grafana/grafana:7.0.4	"/run.sh"	12 days ago	Up 34 seconds	0.0.0.0:3003->3000/tcp	grafana
7197a1e0e0cd	mongo:3.6	"docker-entrypoint.s…"	12 days ago	Up 35 seconds	27017/tcp	orion_mongo

First Header	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
ec31b31f58aa	crate:4.1	"/docker-entrypoint...."	12 days ago	Up 35 seconds	0.0.0.0:4200->4200/tcp, 0.0.0.0:4300->4300/tcp, 0.0.0.0:5432->5432/tcp	cratedb

Once the network services have been initialized, the correct operation of the classifier can be checked by executing the main system script "sound_class_raspbery.py". From the terminal, and within the anaconda environment that we had executed, we go to the project directory where the script is located and we execute it. Automatically, from this script it will classify the first .waw audio file that has been housed in the / audio / input folder, the result will be classified on the terminal and it will automatically send the changes made to the specified broker entity to the server.

4. (open anaconda prompt): C:\Users\sglvladi> activate tensorflow
    1. (tensorflow) C:\Users\sglvladi> python sound_class_raspberry.py (we execute the main scipt to get the audio in input folder classfied)

Class	Classification Index
Air Conditioner	0.000395077338907867670059204101 56
Car Horn	0.000096438983746338635683059692 38
Children Playing	0.040546689182519912719726562500 00
Dog Bark	0.929443240165710449218750000000 00
Drilling	0.004420133307576179504394531250 00
Engine Idling	0.001285246224142611026763916015 62
Gun Shot	0.000011604003702814225107431411 74
Jackhammer	0.006147525273263454437255859375 00
Siren	0.000277619430562481284141540527 34
Street Music	0.017376471310853958129882812500 00
The predicted class is (V2):	Dog Bark

Date	Timestamp
2020-09-03	11:39:22.448415

In this way, if it has been executed correctly, the following output should be obtained from the terminal. This information is the same that is sent to docker to update the entity corresponding to the device on which the script has been executed. This is intended because this script, even if the test is carried out on the server where the network has been deployed with docker (localhost), it is assumed that it will be executed on those remote sensors that must send and connect to the broker through the server's ip where the docker image is deployed. The end label in the capture after execution that appears after the result of the classification is the timestamp that indicates that the broker data and the entity to which it belongs has been sent correctly.

5. Dashboard and Database Visualization

Taking into account from where you want to access to the server servicies, yo would use the IP-Server address or localhost in the case you are using the server monitor. Taking this into account you must use the ip-server address or localhost, where you deploy the docker services and the look up on the browser for the grafana service.

To visualize the dashboard you must look up for thi direction on the browser: https://ip-server:3003 (then it would show the dashboard)

To access the CrateDB database: https://ip-server:4200

Raspberry Pi installations (Tested on RPI 3 Modelo B and RPI 4)

Once we get all scripts needed to get our classification model, and it is running correctly, then we implement these software onto the Raspberry Pi Model 3 B. The installation of the environment that has been used for this project presents its complexity in the compatibility of versions of tensorflow and keras in Raspberry Pi as well as the correct installation of the Librosa library in this device, since it presents common problems with dependencies such as llvmlite and numba . To correctly install these dependencies, the Berryconda tool has been used, which pre-compiles these libraries, simplifying their installation on the Raspberry Pi without the need to resort to a source installation. The procedure followed is as follows:

Info1: http://llvmlite.pydata.org/en/latest/admin-guide/install.html Info2: https://github.com/numba/numba/issues/3670

1. Set up Python3.7 environment
    1. sudo su - (admin access)
    2. sudo apt install libblas-dev llvm python3-pip python3-scipy
    3. sudo apt update
    4. sudo apt install python3-dev python3-pip
    5. sudo apt install libatlas-base-dev

6. sudo pip3 install -U virtualenv
7. sudo apt-get install build-essential tk-dev libncurses5-dev libncursesw5-dev libreadline6-dev libdb5.3-dev libgdbm-dev libsqlite3-dev libssl-dev libbz2-dev libexpat1-dev liblzma-dev zlib1g-dev libffi-dev liblapack-dev libblas-dev wget git cmake
8. cd /root/
9. wget https://www.python.org/ftp/python/3.7.2/Python-3.7.2.tar.xz
10. tar xf Python-3.7.2.tar.xz
11. cd /root/Python-3.7.2
12. ./configure
13. make -j 4
14. sudo make altinstall
15. pip3.7 install --upgrade pip
16. sudo pip3.7 install wheel
17. sudo pip3.7 install scipy
18. sudo pip3.7 install pandas
19. sudo pip3.7 install plotly

2. Install from source LLVM 10.0 (neede for llfmlite and numba librosa's dependencies)

    1. cd /root/
    2. git clone https://github.com/numba/numba.git
    3. git clone https://github.com/numba/llvmlite.git
    4. wget https://github.com/llvm/llvm-project/releases/download/llvmorg-10.0.1/llvm-10.0.1.src.tar.xz
    5. tar -xf llvm-10.0.1.src.tar.xz
    6. cd /root/llvm-10.0.1.src.tar.xz
    7. patch -p1 -i /root/llvmlite/conda-recipes/llvm-lto-static.patch
    8. patch -p1 -i /root/llvmlite/conda-recipes/partial-testing.patch
    9. patch -p1 -i /root/llvmlite/conda-recipes/intel-D47188-svml-VF.patch
    10. patch -p1 -i /root/llvmlite/conda-recipes/0001-Revert-Limit-size-of-non-GlobalValue-name.patch/root/llvmli
    11. export PREFIX=/usr/local CPU_COUNT=4
    12. chmod +x /root/llvmlite/conda-recipes/llvmdev/build.sh
    13. /root/llvmlite/conda-recipes/llvmdev/build.sh (It took about 12 hours to install it, aproximately or more it depends on the rpi)

3. Config LLVM for our environment

    1. cd /root/llvmlite
    2. export LLVM_CONFIG=/bin/llvm-config (if not, try with: export LLVM_CONFIG=/usr/local/bin/llvm-config)
    3. python3.7 setup.py build
    4. python3.7 runtests.py
    5. python3.7 setup.py install
    6. cd /root/numba
    7. python3.7 setup.py install

To test if eveything is install correctly on the same environment just check pip3 packages list. Sometime we could get some output errors, but are not a problem because we installed from source LLVM and whe we used pip to install dependencies it would find htat are already correctly installed.

4. pip3 list (check if llvmlite and numba are correctly installed)
5. Reboot system
6. Install Tensorflow, Librosa and Keras

    1. sudo su -
    2. pip3.7 install librosa (an error will appear, during the installation but it is checked if it has been installed with >> pip3 list)
    3. pip3.7 install tensorflow

    (testing Tensorflow installation)

    4. python3.7 -c 'import tensorflow as tf; print(tf.**version**)'
    5. pip3.7 install keras
    6. sudo apt-get install python3-h5py
    7. pip3.7 install pyaudio
    8. pip3.7 install requests

Finally, we will not have any problem in being able to correctly execute the scripts in this directory. The objective in this installation is to be able to install the book libraries a, tensorflow and keras in the same python environment. For this, in this case, we have chosen to install the LLVM version 10 dependency from the source in the python 3.7 environment since this will be compatible with the necessary book and tensorflow.

To launch the script you would need to use the sequences:

1. sudo su -
2. cd /home/pi/ml-exercise/SSEnCE-merged
3. python3.7 sound_class_raspberry.py (important to take into account python3.7 that we used as our python env)

To launch the sound classificator with the MIC, for streaming classification. You must deploy main.py from raspberryPi folder.

1. cd /home/pi/ml-exercise/SSEnCE-merged
2. python3.7 main.py

Here we have the needed Python 3.7 libraries.

root@raspberrypi:~# pip3 list

Package	Version
absl-py	0.10.0
appdirs	1.4.4
astor	0.8.1
audioread	2.1.8
cachetools	4.1.1
certifi	2020.6.20
cffi	1.14.2
chardet	3.0.4
decorator	4.4.2
gast	0.2.2
google-auth	1.21.1
google-auth-oauthlib	0.4.1
google-pasta	0.2.0
grpcio	1.32.0
h5py	2.10.0
idna	2.10
importlib-metadata	1.7.0
joblib	0.16.0
Keras	2.4.3
Keras-Applications	1.0.8
Keras-Preprocessing	1.1.2
librosa	0.8.0
llvmlite	0.35.0.dev0+5.g4acef2d.dirty
Markdown	3.2.2
numba	0.52.0.dev0+162.gf88c3c5d1
numpy	1.19.1
oauthlib	3.1.0
opt-einsum	3.3.0
packaging	20.4
pandas	1.1.1
pip	20.2.2
plotly	4.9.0

Package	Version
pooch	1.1.1
protobuf	3.13.0
pyasn1	0.4.8
pyasn1-modules	0.2.8
pycparser	2.20
pyparsing	2.4.7
python-dateutil	2.8.1
pytz	2020.1
PyYAML	5.3.1
requests	2.24.0
requests-oauthlib	1.3.0
resampy	0.2.2
retrying	1.3.3
rsa	4.6
scikit-learn	0.23.2
scipy	1.5.2
setuptools	50.3.0
six	1.15.0
SoundFile	0.10.3.post1
tensorboard	2.0.2
tensorflow	1.14.0
tensorflow-estimator	1.14.0
termcolor	1.1.0
threadpoolctl	2.1.0
urllib3	1.25.10
Werkzeug	1.0.1
wheel	0.35.1
wrapt	1.12.1
zipp	3.1.0

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
                /input
            /docker
                docker-compose.yml
            /models
                no*_model.h5
            /raspberryPi
            /report
            class_validation.py
            cnn.py
            create_entity.py
            data-model.json
            functions.py
            pre-detection.ipynb
            sound_class_raspberry.py
            sound_classifier.py
            sound_featuring.py
            usound_classifier_part1.ipynb
            usound_classifier_part2.ipynb
```

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. Taking into account that the dataset is huge to upload them into the repository, in this repository is just indicated the directory where to leave that audio file that is going to be classify. The dataset used in this project it is UrbanSoun8K available on the web.

'input' in this folder we should save the audio file that we want to classify.

Other folders specifications

'docker' In this folder the doker-compose image is stored in which all the services that need to be deployed are specified in which it is used as a server to be able to instantiate the network designed for this project. This image must be displayed on that device that will act as a network server, since it specifies all the services necessary for its correct operation.

'models' This folder stores the already trained and validated model that will be imported into the classifier script in order to perform the classification.

'raspberryPi' This folder stores the entire directory that must be implemented within those sensor nodes (Raspberry Pi) that are going to be instantiated as classifiers within the network. This folder must be copied exactly to the Raspberry Pi that will be used as a classifier sensor, and the main.py file will be executed to launch its service.

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.

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.

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

create_entity: a new data entity is created in this file and sent to the Orion broker as indicated by the IP address of the server where the broker is hosted.

data_model(JSON): This .json file specifies the data structure to be used for each entity on the network.

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

pre-detection (NOTEBOOK): in this jupyter notebook the study carried out to determine the necessary audio filters to create a pre-detection filter is specified.

sound_class_raspberry: This is the main file of the project. This file is the one used to test the correct operation of the system. This file contains all the tasks of obtaining an audio fragment already recorded, its classification and sending it to the broker. It remains to specify in this file the function of capturing audio through the micro since this function is only implemented in those devices that will act as sensors (Raspberry Pi).

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.

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.

usound_classifier_part1 (NOTEBOOK): This jupyter notebook represents the study and obtaining of characteristics that is carried out in this project in order to train the neural network according to the urban sound classifier that you want to design.

usound_classifier_part2 (NOTEBOOK): This jupyter notebook specifies the CNN training, testing and validation phase that constitutes the project classifier. This way you can follow in a more interactive way than following the source code of the python scripts.

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, test_size=validation_size, random_state=seed)

# 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.py* 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. O directly you can use the main script to classify and send the result to the ner, by *sound_class_raspberry.py*