TECHNISCHE UNIVERSITÄT
ILMENAU

Masters Media Project

Realization of a Machine Learning Environment for Use in Audio Signal Processing

Instruction Manual

Supervisors: Dr.-Ing. Stephan Werner (stephan.werner@tu-ilmenau.de)

Dipl.-Ing. Florian Klein (florian.klein@tu-ilmenau.de)

Group Members: Muhammad Jami Raza (muhammad-jami.raza@tu-ilmenau.de)

Muhammad Miqdad Ali (muhammad-miqdad.ali@tu-ilmenau.de)

Alvaro De Jesus Blanco (alvaro.blanco@tu-ilmenau.de)

Contents

1 Abstract

Workstation (PC) has been set up in KirchoffBau (K-3010), TU Ilmenau, for the learning purpose in the field of Machine Learning and Neural Networks. Specifically in the field of audio classification. It is a part of the Master's Project. The goal is to deliver a platform for audio analysts and researchers to perform audio training, examine the results and, further fine-tune it to make the results more accurate.

This document is the guide on how to set up this system and how to use it. It contains the classification strategies applied on different application scenarios and on different datasets. It can be used to train models by tuning the parameters and predict as well on different pre-trained models.

It also contains the guide to use the Acoustic Learning Framework (ALF), which is a repository developed by the Fraunhofer IDMT.

Username: mpcuda
Password: D5g18-M

2 Introduction

Sound Classification [1] is an increasing area of research with various applications. While there is the increasing trend of application of music and speech classification, work on environmental sounds, scene detection was incomparable. But recent developments in the field of image classification where higher numbers of layers have been used to classify with high accuracy give way in this field.

So, the solution is to extract the frequency spectrum of audio signals and treat this as an image. The rest part will remain the same.

The same strategy has been applied to this Workstation. And different datasets have been used and classified.

2.1 Types of Machine Learning

Machine Learning [2] is used to understand the structure of data and then fit it into the models. Then, these models can be used by different people to predict the outcome for practical use. Types of machine learning are defined as how the different classes are made or learning is received from the data. There are many categories, but two mains are,

1. Supervised Learning
2. Unsupervised Learning

In Supervised learning, input and output variables both are available. We take an algorithm to learn the mapping function from input to output. Results are based on how accurately we approximated the mapping function.

In Unsupervised learning, only input variables are available, and learning is based on the distribution of data.

Neural Networks is a branch of machine learning or, more specifically, an advanced version of machine learning. Use to solve more complex problems. It contains highly interconnected units or nodes. Neural Networks is one set of algorithms used in machine learning for modeling the data using graphs of Neurons.

In this Workstation, all the classification is done using neural networks and, most of them are classified using supervised learning apart from one, i.e., Anomaly Detection.

2.2 Steps of Machine Learning

Machine Learning is done by implying various steps which should be done in order and carefully in order to do to perfect and to save time. If one step in between will not be done as it should be, then there will not be good results and, moreover, precious time will be wasted. Classically, there are seven steps,

1. Data Collection: Quality and quantity of data depend much on accuracy. Results are based on the representation of data that we are using for training. Data from Kaggle [3], UCI [4], etc., can be used.
2. Data Pre-processing: Data should be of the same features (no. of channels, bit depth, sampling rate). Most models require the same vectors of data for training. So, data should be normalized and then padding of data as well. Visualization of data is also necessary to detect relevant relationships between variables or class imbalances. They are finally splitted into training and test sets.
3. The architecture of a model.
4. Training
5. Evaluation: Testing of the model on unseen data
6. Parameter Tuning: It includes the tuning of hyperparameters like training steps, learning rate, initialization values, and distribution, etc.
7. Prediction

These steps have been done in all approaches. They can be employed again with some changes for better results and understanding of data.

2.3 Audio Classification

Audio classification is the classification of sound signals into different events. It also includes the classification of different auditory scenes. Recently, a breakthrough in this field has come after the adoption of deep models for image classification. Now, research is going in this segment (audio) also as researchers are now treating audio as an image by the extraction of STFT, MFCC, Mel-spectrogram, etc. This approach has given a major success, and now it is being employed in the real world also.

2.4 Application Scenarios

In this system, three application scenarios have been used. They have been selected after taking three different aspects of audio classification, and these three approaches are part of ongoing research.

1. Audio Event Detection: These include the classification of different real-world daily urban audios like dog bark, car horn, etc. The recognition of these events is helpful, e.g., for automatic tagging in audio indexing, automatic sound analysis for audio segmentation, or audio context classification.
2. Acoustic Scene Classification: Acoustic Scenes refers to the environment where the sounds belong, like airport, station, etc. These can be useful in robots, cars, intelligent monitoring systems, etc. However, significant research is still required in this field.
3. Anomaly Detection: Detection of mechanical failure in order to achieve industrial automation. The goal is to check whether sound from any mechanical unit is normal or an anomaly. This one also requires extensive development due to the lack of data of anomaly machines.

2.5 System Specifications

Processor: Intel(R) Core (TM) i7-4770K CPU @ 3.50GHz 3.50 GHz

Installed RAM: 16 GB

ROM: 2 TB

System Type: 64-bit operating system, x64-based processor

Operating System: Windows 10

OS Build: 19042.985

No. of cores of CPU: 4

2.6 CPU Specifications

No. of Cores: 4

No. of Threads: 8

Processor Base Frequency: 3.5 GHz

Max Turbo Frequency: 3.9 GHz

Bus speed: 5GT/s

2.7 Graphic Card Specifications

Memory: 4GB

Nvidia CUDA Cores: 768

Memory Speed: 7Gbps

3 Setup of Workstation

In this section, there is a brief guide to set up a system for training purposes. To train your model with a huge dataset and deep neural networks, complex computation is required, and that cannot be performed with a normal CPU. Highly dedicated systems are being built specifically to train models on them.

Note that, setup guide only contains the software part.

3.1 Setup of Graphic Card [5]

1. Download GeForce Experience by the following link,

   https://www.geforce.com/geforce-experience/download

   The account is required. Check for updates; if there are any, then it is necessary before going forward.

2. Download CUDA Toolkit
   This system contains Cuda 10.0. Always go for the latest
   https://developer.nvidia.com/cuda-10.0-download-archive?target_os=Windows&target_arch=x86_64&target_version=10&target_type=exelocal

3. Add CUDA Toolkit to the Path environment.

   a. Go to Properties>Advanced System Setting>Advanced (Tab)> Environment Variables.
   b. Go to the bottom side scroll-able window and select the Path variable by double-clicking on it.
   c. Add these three paths by clicking New, selecting them, and then OK.
      C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.1\bin
      C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.1\lib\x64
      C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.1\include

4. Download Nvidia CUDNN
   Nvidia Developer account is required.
   https://developer.nvidia.com/cudnn
   Cuda 7.6.X is compatible with this system.

5. Once downloaded, unzip it to these have 3 files and then:

   a) Copy <installpath>\cuda\bin\cudnn64_7.dll to C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.1\bin.
   b) Copy <installpath>\cuda\ include\cudnn.h to C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.1\include.
   c) Copy <installpath>\cuda\lib\x64\cudnn.lib to C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.1\lib\x64.

3.2 Setup of Environment [6]

1. The first step is obviously to install python. Installing through Anaconda is recommended.
   https://www.anaconda.com/distribution/
   Open Anaconda Navigator, then open Cmd.exe Prompt
   Then, type

```
conda update conda
```

2. Creation of environment

```
conda create -n aed python=3.7
```

"aed" is the name of the environment which can be different.
Any other version of python can be installed by specifying.

3. Activate Environment

```
conda activate aed
```

4. Install Tensorflow.

```
pip install tensorflow
```

The pip version is officially supported, while the conda version is community supported. Google suggests installing the pip version.
This will directly install the latest Tensorflow. It comes with Keras, so it does not need to be installed separately.

4 Audio Event Detection

Audio Event Detection or Sound Event Detection refers to the classification of daily basis urban sounds. The audio files generally consist of daily routine sounds, for example, dog bark, children playing, car horn, etc. These sounds can be of different categories and types, for example, human, mechanical, nature, etc. And can be of different lengths.

Currently, there is a research project named “Stadtlärm” [7], is going on with the collaboration of Fraunhofer IDMT [8], IMMS [9], Software-Service John GmbH [10] and Bischoff-Elektronik GmbH [11]. The goal is to develop a system for capturing, displaying, and predicting ambient noise occurring in urban areas.

In this Workstation, we have trained our models using novel approaches and made it understandable for another individual to train it otherwise by tuning the parameters and observe the results.

4.1 Datasets

Selection of dataset is the most important task when starting any project with machine learning. It is often termed as the fuel of the model. So, it should be balanced in terms of audio properties. For this purpose, we have chosen two well-known datasets that are US8k dataset and the ESC-50 dataset. As these datasets are huge, so we have made the subsets of these datasets and observe how well our model is performing on these subsets. So, in total, we have four datasets that are,

1. US8k_full [12]
2. US8k_sub
3. ESC-50 [13]
4. ESC-10

4.1.1 US8k_full

This dataset contains 8732 audio excerpts, each of them are =<4s. All audios are taken from field recordings uploaded to Freesound. The classes are air conditioning, car horn, children playing, dog bark, drilling, engine idling, gunshot, jackhammer, siren, and street music. These files are pre-sorted into ten folds. The dataset also contains the CSV file containing metadata. The audio files are in .wav format. The CSV file contains:

*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:
0 = air_conditioner
1 = car_horn
2 = children_playing
3 = dog_bark
4 = drilling
5 = engine_idling

6 = gun_shot
7 = jackhammer
8 = siren
9 = street_music
* class:
The class name: air_conditioner, car_horn, children_playing, dog_bark, drilling, engine_idling, gun_shot, jackhammer,
siren, street_music.

This dataset is downloaded in this system in folder at

```
C:\Users\mpcuda\Desktop\MT_Project_WS\Urban_Audio_Classifier\US8K_Dataset\UrbanSound8K\audio\
```

Its csv file, `UrbanSound8K.csv` is in,

```
C:\Users\mpcuda\Desktop\MT_Project_WS\Urban_Audio_Classifier\US8K_Dataset\UrbanSound8K\metadata\
```

The data distribution is,

```
engine_idling       1000
street_music        1000
jackhammer          1000
dog_bark            1000
children_playing    1000
air_conditioner     1000
drilling            1000
siren                929
car_horn             429
gun_shot             374
```

4.1.2 US8k_sub

This dataset is just the subset, but it is made with the same distribution as it is original. It contains 873 audios. The data distribution is,

```
engine_idling       100
street_music        100
jackhammer          100
dog_bark            100
children_playing    100
air_conditioner     100
drilling            100
siren                93
car_horn             43
gun_shot             37
```

The paths for audios and metadata are as follows.

```
C:\Users\mpcuda\Desktop\MT_Project_WS\Urban_Audio_Classifier\US8K_Dataset_sub\UrbanSound8K\audio\
C:\Users\mpcuda\Desktop\MT_Project_WS\Urban_Audio_Classifier\US8K_Dataset_sub\UrbanSound8K\metadata\UrbanSound8K_sub.csv
```

Note that this dataset is arranged in only one folder (fold1).

4.1.3 ESC-50

This dataset is a collection of 2000 audios divided into 50 classes making it 40 audios per class. It consists of 5-sec long recording. It is loosely arranged into five different categories. The following table shows the class distribution.

Animals	Natural soundscapes & water sounds	Human, non-speech sounds	Interior/domestic sounds	Exterior/urban noises
Dog	Rain	Crying baby	Door knock	Helicopter
Rooster	Sea waves	Sneezing	Mouse click	Chainsaw
Pig	Crackling fire	Clapping	Keyboard typing	Siren
Cow	Crickets	Breathing	Door, wood creaks	Car horn
Frog	Chirping birds	Coughing	Can opening	Engine
Cat	Water drops	Footsteps	Washing machine	Train
Hen	Wind	Laughing	Vacuum cleaner	Church bells
Insects (flying)	Pouring water	Brushing teeth	Clock alarm	Airplane
Sheep	Toilet flush	Snoring	Clock tick	Fireworks
Crow	Thunderstorm	Drinking, sipping	Glass breaking	Hand saw

Clips in this dataset have been manually extracted from public field recordings gathered by the Freesound. The csv file contains,

*filename:
The name of the audio file. The name takes the following format: {FOLD}-{CLIP_ID}-{TAKE}-{TARGET}.wav
{FOLD} - index of the cross-validation fold,
{CLIP_ID} - ID of the original Freesound clip,
{TAKE} - letter disambiguating between different fragments from the same Freesound clip,
{TARGET} - class in numeric format [0, 49].
*fold:
number of folds which is 1 of every entity.
*target
*category
*esc-10:
those audios which are included in esc-10 dataset. It is Boolean value.
*src_file:
The Freesound ID of the recording from which this excerpt (slice) is taken.
*take

This dataset is downloaded in this system in the folder at

```
C:\Users\mpcuda\Desktop\MT_Project_WS\Urban_Audio_Classifier\ESC-50_Dataset\audio\
```

Its CSV file, `esc50.csv` is in,

```
C:\Users\mpcuda\Desktop\MT_Project_WS\Urban_Audio_Classifier\ESC-50_Dataset\metadata\
```

4.1.4 ESC-10

This subset contains the audio from 10 classes of the ESC-50 dataset. So, this contains only 400 audios (40 from each class). The selected classes are “chainsaw “, “clock tick", "crackling fire", "crying baby", "dog “, "helicopter", "rain “, „rooster", "sea waves" and sneezing". The path for audios is the same as in ESC-50. Metadata path is,

```
C:\Users\mpcuda\Desktop\MT_Project_WS\Urban_Audio_Classifier\ESC-10_Dataset\metadata\esc10.csv
```

4.2 Environment Setup

First, create the conda environment. Go to Anaconda Navigator and open Cmd.exe Prompt.

Type,

```
conda create -n aed python=3.7
```

Install Tensorflow [14], Keras [15], LibROSA [16], Ipython [17], NumPy [18], Pandas [19], Matplotlib [20], and SciKit Learn [21] in your environment. Following are the respective commands to install these libraries,

```
conda install -c anaconda tensorflow-gpu
conda install -c conda-forge keras
conda install -c conda-forge Librosa
conda install -c anaconda ipython
conda install numpy
conda install pandas
conda install -c conda-forge matplotlib
conda install -c conda-forge scikit-learn
```

Environment “aed” has the following versions.
Tensorflow (2.5)
Keras (2.4.3)
Librosa (0.8)
Ipython (7.22.0)
Numpy (1.19.5)
Pandas (1.2.4)
Matplotlib (3.4.1)
Scikit-learn (0-24.1)

4.3 Approach

As we have discussed earlier that the idea is to extract the time or frequency representation of audio and use it as an image. For this purpose, we have many options, and each of them can be used, but the problem is to have true representation as well as the optimal or at least reasonable results. The simplest one is to feed the network with serial audio data, and it has been proven that the network is capable of learning patterns for sound recognition with this approach, but the problem is results that are not satisfactory.

Another option is spectrogram which is a visual representation of the spectrum of frequencies of the signal as it varies with time. But spectrogram uses the linear scale, and it is completely opposite of how the human auditory system works.

To build up a spectrogram, a windowing function is applied to each audio frame, then the Short-Term Fourier Transform (STFT) is computed on each frame; finally, the power spectrum, and filter banks are computed.

MFCC (Mel-Frequency Cepstral Coefficients) proved to perform better results. MFCC converts the frequency to mel-scale, which is logarithmic. To generate MFCCs, a Discrete Cosine Transform (DCT) is applied to the filter banks obtained in the last process, retaining a few resulting coefficients while the rest are discarded.

Again, there is one problem that DCT is used to decorrelate filter banks coefficients. DCT, as a linear transform, will discard some information from the original signal. Once the filter bank had been computed, they are scaled to the MEL scale to obtain the MEL-Scaled filter banks.

In this Workstation, we have extracted MFCC and mel-spec to observe the results. They can be extracted using the Librosa library. These features are extracted and then saved into numpy arrays to feed it into our model.

Then, data is divided into train and test sets and the feed into a basic 4-layer CNN model. Model is the saved, and then predictions can be made using those models.

4.4 Preprocessing

During our data analysis, we have discovered that the US8K dataset is not a fully balanced dataset. Most of the audios are greater than 3 sec, 14.4 % are less than 3 sec, and 8.5% are less than 1 sec. 91.53% are stereo, and the rest are mono. There is great variance in the bit depth. They vary from 4-bit to 32-bit. And there is a great difference of quality between them. Same is the case of sampling rate. They vary from 8 kHz to 192 kHz.

We have used the Librosa library, which automatically converts audio into 22500 Hz and 16 bits. It is also used to normalize the audio files from -1 to 1. You just upload the data, and they are ready to work.

ESC-50 is a balanced dataset. Mono channel, 16 bit-depths, and 44100 Hz sampling rate.

Data analysis of any audio dataset can be done by running the file `data_analysis.py`. This file is present in every folder of all the above datasets. For example, for Urbansound8k_full,

```
C:\Users\mpcuda\Desktop\MT_Project_WS\Urban_Audio_Classifier\US8K_Dataset\UrbanSound8K\data_analysis.py
```

If you want to edit the file, right-click it and open it through Notepad++. Data analysis of any file can be done by changing the folder path here,

Set your path to the original dataset,

```
us8k_path=os.path.abspath(r"C:\Users\mpcuda\Desktop\MT_Project_WS\Urban_Audio_Classifier/US8K_Dataset/UrbanSound8K/")
```

And metadata path here,

```
metadata_path = os.path.join(us8k_path, 'metadata/UrbanSound8K.csv')
```

Column names should be called according to the targeted CSV file. If there is something missing in CSV then comment it like in the `esc50.csv` we did not have the duration column so, we have commented on it.

4.5 Feature Extraction [22]

In this step, we will extract MFCC and Mel-spec of each audio and save it into numpy arrays. They have been extracted in `pre_processing.py`, present in every folder of all the above datasets. For example, for Urbansound8k_full,

```
C:\Users\mpcuda\Desktop\MT_Project_WS\Urban_Audio_Classifier\US8K_Dataset\UrbanSound8K\pre_processing.py
```

First, spectrogram, MFCC, and Mel-spec of the first file are extracted to observe the one.

We have extracted 40 coefficients per frame, and as the frame count is variable, we are also adding zero padding to those samples that do not reach the maximum (padded is added at both sides of the feature).

Then, they have been saved at the following path.

```
C:\Users\mpcuda\Desktop\MT_Project_WS\Urban_Audio_Classifier\US8K_Dataset\UrbanSound8K\data\X-mfcc_us8k_full.npy

C:\Users\mpcuda\Desktop\MT_Project_WS\Urban_Audio_Classifier\US8K_Dataset\UrbanSound8K\data\y-mfcc_us8k_full.npy
```

`X-*.npy` contains the features of audios, and `y-*.npy` contains the labels.

Following are the parameters for extraction:
For MFCC, 40 coefficients per frame.
For Mel-spec, n_fft =2048, hop length = 512, n_mel=128.

Set the dataset path, metadata path and the targeted path where you want to save the .npy files. Targeted path can be changed by changing the following,

For MFCC,

```
np.save("data/X-mfcc", X)
np.save("data/y-mfcc", y)
```

Or,

For Mel-spec,

```
np.save("data/X-mel_spec", X)
np.save("data/y-mel_spec", y)
```

As far as the extraction of other features of the audio files according to the requirements of the use, it can be easily extracted by creating the separate function in `helpers.py`. Librosa library can extract the spectral features such as CQT (Constant Q Transform), Spectral Centroid, Zero Crossing Rate as well rhythmic features such as Tempogram. For Example, to extract CQT the Librosa command will be,

```
librosa.feature.cqt(numpy array, sr=22050, hop_length=512, n_bins=84)
```

4.6 Training

The next step is feeding these numpy arrays in our model. We have trained by using MFCC and Mel-spec both in `cnn_training_mfcc.py` and `cnn_training_mel_spec.py`.

First, splitting the dataset in Train/Test split by 80/20. The next step is to encode the labels with values between 0 and n_classes-1. Reshape the tensor to feed it into the network. We have used the channel last.

Then, the architecture of the network is defined in the `create_model` function. It is a 4-layer CNN model. 32 filters are used for first two layers and 64 for the other two. Kernel size is (3,3). Regularization is used with a value of 0.0005.
The activation function, "LeakyRelu" is used with 0.1 alpha rate. Each layer is followed by Batch normalization. Max Pooling (2,2) is added after the 2nd layer and Global Max pooling after the 4th layer. Spatial Dropout has been used after the first three-layer with a rate of 0.07, increasing to 0.14 when used after the third layer. In the end, the Dense layer is used with the "SoftMax" function and size equal to the no. of labels (i.e., 10 for UrbanSound8k).
Adam Optimizer has been used with the lr=1e-4, beta_1=0.99, beta_2=0.999 values and model is compiled using categorical_crossentropy as loss function.
We have set the batch size = 128. It should be selected according to the system capacity.
All these values have been taken from the pretested approach. These can be changed according to the datasets.
Training is then started using `model.fit`.

We first load the checkpoint with the lowest validation loss achieved during training. We then use the Test set to evaluate the model with data that has not been seen by the model nor by me when tuning hyper-parameters. Then, graphs of accuracy and loss will appear. After then, we have plotted the confusion matrix, class-wise accuracy table and precision, recall, and f1-support table.
This model can hyper tuned by changing the above values and then change the name of the model file here,

```
model_file = 'us8k_full_mel_spec.hdf5'
```

Models are saved to

```
C:\Users\mpcuda\Desktop\MT_Project_WS\Urban_Audio_Classifier\US8K_Dataset\UrbanSound8K\models\
```

4.7 Augmentations [23]

Data augmentations in data analysis are strategies used to increase the amount of data by adding modified copies of existing data or newly created synthetic data from existing data. It acts as a regularize and helps reduce overfitting when training a machine learning model prediction.
Augmentation techniques which we have used here are,

1. Time Stretch
2. Pitch Shift
3. Noise

Time Stretching is changing the speed of audio without changing its pitch. We have used here the rates of 0.81 and 1.07.
Pitch shifting is raising or lowering of the original pitch of the sound. We have used here the values of -1, -2, 1, 2 to shift the pitches.
We have also added random noise to our audio.
In this way, we have increased our no. of audios to 8 times.

These values can be changed but user should be aware of the limitations of the augmentation strategy and apply it according to the nature of the audio file.

This is done in `augmentation.py`, and then a new augmented CSV is created, which has an extra column of `augment`. The new CSV file is,

```
C:\Users\mpcuda\Desktop\MT_Project_WS\Urban_Audio_Classifier\US8K_Dataset\UrbanSound8K\data\augmented-data_us8k_full.csv
```

Again, then features have been extracted in `augmented_pre_processing.py`, and new numpy arrays have been created.
Now, we have trained a new model using augmented Mel-spec in,

```
augmented_cnn_training_mel_spec.py
```

New model files have been created in the `models` folder.

Again here, more augmentation techniques can be applied in order to get the optimal results. User just have to create a separate function in `augmentations.py` file. Other effects can be applied such as remix of an audio, separation of precursive and harmonic components, trim, Mix-up. For Example, for remix

```
librosa.effects.remix(numpy array , intervals, align_zeros=True)
```

4.8 Prediction

Now, prediction using the pre-trained models can also be done. For this purpose, we must create the numpy arrays of our test dataset. This time only `X-mfcc_us8k_prediction.npy` and

`X-mel_spec_us8k_prediction.npy` are created as we have to predict the labels using our model. This can be done using `prediction_pre_processing.py`. The path should be set for dataset, metadata, and for targeted .npy files.
Labels will be predicted by using the `prediction.py` file. Specify the path of the model. Another .txt file, `predictions.txt`, is created, which has the predicted labels for unknown data.

4.9 Acoustic Learning Framework (ALF)

ALF is a toolbox created by the Fraunhofer IDMT to help make deep learning easier to understand for non-experts and enable faster model improvement iteration cycles for experienced machine learning developers and researchers alike.

Basically, here we define our path to the dataset, feature extraction, and model architecture in a single .json file.
A JSON extension is a file that has the simple data structures and objects in JavaScript Object Notation (JSON) format and is a standard data interchange format. It is mainly used for transmitting data between a web application and a server.

First, create an environment,

```
conda create -n alfpaka python==3.7
```

Then, activate it,

```
conda activate alfpaka
```

These libraries should be installed in this environment,

1. tensorflow 2.3
2. tqdm
3. matplotlib
4. pandas
5. Librosa
6. soundfile
7. albumentations
8. scikit-learn
9. openpyxl
10. PyYAML

Dataset files should be arranged in such a manner that each label has its own folder, and it contains the audio files of its category. So, we had made the datasets of UrbanSound8K_full, UrbanSound8K_sub, and esc_10. We had also created the separate .son files as `SED_us8k_full.json`, `SED_us8k_sub.json`, and `SED_esc_50.json`.

All the datasets and there .json files are at,

```
C:\Users\mpcuda\Desktop\MT_Project_WS\Urban_Audio_Classifier\ALF\alf-v0.2.1_alpha\
```

Then, in .json file, specify the dataset path in “dir_train”. If there is any test folder, then specify at “dir_test”. The pattern of audios is specified at “train_pattern” and “test_pattern”. We have specified here as “*.wav”. Classes will be specified at “classes”.

```
"dir_train":"Alf_audio_dataset_us8k_full/",
"dir_test":null,
"train_pattern":"*.wav",
"test_pattern":null,
```

```
"classes":["air_conditioner",
        "car_horn",
        "children_playing",
        "dog_bark",
        "drilling",
        "engine_idling",
        "gun_shot",
        "jackhammer",
        "siren",
        "street_music"],
```

Then, training settings will be set such as "data_split_ratio", "extract_features", "number_of_cv_steps", "num_epochs", "batch_size". Set it according to your requirements.

```
"data_split_ratio":0.8,
"extract_features":true,
"number_of_cv_steps":5,
"num_epochs":53,
"batch_size":512,
```

Then, you must select the pre-build models which are built-in,

```
C:\Users\mpcuda\Desktop\MT_Project_WS\Urban_Audio_Classifier\ALF\alf-
v0.2.1_alpha\ALFpaka\core\general_models.py
```

Different models like Dense Neural Network in function `createDNNmodel`, Convolutional 1D in `createModelCNN1D`, Convolutional 2D in `createModelCNN2D`, and Resnet in `createResNet` are already present there. You just have to call them and specify their parameters as we have in our .json files.

```
"create_model":"createModelCNN2D",
"model_parameter": {
  "layer_dims": [[32,[3,3]],[32,[3,3]],[64,[3,3]],[64,[3,3]]],
  "dropout": 0.007,
  "l2":0.001,
  "activation":"relu",
  "activation_output":"softmax",
  "addpooling":"addpooling",
  "batchNorm":"True",
  "padding":"valid",
  "final_global_pooling":"avg"}
```

Then, the extraction method should be specified. Likewise, there are also some of them prebuild in,

```
C:\Users\mpcuda\Desktop\MT_Project_WS\Urban_Audio_Classifier\ALF\alf-
v0.2.1_alpha\ALFpaka\core\ general_extract_methods.py
```

Different feature like STFT in `extractSTFT`, STFT 2D in `extractFeaturesSTFTCNN2D`, CQT in `extractCQT` and MFCC in `extractMFCC` are defined. And there parameters should be set like,

```
"extract_method_to_use":"extractFeaturesSTFTCNN2D",
"extract_feature_parameter": {
  "fftsize": 256,
  "mel_bands":40
},
```

Other things should also be specified like,

```
"normalize_function": "normalizeToZeroMeanPerBin",
"optimizer":"tensorflow.keras.optimizers.Adam",
```

```
"optimizer_parameter": {
  "learning_rate": 1e-3
}
```

In Terminal, after activating the ALF environment type and then go to the ALF folder,

```
cd C:\Users\mpcuda\Desktop\MT_Project_WS\Urban_Audio_Classifier\ALF\alf-v0.2.1_alpha
```

Then, type the following to start training,

```
python training_main.py -t temp_us8k_full -c SED_us8k_full.json -o results_us8k_full
```

`temp_us8k_full` and `results_us8k_full` is the folder for temporary files and result files, you can give any other folder or path.

5 Acoustics Scene Detection

Acoustics Scene Detection and Classification refers to classifying the audio recording as one of the predefined classes that characterize the environment in which it was recorded. These environments constitute the urban scenes such as airport, bus stop, mero-train, public square. It can be an open environment or the scene of transportation such as bus or tram.

This kind of detection can be helpful in numerous applications such as hearing aid, monitoring natural habitats, smart cities, autonomous navigation.

In this Workstation, we have worked on two datasets. We have analyzed the results by extracting two different features and augmenting our data. In this approach, we have also extracted the Binaural features and observe the difference in results.

5.1 Datasets

Unlike the Urban Event Detection, the variety of datasets is very limited here. There are not many open datasets available in this context. If there are some, then they may not be balanced. This limited availability is the intensity of the complexity of the problem, which is very high. The environmental scenes may contain noises and temporary sounds, which technically should not be part of that scene. The other reason is the device from which it is being recorded. We have chosen two datasets for this task.

1. TUT Urban Acoustics Scene 2018 Dataset [24]
2. Automatic Spatial Audio Dataset [25]

5.1.1 TUT Urban Acoustics Scene 2018 Dataset

Detection and Classification of Acoustics Scene and Events (DCASE) is a community that organizes the annual challenge and workshop for the researchers to work in the field of environmental sound detection and share their ideas. So, for every challenge, they provide a dataset with different complexities. The dataset for Acoustics Scene Classification is originally collected by the Tampere University of Technology in 2016. The original dataset consists of 30-seconds audio segments from 15 different classes. The main recording device consists of Soundman OKM II Klassik/studio A3 [26], electret binaural microphone, and a Zoom F8 audio recorder [27] using a 48kHz sampling rate and 24-bit resolution. These audios are classified as “A.” The other devices are common smartphones and recorders. These audios are classified as “B” and “C.” These audios are recorded in European metropolitan cities like London, Paris, Prague.

DCASE prepares a unique dataset from the original one according to the task complexities. TUT Urban Acoustics Scene 2018 Dataset is the 2018 dataset with a length of 10 seconds. They are from 10 classes which are,

Airport - `airport`
Indoor shopping mall - `shopping_mall`
Metro station - `metro_station`
Pedestrian street - `street_pedestrian`
Public square - `public_square`
Street with medium level of traffic - `street_traffic`
Travelling by a tram - `tram`
Travelling by a bus - `bus`
Travelling by an underground metro - `metro`
Urban park – `park`

This workstation consists of 6 datasets files from TUT Urban Acoustics Scene 2018 Dataset in,

```
C:\Users\mpcuda\Desktop\MT_Project_WS\Acoustics_Scene_Classification\Dataset_DCASE_2018,
```

- TUT-urban-acoustic-scenes-2018-development-This folder consists of 864 audio files from device “A.” Its metadata is labeled, and it is divided into a 70/30 train-test split. These audios are stereo.
- TUT-urban-acoustic-scenes-2018-evaluation-This folder consists of 3600 unlabeled audios. These audios are from devices all devices.
- TUT-urban-acoustic-scenes-2018-leaderboard-This folder consists of extra evaluation audios for the challenge.
- TUT-urban-acoustic-scenes-2018-mobile-development-This folder consists of all devices but majorly from” A.” Theses constitute 10080 files, but all are single-channel and resampled.
- TUT-urban-acoustic-scenes-2018-mobile-evaluation-This folder consists of test unlabeled data from all devices. These are 15120 audio files.
- TUT-urban-acoustic-scenes-2018-mobile-leaderboard- This folder consists of extra evaluation audios for the challenge.

5.1.2 Automatic Spatial Audio Dataset

First, 152 adios were selected (112 for training and 40 for testing). These audios are the anechoic and semi-anechoic recordings of the orchestra and opera. Then, it is preprocessed, then RMS-level normalized, and then stored as an uncompressed file with 24-bit resolution at a 48 kHz sample rate. Then, these audios are synthesized with the 13 databases of BRIR. These databases are,

S.No.	Acronym	Description	Dummy Head	RT60(s)
1	sbs	Salford, British Broadcasting Corporation (BBC) – Listening Room	B&K HATS Type 4100	0.27
2	mair	Huddersfield-Concert Hall	Neumann KU100	2.1
3	thkcr1	Westdeutscher Rundfunk (WDR) Broadcast Studios–Control Room 1	Neumann KU100	0.23
4	thkcr7	WDR Broadcast Studios–Control Room 7	Neumann KU100	0.27
5	thksbs	WDR Broadcast Studios–Small Broadcast Studio (SBS)	Neumann KU100	1.0
6	thklbs	WDR Broadcast Studios–Large Broadcast Studio (LBS)	Neumann KU100	1.8
7	calypso	Technische Universität (TU) Berlin–Calypso Room	KEMAR 45BA	0.17
8	tvtui	Hall TU Ilmenau–TV Studio (distance of 3.5m)	KEMAR 45BA	0.7
9	tvtui2	TU Ilmenau–TV Studio (distance of 2m)	KEMAR 45BA	0.7
10	labtui	TU Ilmenau–Listening Laboratory	KEMAR 45BA	0.3
11	rehabtui	Concert TU Ilmenau–Rehabilitation Laboratory	KEMAR 45BA	NA
12	tuburo250	University of Rostock–Audio Laboratory (additional absorbers)	KEMAR 45BA	0.25
13	tuburo310	tuburo310 University of Rostock–Audio Laboratory (all broadband absorbers)	KEMAR 45BA	0.31

Overview of BRIR sets [28]

In total 4368 excerpts were synthesized for training and 1560 for testing. They are classified into three categories and are classified concerning a horizontal distribution of foreground and background audio content. Foreground refers to identifiable and perceived audio. Background refers to the unimportant, unclear, and “foggy” sounds.

The three classes are;

1. Foreground- Background: A listener perceives foreground audio content in the front and background content behind the head.
2. Background- Foreground: A listener perceives background audio content in the front and foreground content behind the head.
3. Foreground- Foreground: A listener is surrounded by foreground audio content.

These audio files are kept as separate training and testing files as,

```
C:\Users\mpcuda\Desktop\MT_Project_WS\Acoustics_Scene_Classification\Dataset_Automatic_Spatial_Audio_Classification\training_data

C:\Users\mpcuda\Desktop\MT_Project_WS\Acoustics_Scene_Classification\Dataset_Automatic_Spatial_Audio_Classification\testing_data
```

5.2 Environment Setup

First, create the conda environment. Go to Anaconda Navigator and open Cmd.exe Prompt.

Type,

```
conda create -n asc python=3.6
```

Install Tensorflow, Keras, LibROSA, IPython, NumPy, Pandas, Matplotlib, and SciKit Learn in your environment. Following are the respective commands to install these libraries,

```
conda install -c anaconda tensorflow-gpu
conda install -c conda-forge keras
conda install -c conda-forge Librosa
conda install -c anaconda ipython
conda install numpy
conda install pandas
conda install -c conda-forge matplotlib
conda install -c conda-forge scikit-learn
```

Environment “asc” has the following versions.
Tensorflow (1.14)
Keras (2.3.1)
Librosa (0.6.1)
Ipython (7.16.1)
Numpy (1.19.5)
Pandas (0.25.3)
Matplotlib (2.2.2)
Scikit-learn (0-19.1)
Audioread (2.1.6)

5.3 Approach

In Acoustics Scene Classification, we have used the same approach to extract the features as in Event Detection. Some additional changes are that we have extracted the features of the DCASE dataset by using both channels. We have used this strategy because of the difficult prediction of the scenes. The prediction of scenes is much difficult here because of the audios’ noise and unwanted sound events. Another reason is that both channels of these audios have been recorded. If one is using a mono channel then, a huge amount of information has been discarded.

Another difference is that the layers of the model have been increased. Here, we have used the 8-Layer CNN model to train the audio features on it. Preprocessing strategies and Model structure can also be changed easily to make them more optimal.

5.4 Feature Extraction

For TUT Urban Acoustics Scene 2018 Dataset, we have extracted only Mel-spectrogram of audio files here. The parameters which we have set here are,

```
sample_rate = 44100
window_size = 2048
overlap = 672
seq_len = 320
mel_bins = 64
```

It can be changed according to the requirements in file,

```
C:\Users\mpcuda\Desktop\MT_Project_WS\Acoustics_Scene_Classification\Approach_DCASE\dcase2018_task1-master\utils\config.py
```

Features can be extracted by running the file,

```
C:\Users\mpcuda\Desktop\MT_Project_WS\Acoustics_Scene_Classification\Approach_DCASE\dcase2018_task1-master\utils\features.py
```

The python command has been set up to specify the dataset folders and the target feature folders for each of the datasets. For example, for `TUT-urban-acoustic-scenes-2018-development`, first go to the desired folder by command,

```
Cd
C:\Users\mpcuda\Desktop\MT_Project_WS\Acoustics_Scene_Classification\Approach_DCASE\dcase2018_task1-master
```

And then extract features by,

```
python utils/features.py logmel --
dataset_dir="C:\Users\mpcuda\Desktop\MT_Project_WS\Acoustics_Scene_Classification\Dataset_DCASE_2018" --subdir="TUT-urban-acoustic-scenes-2018-development" --
data_type=development --
workspace="C:\Users\mpcuda\Desktop\MT_Project_WS\Acoustics_Scene_Classification\Approach_DCASE\dcase2018_task1-master"
```

`logmel` in the command specifies the type of feature which we are going to extract, if someone wants to extract another one then, a separate function should be made and then called accordingly. `--dataset_dir` and `--subdir` is the dataset directories. `--data_type` is the type of the datasets of `TUT Urban Acoustics Scene 2018 Dataset`, `leaderboard`, and `evaluation` is the other two types.

Then, these features are stored as a .h5 file. H5 file is a data file stored in a hierarchical data format. It contains multidimensional arrays of scientific data. Features are extracted for all six datasets. And are stored at,

```
C:\Users\mpcuda\Desktop\MT_Project_WS\Acoustics_Scene_Classification\Approach_DCASE\dcase2018_task1-master\features\logmel
```

Then, each of the folder contain it's `*.h5` feature file.

For the Automatic Spatial Audio Dataset, we have extracted MFCC and mel-spec both and then stored them in a .npy file. It can be extracted by using,

```
C:\Users\mpcuda\Desktop\MT_Project_WS\Acoustics_Scene_Classification\Dataset_Automatic_Spatial_Audio_Classification\pre_processing.py
```

And parameters can be changed in the file,

```
C:\Users\mpcuda\Desktop\MT_Project_WS\Acoustics_Scene_Classification\Dataset_Automatic_Spatial_Audio_Classification\include\helpers.py
```

The resulted .npy files are stored at,

```
C:\Users\mpcuda\Desktop\MT_Project_WS\Acoustics_Scene_Classification\Dataset_Automatic_Spatial_Audio_Classification\data
```

5.5 Training

For TUT Urban Acoustics Scene 2018 Dataset, the training is done by using,

```
C:\Users\mpcuda\Desktop\MT_Project_WS\Acoustics_Scene_Classification\Approach_DCASE\dcase2018_task1-master\pytorch\main_pytorch.py
```

The file to train using Keras framework is also there and training can also be done by using `keras\main_keras.py`. The training has been done by using the command,

```
python pytorch/main_pytorch.py train --dataset_dir="C:\Users\mpcuda\Desktop\MT_Project_WS\Acoustics_Scene_Classification\Dataset_DCASE_2018" --subdir="TUT-urban-acoustic-scenes-2018-development" --workspace="C:\Users\mpcuda\Desktop\MT_Project_WS\Acoustics_Scene_Classification\Approach_DCASE\dcase2018_task1-master" --validate --holdout_fold=1 -cuda
```

`train` here is the type of operation. To evaluate the performed training `--inference_validation_data` should be used to evaluate the unknown data in `evaluation` and `leaderboard` files, `--inference_evaluation_data and --leaderboard` should be used, respectively. Here, `--dataset_dir` and `--subdir` are the same as used before. To evaluate the unknown files, the training dataset should be specified as `--dev_subdir` and the targeted directory as `--eval_subdir` or `--leaderboard_subdir`. `-workspace` is the folder to store all the files. If validation of data is required, then `-validate` should be used. `--holdout_fold` is the number of training.

All the parameters for training can be changed in the same file, i.e., `main_pytorch.py`. However, the architecture should be specified in `models_pytorch.py`.

For Automatic Spatial Audio Dataset, we have trained using the file,

```
C:\Users\mpcuda\Desktop\MT_Project_WS\Acoustics_Scene_Classification\Dateset_Automatic_Spatial_Audio_Classification\cnn_training_mfcc.py

C:\Users\mpcuda\Desktop\MT_Project_WS\Acoustics_Scene_Classification\Dateset_Automatic_Spatial_Audio_Classification\cnn\training_mel_spec.py
```

Here also the training parameters and model architecture can be changed easily.

First, splitting the dataset in Train/Test split by 70/30. The next step is to encode the labels with values between 0 and n_classes-1. Reshape the tensor to feed it into the network. We have used the channel last.

Then, the architecture of the network is defined in the `create_model` function. The model we have used in the aforementioned case is an 8-layer CNN model named VGG (Visual Geometry Group). 64 filters are used for the first two layers, 128 for the other two, 256 for the next two, and 512 for the last two. the Kernel size is (3,3).
The activation function, "Relu," is used. Each layer is followed by Batch normalization. Max Pooling (2,2) is added after the 2nd,4th and 6th layer and Global Max pooling after the 8th layer. In the end, the Dense layer is used with the "SoftMax" function and size equal to the no. of labels (i.e., 10 for TUT Urban Acoustics Scene 2018 Dataset).
Adam Optimizer has been used with the lr=1e-4, beta_1=0.99, beta_2=0.999 values and model are compiled using categorical_crossentropy as loss function.
We have set the batch size = 24. It should be selected according to the system capacity.
All these values have been taken from the pretested approach. These can be changed according to the datasets.
Training is then started using `model.fit`.
Models are saved to,

```
C:\Users\mpcuda\Desktop\MT_Project_WS\Acoustics_Scene_Classification\Approach_DCASE\dcase2018_task1-master\pytorch\main_pytorch\holdout_fold=1
```

(for TUT Urban Acoustics Scene 2018 Dataset)

```
C:\Users\mpcuda\Desktop\MT_Project_WS\Acoustics_Scene_Classification\Dateset_Automatic_Spatial_Audio_Classification\models
```
(for automatic spatial audio)

5.6 Augmentations

We have used the same augmentation strategies, as in Event Detection i.e.,

1. Time Stretch
2. Pitch Shifting
3. Noise

They are done in the `Augmentation` folder. Separate python files have been made for each dataset, and their respected .csv files are also created. Now, their .npy files can be created easily. And then, it can also be trained using the same training files. Likewise, augmentation parameters can also be altered.

Other strategies can also be applied by specifying the separate function.

5.7 Binaural

The TUT-urban-acoustic-scenes-2018-development and Automatic spatial audio are the stereo channel audios. Moreover, to get the maximum result out of these files is to extract the features of both channels. In the above training, we were using the mono channel. The strategy was that the average of two channels was taken and then extracted the single feature. However, here we have extracted both channels feature separately and then save them into a single .npy file. Then, this file is trained using the same training file. Note that the numpy file will take double space as compared to the former one. We have extracted the binaural features in,

```
C:\Users\mpcuda\Desktop\MT_Project_WS\Acoustics_Scene_Classification\Approach_Binaural\ pre_processing_dcase_2018_mel_spec_binaural.py
```

And then it is being trained by using the `cnn_training_mel_spec.py`. The model is saved in,

```
C:\Users\mpcuda\Desktop\MT_Project_WS\Acoustics_Scene_Classification\Approach_Binaural\models
```

5.8 ALF

As we have used ALF earlier, the same is the case here. First, we had made the separate datasets from the TUT-urban-acoustic-scenes-2018-development and Automatic spatial audio in which each folder has the files from the same class. These datasets are arranged in `the dcase_2018_dataset_alf_train` and `asa_dataset_alf_train`, respectively. Then we have made a separate JSON file in which the training parameters have been specified. For example, for TUT-urban-acoustic-scenes-2018-development,

```
{
"dir_train":"dcase_2018_dataset_alf_train/",
"dir_test ": null,
"train_pattern":"*.wav",
"test_pattern": null,
"classes":
["airport","bus","metro_station","metro_train","park","public_square","shopping_mall","street_pedestrian","street_traffic","tram"],
"data_split_ratio":0.8,
"extract_features":true,
"number_of_cv_steps":1,
"num_epochs":100,
"batch_size":64,
"create_model":"createModelCNN2D",
"model_parameter": {
```

```
  "layer_dims":
[[64,[3,3]],[64,[3,3]],[128,[3,3]],[128,[3,3]],[256,[3,3]],[256,[3,3]],[512,[3,
3]],[512,[3,3]]],
  "dropout": 0,
  "l2":0,
  "activation":"relu",
  "activation_output":"softmax",
  "addpooling":"addpooling",
  "batchNorm":"True",
  "final_global_pooling":"avg"},
"extract_method_to_use":"extractFeaturesSTFTCNN2D",
"extract_feature_parameter": {
  "fftsize": 2048,
  "mel_bands":40
},
"normalize_function": "normalizeToZeroMeanPerBin",
"optimizer":"tensorflow.keras.optimizers.Adam",
"optimizer_parameter": {
  "learning_rate": 1e-3
}
}
```

Then, the training can be started by,

```
python training_main.py -t temp_dcase_2018 -c dcase_2018_train.json -o
results_dcase_2018
```

6 Anomaly Detection

Anomaly Detection is the task of identifying whether the sound from a machine or any other targeted object is normal, or it is an outlier. Automatic Detection of failure is the need for the future industries, and a huge amount of research is going in this aspect. Also, this can be useful in machine condition monitoring.

This one is unsupervised machine learning, where we will train our model on the datasets of sounds of the selected machines. The challenge in this approach is that the availability of only normal sounds. Anomalous sounds rarely occur in industries as compared to normal ones. So, training will be done on normal sounds, and then it is evaluated on unknown audios to check how far it is from the normal one. If they are within their allowed limits, then these will be categorized as normal, and if not, then the intensity of the difference will be measured to check whether it can be repaired or should be replaced.

DCASE is also organizing the challenge in this application. They have included Anomaly in their 2020 challenge. In this Workstation, we have worked on two datasets provided on their website.

6.1 Dataset

We have used the two datasets here. Both have been downloaded from the DCASE website.

1. 2020_Dataset [29]
2. 2021_Dataset [30]

6.1.1 2020_Dataset

This dataset contains the parts of ToyADMOS and the MIMII Dataset of normal and anomalies from the six different parts/machines from the daily use. The audios are single-channel, 10-sec length, and containing some noise. The parts are,

- Toy-car (ToyADMOS)
- Toy-conveyor (ToyADMOS)
- Valve (MIMII Dataset)
- Pump (MIMII Dataset)
- Fan (MIMII Dataset)
- Slide rail (MIMII Dataset)

The ToyADMOS audios are recorded with four microphones and MIMII audios are recorded with eight microphones. All audios have been downsampled to 16kHz.
There are three datasets available.

- Development Dataset containing 1000 normal sounds of each machine for training and 100-200 normal and anomalous sounds of each machine for testing. These sounds will not be used for training. They are just there for checking performance. These are saved at,
  `C:\Users\mpcuda\Desktop\MT_Project_WS\Anomalous_Detection\2020_dataset\Development`
- Evaluation Dataset containing unlabelled 400 audios of each machine for testing.
  `C:\Users\mpcuda\Desktop\MT_Project_WS\Anomalous_Detection\2020_dataset\Evaluation`
- Additional Training Dataset 1000 normal samples of each machine for training. We have copied these audios in the `Evaluation` folder as training audio.

6.1.2 2021_Dataset

This dataset contains one more type of machine, i.e., Gearbox. This dataset is approximately the same one with one addition, i.e., the source domain and target domain. The source domain is the original condition which has enough training clips, and the target domain is the shifted condition where only a few audio clips. The conditions of two conditions differ in terms of operating speed, machine load, viscosity, heating temperature, environmental noise, SNR. This dataset consists of three sections,

- Development Dataset containing around 1,000 clips of normal sounds in a source domain for training, only three clips of normal sounds in a target domain for training, around 100 clips each of normal and anomalous sounds in the source domain for the test, and around 100 clips each of normal and anomalous sounds in the target domain for the test. These are saved at,

```
C:\Users\mpcuda\Desktop\MT_Project_WS\Anomalous_Detection\2021_dataset\Development
```

- Additional Training Dataset containing around 1,000 clips of normal sounds in a source domain for training and only three clips of normal sounds in a target domain for training. These are saved at,

```
C:\Users\mpcuda\Desktop\MT_Project_WS\Anomalous_Detection\2021_dataset\Evaluation
```

- Evaluation Dataset contains test clips in the source domain and test clips in the target domain, none have a condition label (i.e., normal or anomaly).

6.2 Environment Setup

First, create the conda environment. Go to Anaconda Navigator and open Cmd.exe Prompt.

Type,

```
conda create -n ad python=3.6.5
```

Install Tensorflow, Keras, LibROSA, IPython, NumPy, Pandas, Matplotlib, and SciKit Learn in your environment. Following are the respective commands to install these libraries,

```
conda install -c anaconda tensorflow-gpu
conda install -c conda-forge keras
conda install -c conda-forge Librosa
conda install numpy
conda install -c conda-forge matplotlib
conda install -c conda-forge scikit-learn
```

Environment “ad” has the following versions.
Tensorflow (1.15)
Keras (2.3.1)
Librosa (0.6.1)
Numpy (1.18)
Matplotlib (3.0.3)
Scikit-learn (0.22.2)
Audioread (2.1.5)

6.3 Approach

In order to develop an anomalous sound detector, we have to input a large amount of data. Moreover, to calculate the anomalous scores, we must develop a score calculator with certain value parameters and then observe if the target value exceeds this reference value. If so, then it is anomalous. However, it is a difficult task as we have only normal sounds for the training.

So, for 2020_dataset we have Autoencoder-based anomaly detector, and for 2021_dataset we have used Autoencoder plus Mobile Net V2. The anomaly score is calculated as the reconstruction error of the observed sound. The model is trained to minimize the reconstruction error of normal training data to obtain small errors. It is based on one assumption the model cannot reconstruct sounds that are not used in training.

First, we extract the log-mel spectrogram of input, number of mel filter = 64, time frames =5, n_fft = 1024, hop_length =512. These values can be altered in any common.py file. For example, for 2020_dataset,

```
C:\Users\mpcuda\Desktop\MT_Project_WS\Anomalous_Detection\dcase2020_task2_baseline-master\common.py
```

6.4 Training

Auto Encoder [31] consists of one fully connected neural networks, three hidden full connected layers, and one output fully connected layer. Each hidden layer has 128 units, and the dimension of the encoder output is 8. The activation function "ReLU" is used except for the output layer. We have set the no. of epochs to 100 and the batch size of 512. The ADAM optimizer is used, and we fix the learning rate as 0.001. We train specialized AEs for each machine Type/ID using only the normal training samples of each Machine Type/ID

MobileNetV2 [32] contains the initial fully convolution layer with 32 filters, followed by 19 residual bottleneck layers. The input shape is 64*128 images, and each image is triplicated into each color channel. The output function is the softmax for each of the three sections of audio. No. of epochs are 20, batch size 32, and Adam optimizer with a learning rate of 0.00001.

The training can be done by running any train.py file. For example, to train on development dataset of the 2020_dataset,

```
Cd
C:\Users\mpcuda\Desktop\MT_Project_WS\Anomalous_Detection\dcase2020_task2_basel
ine-master

python 00_train.py -d
```

And for the training of evaluation or additional training data,

```
python 00_train.py -e
```

The model is then saved in the model folder.

6.5 Testing

For testing, we have been provided the evaluation dataset, so the area under the receiver operating characteristic (ROC) curve (AUC) and the partial-AUC (pAUC) is calculated. The pAUC is an AUC calculated from a portion of the ROC curve over a prespecified range of interest. The overall ranking was determined as the average ranking of all Machine Types/ID.

The testing is done by running the test.py file. For example, to test on development of 2020_dataset,

```
Cd
C:\Users\mpcuda\Desktop\MT_Project_WS\Anomalous_Detection\dcase2020_task2_basel
ine-master

python 01_test.py -d
```

And for testing of evaluation or additional training data,

```
python 01_test.py -e
```

The results are then saved in results.csv in result folder.

7 References

1. K. J. Piczak, "Environmental sound classification with convolutional neural networks," 2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP), 2015, pp. 1-6, doi: 10.1109/MLSP.2015.7324337.
2. "Machine learning in acoustics: Theory and applications ", The Journal of the Acoustical Society of America 146, 3590 (2019); https://doi.org/10.1121/1.5133944 (Accessed on 05-12-2020). Michael J. Bianco, Peter Gerstoft, James Traer, Emma Ozanich, Marie A. Roch, Sharon Gannot, and Charles-Alban Deledalle.
3. https://www.kaggle.com/ (Accessed on 11-11-2020)
4. https://archive.ics.uci.edu/ml/index.php (Accessed on 02-06-2021)
5. https://towardsdatascience.com/installing-tensorflow-with-cuda-cudnn-and-gpu-support-on-windows-10-60693e46e781 (Accessed on 8-01-2021)
6. https://docs.anaconda.com/anaconda/install/windows/ (Accessed on 24-10-2020)
7. https://www.idmt.fraunhofer.de/en/institute/projects-products/projects/StadtLaerm.html (Accessed on 14-11-2020)
8. https://www.idmt.fraunhofer.de/en.html (Accessed on 02-06-2021)
9. https://www.imms.de/institut.html (Accessed on 02-06-2021)
10. https://john-software.de/ (Accessed on 02-06-2021)
11. https://www.bischoff-elektronik.de/index.php/unternehmen.html (Accessed on 02-06-2021)
12. https://urbansounddataset.weebly.com/urbansound8k.html (Accessed on 12-12-2020)
13. https://www.kaggle.com/mmoreaux/environmental-sound-classification-50 (Accessed on 13-02-2021)
14. https://www.tensorflow.org/ (Accessed on 02-06-2021)
15. https://keras.io/ (Accessed on 02-06-2021)
16. https://librosa.org/ (Accessed on 02-06-2021)
17. https://ipython.org/ (Accessed on 02-06-2021)
18. https://numpy.org/devdocs/index.html# (Accessed on 02-06-2021)
19. https://pandas.pydata.org/ (Accessed on 02-06-2021)
20. https://matplotlib.org/ (Accessed on 02-06-2021)
21. https://scikit-learn.org/stable/ (Accessed on 02-06-2021)
22. "Comparison of Time-Frequency Representations for Environmental Sound Classification using Convolutional Neural Networks", Muhammad Huzaifa, 22-06-2017, https://arxiv.org/pdf/1706.07156.pdf (Accessed on 15-12-2020)
23. https://medium.com/@keur.plkar/audio-data-augmentation-in-python-a91600613e47 (Accessed on 23-12-2020)
24. http://dcase.community/challenge2018/task-acoustic-scene-classification#subtask-a (Accessed on 13-01-2021)
25. https://zenodo.org/record/2639058#.YLdiY6gzaUk (Accessed on 04-03-2021)
26. https://www.dacs-audio.com/product/soundman-okm-ii/ (Accessed on 02-06-2021)
27. https://zoomcorp.com/en/jp/handheld-video-recorders/field-recorders/f8/ (Accessed on 02-06-2021)
28. "Automatic Spatial Audio Scene Classification in Binaural Recordings of Music". Sławomir K. Zielinski and Hyunkook Lee, Faculty of Computer Science, Białystok University of Technology, 15-351 Białystok, Poland; s.zielinski@pb.edu.pl, Applied Psychoacoustics Laboratory (APL), University of Huddersfield, Huddersfield HD1 3DH, UK. Correspondence: h.lee@hud.ac.uk; Tel.: +44-1484-471893. Binaural audio corpus, database with extracted features, and machine-learning code is available at: 10.5281/zenodo.2639058.
29. http://dcase.community/challenge2020/task-unsupervised-detection-of-anomalous-sounds (Accessed on 13-01-2021)
30. http://dcase.community/challenge2021/task-unsupervised-detection-of-anomalous-sounds (Accessed on 21-03-2021)
31. https://www.jeremyjordan.me/autoencoders/ (Accessed on 02-02-2021)

32. “MobileNetV2: Inverted Residuals and Linear Bottlenecks”, Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen. {sandler, howarda, menglong, azhmogin, lcchen}@google.com. 21-03-2019. https://arxiv.org/pdf/1801.04381.pdf. (Accessed on 09-04-2021)