Turkish Online Journal of Qualitative Inquiry (TOJQI)

Volume 12, Issue 7, July 2021: 7206 - 7237

Research Article

Application of Deep Learning in Urban Sounds Classification

Simriti Koul[a]

[a] Vellore Institute of Technology (VIT University)

Abstract

There is a large area of research for automatic sound classification with countless real world applications. Even though there is a large body of research in related audio fields such as speech and music, work on the classification of environmental sounds is comparatively scarce. Hence, when we observe the recent advancements in the field of image classification where convolutional neural networks are used to classify images with high accuracy and at scale, we come across questions on applicability of these techniques in other domains, such as sound classification, where discrete sounds happen over time. This paper is the analysis of the project that tests audio samples and classifies them accordingly using Deep Learning techniques, namely, Multi-Level Perceptron (MLP) and modified Convolutional Neural Networks (CNN).

Keywords: Multi-Level Perceptron (MLP), modified Convolutional Neural Networks (CNN)

Introduction

Sounds surround us. Whether directly or indirectly, we are always in contact with audio data. The sounds describe the context of our day-to-day operations, based on the conversations we have as we interact with people, the music we listen to, and all the various environmental sounds that we hear on daily basis such as children playing, street music, engine drilling and other audio data, either consciously or subconsciously, giving us information about the environment around us.

When we input sound in this program, it will automatically classify it into specific categories using Deep Learning techniques: Multi-Level Perceptron (MLP) and modified Convolutional Neural Networks (CNN). Multi-layer perceptron's (MLP) are classed as a type of Deep Neural Network as they are composed of more than one layer of perceptrons and use non-linear activation which distinguish them from linear perceptrons. Their architecture consists of an input layer, an output layer that ultimately make a prediction about the input, and inbetween the two layers there is an arbitrary number of hidden layers. Whereas, Convolutional Neural Networks (CNNs) build upon the architecture of MLPs but with a number of important changes. Firstly, the layers are organized into three dimensions, width, height and depth. Secondly, the nodes in one layer do not necessarily connect to all nodes in the subsequent layer, but often just a sub region of it.

The objective of this project will be to use Deep Learning techniques to classify urban sounds. For this project, we will use data set called Urbansound8K. It contains 8732 sound excerpts of urban sounds from 10 classes. Due to this project, an automatic classification of urban sounds can be made just by testing the audio clip via this program. Researchers don't have to guess on what kind of sound they are hearing. They will get a specific result which will include its frequency, time and graph. The main goal of this capstone project is to apply Deep Learning techniques to the classification of environmental sounds, specifically focusing on the identification of particular urban sounds. There is a plethora of real world applications for this research, such as:

• Content-based multimedia indexing and retrieval

• Assisting deaf individuals in their daily activities

• Smart home use cases such as 360-degree safety and security capabilities

• Automotive where recognizing sounds both inside and outside of the car can improve safety

• Industrial uses such as predictive maintenance

The assessment measure for this issue will be ‘Classification Accuracy’ which is defined as the percentage of accurate predictions.

Accuracy = correct classifications / number of classifications

Provided that the dataset would be fairly symmetrical (as we will see in the next section) and that this is a multi-class classifier with target data classes that are usually uniform in size, Classification Accuracy was deemed to be the best choice metric.

Other metrics, such as Precision and Recall (or the F1 score when combined), were ruled out because they are better suited to classification tasks with a small target class in an unbalanced data set.

II. Analysis

1. Data Exploration and Visualization

A. UrbanSound Datatset

For this project we will use a dataset called Urbansound8K. The dataset contains 8732 sound excerpts (<=4s) of urban sounds from 10 classes, which are:

• Air Conditioner

• Car Horn

• Children Playing

• Dog bark

• Drilling

• Engine Idling

• Gun Shot

• Jackhammer

• Siren

• Street Music

The accompanying metadata contains a unique ID for each sound excerpt along with it’s given class name.

A sample of this dataset is included with the accompanying git repo and the full dataset can be downloaded from Urbansounds8K official website.

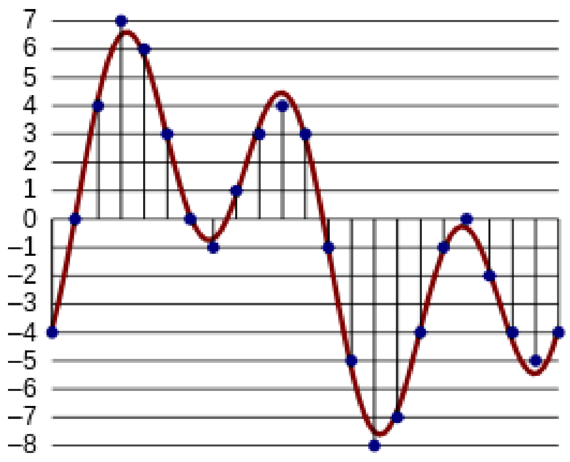
B. Audio sample file data overview

These sound excerpts are digital audio files in .wav format.

Sound waves are digitised by sampling them at discrete intervals known as the sampling rate (typically 44.1kHz for CD quality audio meaning samples are taken 44,100 times per second).

Each sample is the amplitude of the wave at a particular time interval, where the bit depth determines how detailed the sample will be also known as the dynamic range of the signal (typically 16bit which means a sample can range from 65,536 amplitude values).

This can be represented with the following image:



Therefore, the data we will be analysing for each sound excerpts is essentially a one dimensional array or vector of amplitude values.

C. Analyzing audio data

For audio analysis, we will be using the following libraries:

• IPython.display.Audio

This allows us to play audio directly in the Jupyter Notebook.

• Librosa

librosa is a Python package for music and audio processing by Brian McFee and will allow us to load audio in our notebook as a numpy array for analysis and manipulation.

You may need to install librosa using pip as follows:

pip install librosa

D. Auditory inspection

WewilluseIPython.display.Audiotoplaytheaudiofilessowecaninspectaurally.

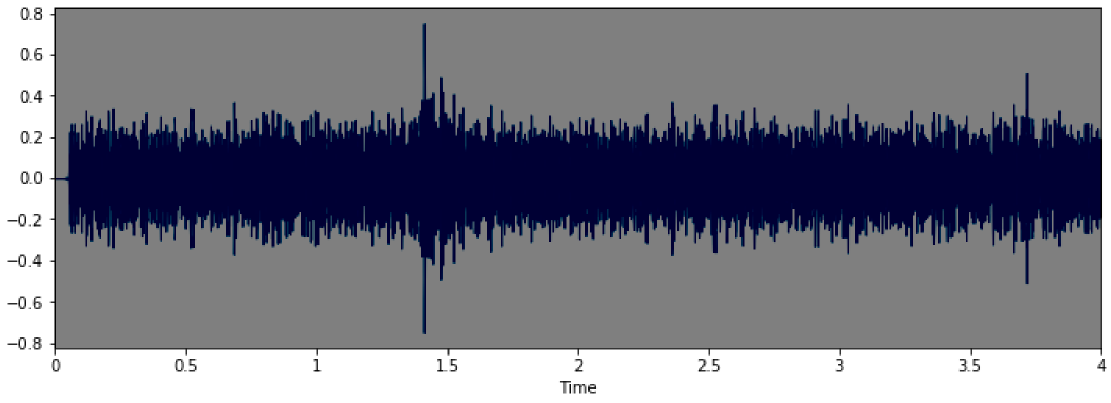
importIPython.displayasipd

ipd.Audio('../UrbanSoundDatasetsample/audio/100032-3-0-0.wav')

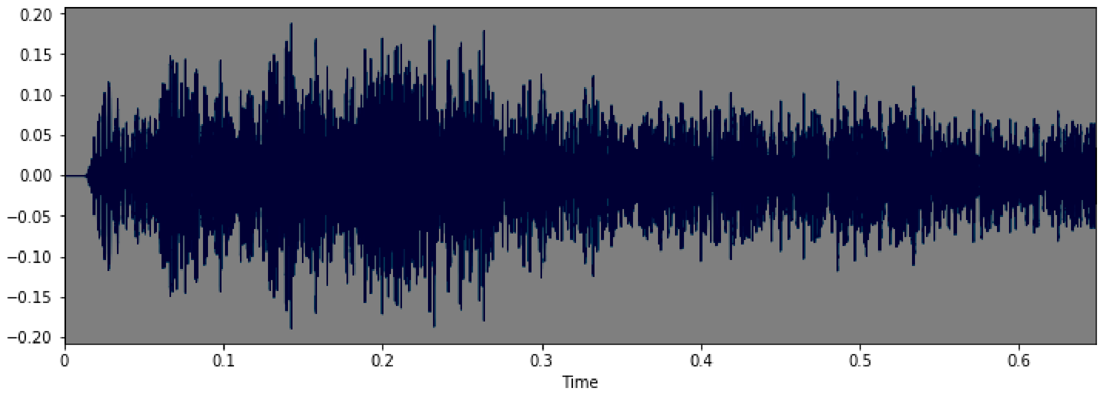
E. Visual Inspection

We will load a sample from each class and visually inspect the data for any patterns.We willuse librosa to load the audio file into an array then librosa.display and matplotlib to display thewaveform.

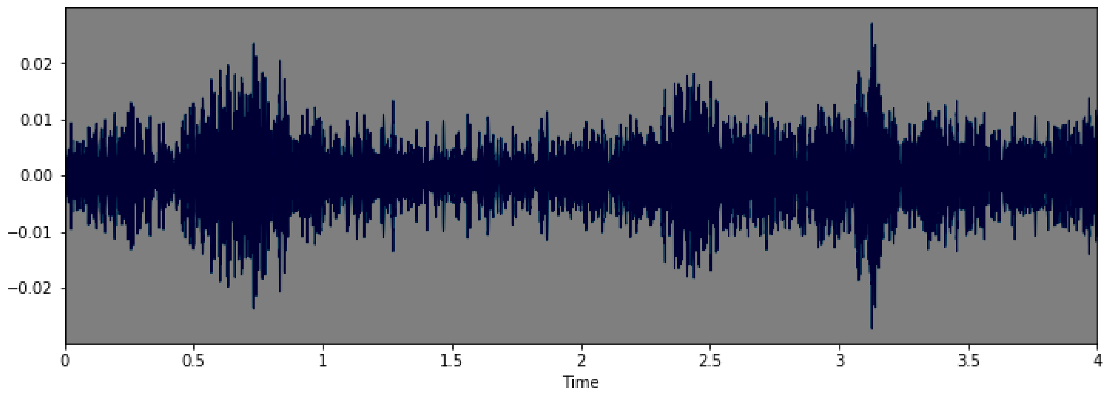
```
# Load imports

importIPython.displayasipdimportlibrosa

importlibrosa.display

importmatplotlib.pyplotasplt

#Class:AirConditioner

filename='../UrbanSoundDatasetsample/audio/100852-0-0-0.wav' plt.figure(figsize=(12,4))

data,sample_rate=librosa.load(filename)

_=librosa.display.waveplot(data,sr=sample_rate)ipd.Audio(filename)
```



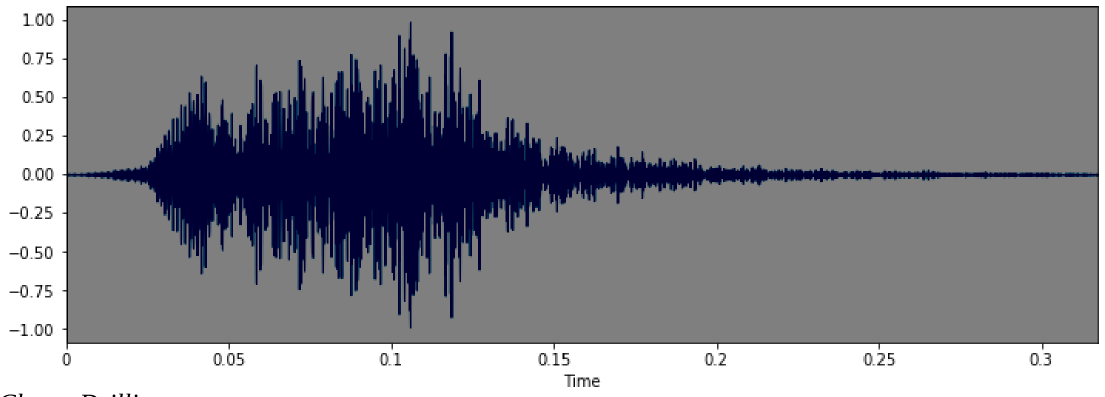
#Class: Car horn



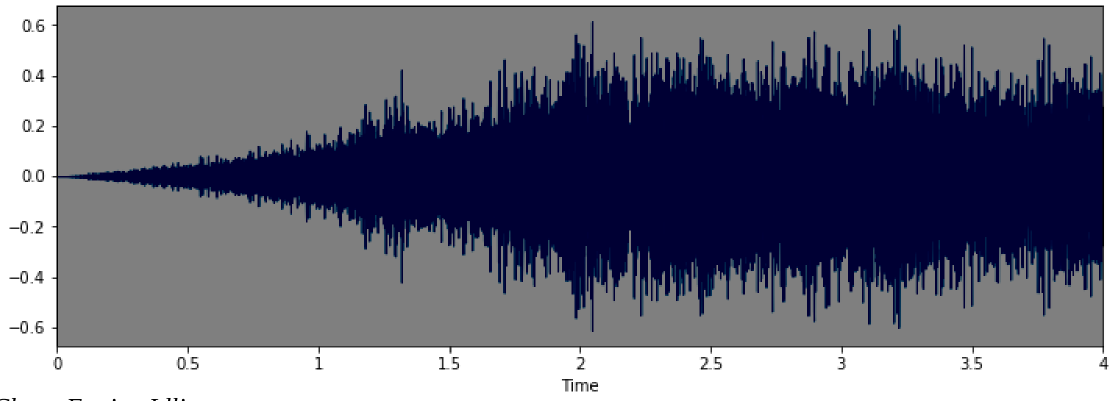
#Class: Children playing



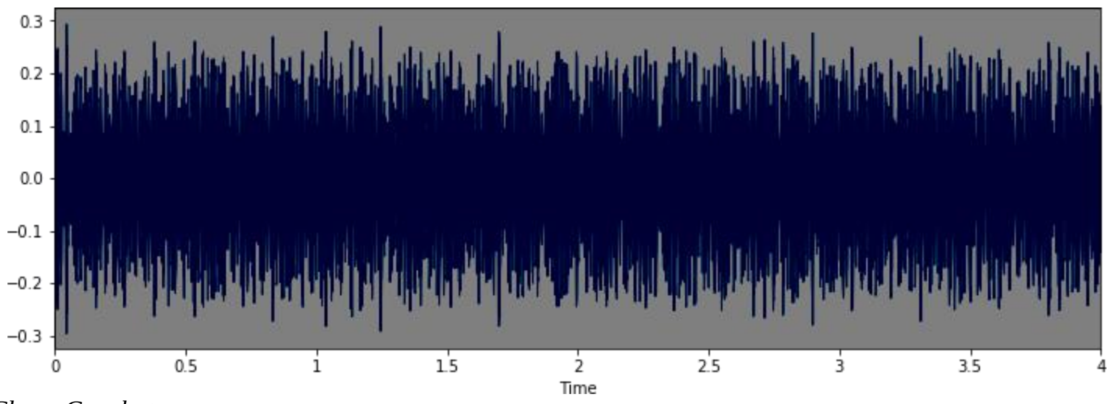
# Class:Dog bark



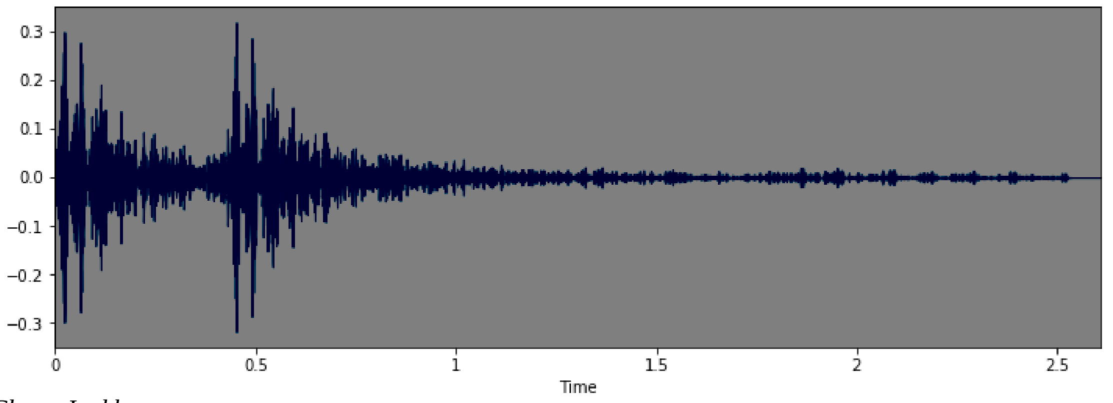
# Class: Drilling



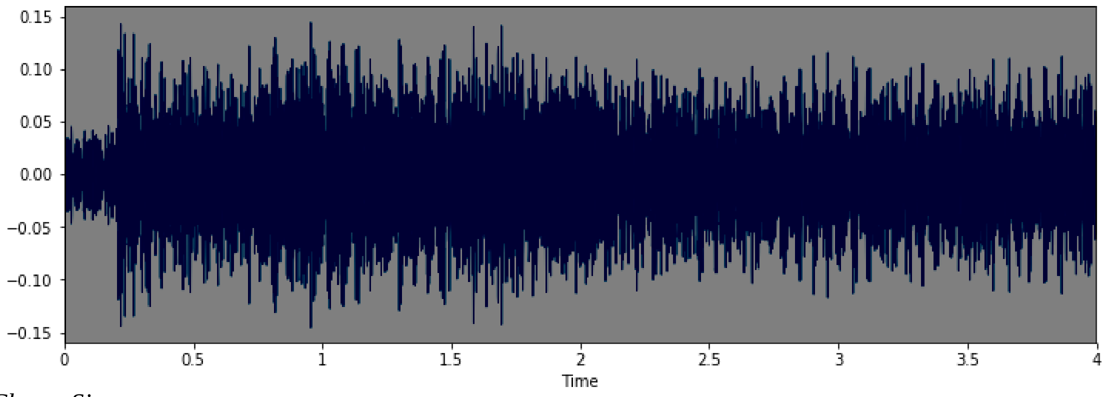
# Class: Engine Idling



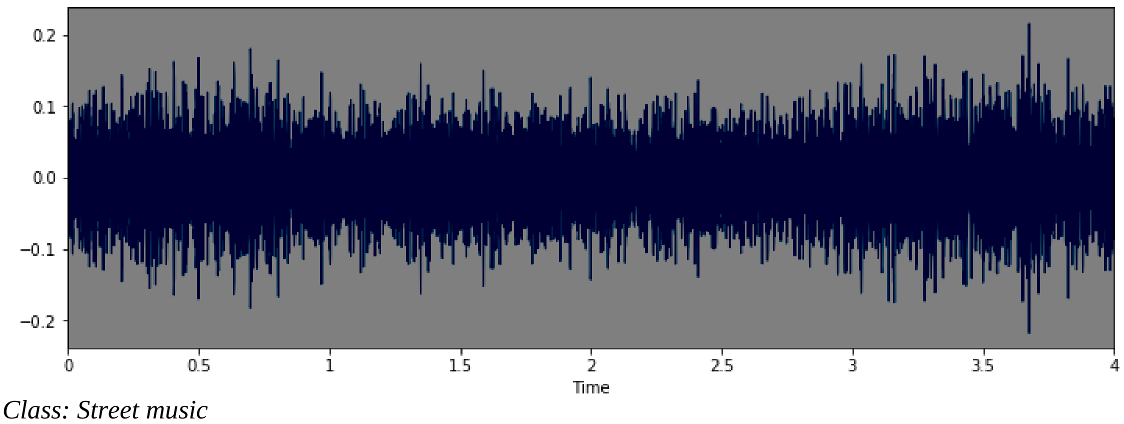
# Class: Gunshot



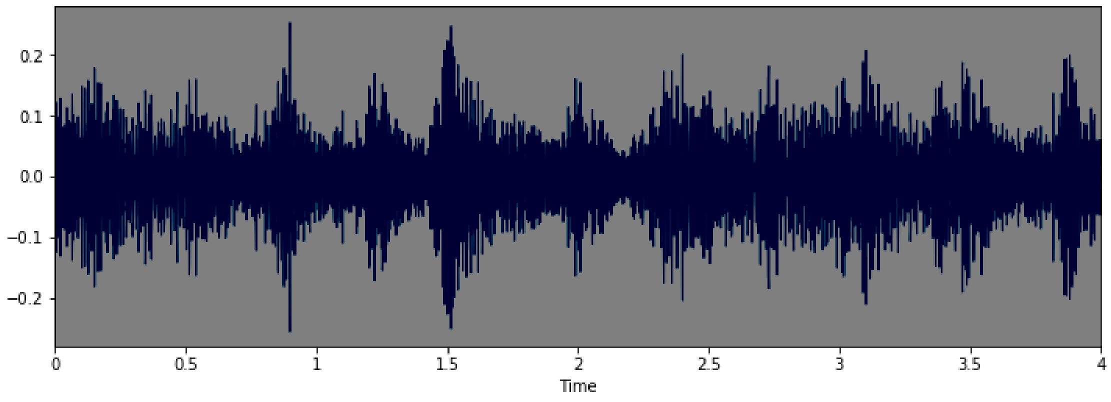
# Class: Jackhammer



# Class: Siren



# Class: Street music



• Observations

From a visual inspection we can see that it is tricky to visualise the difference between some of the classes.

Particularly, the waveforms for repetitive sounds for air conditioner, drilling, engine idling and jackhammer are similar in shape.

Likewise the peak in the dog barking sample is similar in shape to the gun shot sample (albeit the samples differ in that there are two peaks for two gunshots compared to the one peak for one dog bark). Also, the car horn is similar too. There are also similarities between the children playingand street music.

The human ear can naturally detect the difference between the harmonics, it will be interesting to see how well a deep learning model will be able to extract the necessary features to distinguish between these classes.

However, it is easy to differentiate from the waveform shape, the difference between certain classes such as dog barking and jackhammer.

F. Dataset Metadata

HerewewillloadtheUrbanSoundmetadata.csvfileintoaPandadataframe.

importpandasaspd

metadata=pd.read_csv('../UrbanSoundDatasetsample/metadata/UrbanSound8K.csv') metadata.head()

• Observations

```
slice_file_name      fsIDstart               endsalience      fold   classID      class_name
0    100032-3-0-0.wav100032          0.0      0.317551       1      53           dog_bark
1100263-2-0-117.wav        100263    58.5    62.500000       1      52           children_playing
2100263-2-0-121.wav        100263    60.5    64.500000       1      52           children_playing
3100263-2-0-126.wav        100263    63.0    67.000000       1      52           children_playing
4100263-2-0-137.wav        100263    68.5    72.500000       1      52           children_playing
```

G. Class Distributions

print(metadata.class_name.value_counts())

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

• Observations

Here we can see the Class labels are unbalanced. Although 7 out of the 10 classes all have exactly 1000 samples, and siren is not far off with 929, the remaining two (car_horn, gun_shot) have significantly less samples at 43% and 37% respectively.

This will be a concern and something we may need to address later on.

H. Audio sample file properties

Nextwewilliteratethrougheachoftheaudiosamplefilesandextract,numberofaudiochannels,samplerateandbit-depth.

*#Loadvariousimports*import pandas as pdimportos

importlibrosa

importlibrosa.display

fromhelpers.wavfilehelperimportWavFileHelperwavfilehelper=WavFileHelper()

audiodata=[]

forindex,rowinmetadata.iterrows():

file_name=os.path.join(os.path.abspath('/Volumes/Untitled/ML_Data/UrbanSound/UrbanSound8K data=wavfilehelper.read_file_properties(file_name)

audiodata.append(data)

#ConvertintoaPandadataframe

audiodf=pd.DataFrame(audiodata,columns=['num_channels','sample_rate','bit_depth'])

I. Audio Channels

Mostofthesampleshavetwoaudiochannels(meaningstereo)withafewwithjusttheonechannel(mono).

The easiest option here to make them uniform will be to merge the two channels in the stereosamplesintoonebyaveragingthevaluesofthetwochannels.

#numofchannels

print(audiodf.num_channels.value_counts(normalize=True))

```
2    0.915369

1    0.084631
```

J. Sample Rate

ThereisawiderangeofSampleratesthathavebeenusedacrossallthesampleswhichisaconcern(rangingfrom96kto8k).

Thislikelymeansthatwewillhavetoapplyasample-rateconversiontechnique(eitherup-conversion or down-conversion) so we can see an agnostic representation of their waveform whichwillallowustodoafaircomparison.

#sample rates

print(audiodf.sample_rate.value_counts(normalize=True))

```
44100     0.614979
48000     0.286532
96000     0.069858
24000     0.009391
16000     0.005153
22050     0.005039
11025     0.004466
192000    0.001947
8000      0.001374
11024     0.000802
32000     0.000458
```

K. Bit Depth

There is also a wide range of bit-depths. It’s likely that we may need to normalise them by taking the maximum and minimum amplitude values for a given bit-depth.

```
# bit depth

print(audiodf.bit_depth.value_counts(normalize=True))
```

```
16    0.659414
24    0.315277
32    0.019354
8     0.004924
4     0.001031
```

L. Other properties

We may also need to consider normalising the volume levels (wave amplitude value) if this is seen to vary greatly, by either looking at the peak volume or the RMS volume.

2. Algorithms & Techniques

The proposed solution to this problem is to apply Deep Learning techniques that have proved to be highly successful in the field of image classification.

First we will extract Mel-Frequency Cepstral Coefficients (MFCC) [2] from the the audio samples on a per frame basis with a window size of a few milliseconds. The MFCC summarises the frequency distribution across the window size, so it is possible to analyse both the frequency and time characteristics of the sound. These audio representations will allow us to identify features for classification.

The next step will be to train a Deep Neural Network with these data sets and make predictions. We will begin by using a simple neural network architecture, such as Multi-Layer Perceptron before experimenting with more complex architectures such as Convolutional Neural Networks.

Multi-layer perceptron’s (MLP) are classed as a type of Deep Neural Network as they are composed of more than one layer of perceptrons and use non-linear activation which distinguish them from linear perceptrons. Their architecture consists of an input layer, an output layer that ultimately make a prediction about the input, and in-between the two layers there is an arbitrary number of hidden layers.

These hidden layers have no direct connection with the outside world and perform the model computations. The network is fed a labelled dataset (this being a form of supervised learning) of input-output pairs and is then trained to learn a correlation between those inputs and outputs. The training process involves adjusting the weights and biases within the perceptrons in the hidden layers in order to minimise the error.

The algorithm for training an MLP is known as Backpropagation. Starting with all weights in the network being randomly assigned, the inputs do a forward pass through the network and the decision of the output layer is measured against the ground truth of the labels you want to predict. Then the weights and biases are backpropagated back though the network where an optimisation method, typically Stochastic Gradient descent is used to adjust the weights so they will move one step closer to the error minimum on the next pass. The training phase will keep on performing this cycle on the network until it the error can go no lower which is known as convergence.

Convolutional Neural Networks (CNNs) build upon the architecture of MLPs but with a number of important changes. Firstly, the layers are organised into three dimensions, width, height and depth. Secondly, the nodes in one layer do not necessarily connect to all nodes in the subsequent layer, but often just a sub region of it.

This allows the CNN to perform two important stages. The first being the feature extraction phase. Here a filter window slides over the input and extracts a sum of the convolution at each location which is then stored in the feature map. A pooling process is often included between CNN layers where typically the max value in each window is taken which decreases the feature map size but retains the significant data. This is important as it reduces the dimensionality of the network meaning it reduces both the training time and likelihood of overfitting. Then lastly we have the classification phase. This is where the 3D data within the network is flattened into a 1D vector to be output.

For the reasons discussed, both MLPs and CNN's typically make good classifiers, where CNN's in particular perform very well with image classification tasks due to their feature extraction and classification parts. I believe that this will be very effective at finding patterns within the MFCC's much like they are effective at finding patterns within images.

We will use the evaluation metrics described in earlier sections to compare the performance of these solutions against the benchmark models in the next section.

3. Benchmark Model

For the benchmark model, we will use the algorithms outlined in the paper "A Dataset and Taxonomy for Urban Sound Research" (Salamon, 2014) [3]. The paper describes five different algorithms with the following accuracies for a audio slice maximum duration of 4 seconds using the same UrbanSound dataset.

Algorithm	**ClassificationAccuracy**
SVM_rbf	68%
RandomForest500	66%
IBk5	55%
J48	48%
ZeroR	10%

III. Methodology

1. Data Preprocessing & data splitting

Following on from the previous section, we identified the following audio properties that need preprocessing to ensure consistency across the whole dataset:

- Audio Channels
- Sample rate
- Bit-depth

We will continue to use Librosa which will be useful for the pre-processing and feature extraction.

A. Audio properties that require normalizing

FormuchofthepreprocessingwewillbeabletouseLibrosa'sload()function.

We will compare the outputs from Librosa against the default outputs of scipy's wavfile libraryusingachosenfilefromthedataset.

B. Preprocessing stage

• Sample rate conversion

By default, Librosa’s load function converts the sampling rate to 22.05KHzwhichwecanuseasourcomparisonlevel.

importlibrosa

fromscipy.ioimportwavfileaswavimportnumpyasnp

filename='../UrbanSoundDatasetsample/audio/100852-0-0-0.wav'

librosa_audio,librosa_sample_rate=librosa.load(filename)scipy_sample_rate,scipy_audio=wav.read(filename)

print('Originalsamplerate:',scipy_sample_rate)

print('Librosasamplerate:',librosa_sample_rate)

Originalsamplerate:44100

Librosasamplerate:22050

• Bit Depth

Librosa’sloadfunctionwillalsonormalisethedatasoit’svaluesrangebetween 1and1.Thisremovesthecomplicationofthedatasethavingawiderangeofbit-depths.

print('Originalaudiofilemin~maxrange:',np.min(scipy_audio),'to',np.max(scipy_audio)) print('Librosaaudiofilemin~maxrange:',np.min(librosa_audio),'to',np.max(librosa_audio))

Originalaudiofilemin~maxrange:-23628to27507

Librosaaudiofilemin~maxrange:-0.50266445to0.74983937

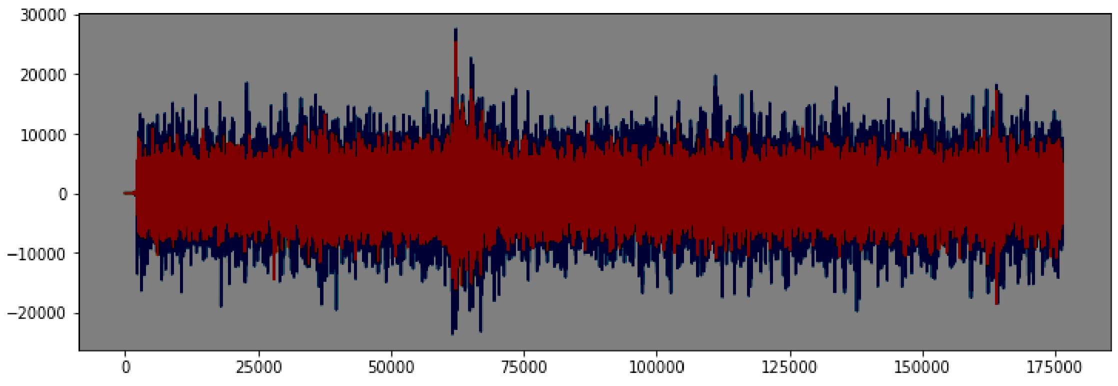
• Merge audio channels

Librosawillalsoconvertthesignaltomono,meaningthenumberofchannelswillalwaysbe1.

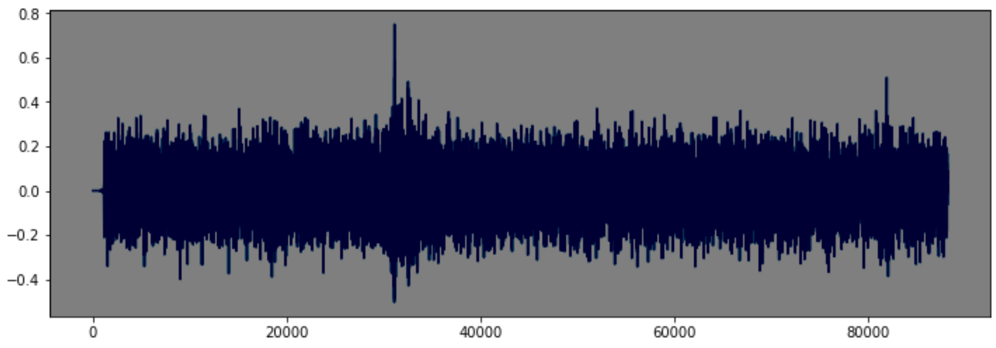
importmatplotlib.pyplotasplt

#Originalaudiowith2channels

plt.figure(figsize=(12, 4))plt.plot(scipy_audio)



*#Librosaaudiowithchannelsmerged*plt.figure(figsize=(12, 4))plt.plot(librosa_audio)



• Other properties

At this stage it is not yet clear whether other factors may also need to be taken into account, such as sample duration length and volume levels.

We will proceed as is for the meantime and come back to address these later if it’s perceived to be effecting the validity of our target metrics.

C. Extract Features

As outlined in the proposal, we will extract Mel-Frequency Cepstral Coefficients (MFCC) from the the audio samples.

The MFCC summarises the frequency distribution across the window size, so it is possible to analyse both the frequency and time characteristics of the sound. These audio representations will allow us to identify features for classification.

• Extracting a MFCC

ForthiswewilluseLibrosa’smfcc()functionwhichgeneratesanMFCCfromtimeseriesaudiodata.

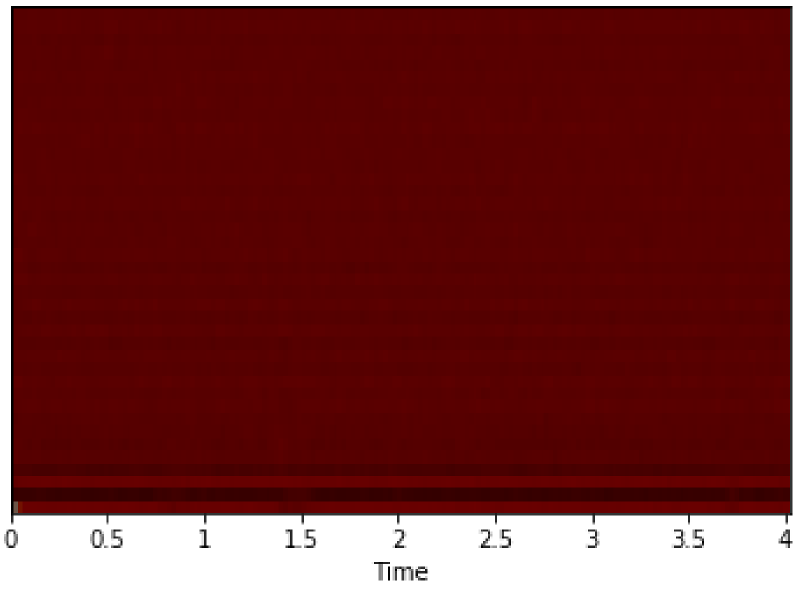
mfccs=librosa.feature.mfcc(y=librosa_audio,sr=librosa_sample_rate,n_mfcc=40)print(mfccs.shape)

(40,173)

Thisshowslibrosacalculatedaseriesof40MFCCsover173frames.

importlibrosa.display

librosa.display.specshow(mfccs,sr=librosa_sample_rate,x_axis='time')



• Extracting MFCC’s for every file

WewillnowextractanMFCCforeachaudiofileinthedatasetandstoreitinaPandaDataframealongwithit’sclassificationlabel.

```
defextract_features(file_name):
try:
audio,sample_rate=librosa.load(file_name,res_type='kaiser_fast')
mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=40)
mfccsscaled=np.mean(mfccs.T,axis=0)
exceptExceptionase:
print("Errorencounteredwhileparsingfile:",file)
returnNone
returnmfccsscaled
#Loadvariousimports
import pandas as pd
importos
importlibrosa
#SetthepathtothefullUrbanSounddataset
fulldatasetpath='/Volumes/Untitled/ML_Data/UrbanSound/UrbanSound8K/audio/'
metadata=pd.read_csv('../UrbanSoundDatasetsample/metadata/UrbanSound8K.csv')
features=[]
#Iteratethrougheachsoundfileandextractthefeatures
forindex,rowinmetadata.iterrows():
```

file_name=os.path.join(os.path.abspath(fulldatasetpath),'fold'+str(row["fold"])+'/',str(ro

class_label = row["class_name"]data=extract_features(file_name)

features.append([data,class_label])

#ConvertintoaPandadataframe

featuresdf=pd.DataFrame(features,columns=['feature','class_label'])

print('Finishedfeatureextractionfrom',len(featuresdf),'files')

Finishedfeatureextractionfrom8732files

D. Convert the data and labels

Wewillusesklearn.preprocessing.LabelEncodertoencodethecategoricaltextdataintomodel-understandablenumericaldata.

fromsklearn.preprocessingimportLabelEncoderfromkeras.utilsimportto_categorical

#Convertfeaturesandcorrespondingclassificationlabelsintonumpyarrays

X=np.array(featuresdf.feature.tolist())

y=np.array(featuresdf.class_label.tolist())

# Encode the classification labels

le=LabelEncoder()

yy=to_categorical(le.fit_transform(y))

E. Split the dataset

Herewewillusesklearn.model_selection.train_test_splittosplitthedatasetintotrainingandtestingsets.Thetestingsetsizewillbe20%andwewillsetarandomstate.

#splitthedataset

fromsklearn.model_selectionimporttrain_test_split

x_train,x_test,y_train,y_test=train_test_split(X,yy,test_size=0.2,random_state=42)

2. Implementation

A. Initial model architecture – MLP

We will start with constructing a Multilayer Perceptron (MLP) Neural Network using Keras and a Tensorflow backend.

Starting with a sequential model so we can build the model layer by layer.

We will begin with a simple model architecture, consisting of three layers, an input layer, a hidden layer and an output layer. All three layers will be of the dense layer type which is a standard layer type that is used in many cases for neural networks.

The first layer will receive the input shape. As each sample contains 40 MFCCs (or columns) we have a shape of (1x40) this means we will start with an input shape of 40.

The first two layers will have 256 nodes. The activation function we will be using for our first 2 layers is the ReLU, or Rectified Linear Activation. This activation function has been proven to work well in neural networks.

We will also apply a Dropout value of 50% on our first two layers. This will randomly exclude nodes from each update cycle which in turn results in a network that is capable of better generali sation and is less likely to overfit the training data.

Our output layer will have 10 nodes (num_labels) which matches the number of possible classifi- cations. The activation is for our output layer is softmax. Softmax makes the output sum up to 1 so the output can be interpreted as probabilities. The model will then make its prediction based on which option has the highest probability.

```
importnumpyasnp

fromkeras.modelsimportSequential

fromkeras.layersimportDense,Dropout,Activation,Flatten

fromkeras.layersimportConvolution2D,MaxPooling2D

fromkeras.optimizersimportAdam

from keras.utils import np_utils

fromsklearnimportmetrics

num_labels=yy.shape[1]filter_size=2

#Construct model

model=Sequential()

model.add(Dense(256,input_shape=(40,)))model.add(Activation('relu')) model.add(Dropout(0.5))

model.add(Dense(256))model.add(Activation('relu'))

model.add(Dropout(0.5))

model.add(Dense(num_labels))model.add(Activation('softmax'))
```

B. Compiling the model

Forcompilingourmodel,wewillusethefollowingthreeparameters:

- Lossfunction-wewillusecategorical_crossentropy.Thisisthemostcommonchoiceforclassification.Alowerscoreindicates thatthemodelisperformingbetter.
- Metrics-wewillusetheaccuracymetricwhichwillallowustoviewtheaccuracyscoreonthevalidationdatawhenwetrainthemodel.
- Optimizer-herewewilluseadamwhichisagenerallygoodoptimizerformanyusecases.

```
#Compilethemodel

model.compile(loss='categorical_crossentropy',metrics=['accuracy'],optimizer='adam')

#Display model architecture summary

model.summary()
```

```
#Calculatepre-trainingaccuracy

score=model.evaluate(x_test,y_test,verbose=0)accuracy=100*score[1]

print("Pre-trainingaccuracy:%.4f%%"%accuracy)
```

```
_________________________________________________________________
Layer(type)    OutputShape       Param#
=================================================================
  dense_1(Dense)                 (None,      256)           10496
_________________________________________________________________
  activation_1(Activation)       (None,      256)           0
_________________________________________________________________
  dropout_1(Dropout)             (None,      256)           0
_________________________________________________________________
  dense_2(Dense)                 (None,      256)           65792
_________________________________________________________________
  activation_2(Activation)       (None,      256)           0
_________________________________________________________________
  dropout_2(Dropout)             (None,      256)           0
_________________________________________________________________
  dense_3(Dense)                 (None,      10)            2570
_________________________________________________________________
  activation_3(Activation)       (None,      10)            0
=================================================================
Totalparams:78,858

Trainableparams:78,858

Non-trainableparams:0
_________________________________________________________________
Pre-trainingaccuracy:11.5627%
```

C. Training

Herewewilltrainthemodel.

Wewillstartwith100epochswhichisthenumberoftimesthemodelwillcyclethroughthedata.The modelwillimproveoneachcycleuntilitreachesacertainpoint.

Wewillalsostartwithalowbatchsize,ashavingalargebatchsizecanreducethegeneralisationabilityofthemodel.

```
fromkeras.callbacksimportModelCheckpointfromdatetimeimportdatetime

num_epochs=100

num_batch_size=32

checkpointer=ModelCheckpoint(filepath='saved_models/weights.best.basic_mlp.hdf5',

verbose=1,save_best_only=True)
```

start=datetime.now()

model.fit(x_train,y_train,batch_size=num_batch_size,epochs=num_epochs,validation_data=(x_tes

duration = datetime.now() - startprint("Trainingcompletedintime:",duration)

Train on 6985 samples, validate on 1747 samples.

```
Epoch   00097:val_lossdidnotimprovefrom             0.42049

Epoch   98/100

6985/6985[============================ -2s329us/step    -loss:     0.5246   -acc:     0.8241   -val_lo
==]

Epoch00098:val_lossdidnotimprovefromEpoch99/  0.42049
100

6985/6985[============================
==]                                    -2s347us/step    -loss:     0.5346   -acc:     0.8169   -val_lo

Epoch00099:val_lossdidnotimprovefromEpoch100  0.42049
/100

6985/6985[============================
==]                                    -2s351us/step    -loss:     0.5413   -acc:     0.8153   -val_lo

Epoch00100:val_lossdidnotimprovefromTrainingc  0.42049
ompletedintime:0:04:15.582298
```

D. Test the model

Herewewillreviewtheaccuracyofthemodelonboththetrainingandtestdatasets.

*#Evaluatingthemodelonthetrainingandtestingset*score = model.evaluate(x_train, y_train, verbose=0)print("TrainingAccuracy:",score[1])

score=model.evaluate(x_test,y_test,verbose=0)print("TestingAccuracy:",score[1])

TrainingAccuracy:0.9252684323550465

TestingAccuracy:0.8763594734511787

TheinitialTrainingandTestingaccuracyscoresarequitehigh.AsthereisnotagreatdifferencebetweentheTraining andTestscores(~5%)thissuggeststhatthemodelhasnotsufferedfromoverfitting.

E. Predictions

Herewewillbuildamethodwhichwillallowustotestthemodelspredictionsonaspecifiedaudio.wavfile.

import librosa

importnumpyasnp

defextract_feature(file_name):

try:

audio_data,sample_rate=librosa.load(file_name,res_type='kaiser_fast')

mfccs= librosa.feature.mfcc(y=audio_data, sr=sample_rate, n_mfcc=40)

```
mfccsscaled=np.mean(mfccs.T,axis=0)
exceptExceptionase:
print("Errorencounteredwhileparsingfile:",file)
returnNone,None
return np.array([mfccsscaled])
defprint_prediction(file_name):
prediction_feature=extract_feature(file_name)
predicted_vector=model.predict_classes(prediction_feature)predicted_class =
le.inverse_transform(predicted_vector)
print("Thepredictedclassis:",predicted_class[0],'\n')
predicted_proba_vector=model.predict_proba(prediction_feature)
predicted_proba=predicted_proba_vector[0]
foriinrange(len(predicted_proba)):
category=le.inverse_transform(np.array([i]))
print(category[0],"\t\t:",format(predicted_proba[i],'.32f'))
```

F. Validation

Test with sample data Initial sanity check to verify the predictions using a subsection of thesample audio files we explored in the first notebook.We expect the bulk of these to be classifiedcorrectly.

```
#Class:AirConditioner
filename='../UrbanSoundDatasetsample/audio/100852-0-0-0.wav' print_prediction(filename)
Thepredictedclassis:air_conditioner
# Class: Drilling
filename='../UrbanSoundDatasetsample/audio/103199-4-0-0.wav' print_prediction(filename)
Thepredictedclassis:drilling
#Class:Streetmusic
filename='../UrbanSoundDatasetsample/audio/101848-9-0-0.wav' print_prediction(filename)
Thepredictedclassis:street_music
#Class:CarHorn
filename='../UrbanSoundDatasetsample/audio/100648-1-0-0.wav' print_prediction(filename)
Thepredictedclassis:car_horn
```

Observations

From this brief sanity check the model seems to predict well. One error was observed whereby a car horn was incorrectly classified as a dog bark.

We can see from the per class confidence that this was quite a low score (43%). This allows follows our early observation that a dog bark and car horn are similar in spectral shape.

G. Other Audio

Herewewilluseasampleofvariouscopyrightfreesoundsthatwenotpartofeitherourtestortrainingdatatofurthervalid ateourmodel.

filename='../Evaluationaudio/dog_bark_1.wav' print_prediction(filename)

Thepredictedclassis:dog_bark

filename='../Evaluationaudio/drilling_1.wav' print_prediction(filename)

Thepredictedclassis:drilling

filename='../Evaluationaudio/gun_shot_1.wav' print_prediction(filename)

#sampledataweightedtowardsgunshot-peakinthedogbarkingsampleissimmilarin

#shapetothegunshotsample.

Thepredictedclassis:dog_bark

filename='../Evaluationaudio/siren_1.wav' print_prediction(filename)

Thepredictedclassis:siren

ObservationsTheperformanceofourinitialmodelissatisfactoryandhasgeneralisedwell,seemingtopredictwellwh entestedagainstnewaudiodata.

3. Refinement

In our initial attempt, we were able to achieve a Classification Accuracy score of:

- Training data Accuracy: 92.3%
- Testing data Accuracy: 87%

We will now see if we can improve upon that score using a Convolutional Neural Network (CNN).

Feature extraction refinement

In the previous feature extraction stage, the MFCC vectors would vary in size for the different audio files (depending on the samples duration).

However, CNNs require a fixed size for all inputs which means we will have to revisit the feature extraction code that we previously wrote. To overcome this we will zero pad the output vectors to make them all the same size.importnumpyasnpmax_pad_len=174

defextract_features(file_name):

try:

audio,sample_rate=librosa.load(file_name,res_type='kaiser_fast')

mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=40)

pad_width=max_pad_len-mfccs.shape[1]

mfccs=np.pad(mfccs,pad_width=((0,0),(0,pad_width)),mode='constant')

```
exceptExceptionase:
print("Errorencounteredwhileparsingfile:",file_name)
returnNone
returnmfccs
#Loadvariousimports
import pandas as pd
importos
importlibrosa
#SetthepathtothefullUrbanSounddataset
fulldatasetpath='/Volumes/Untitled/ML_Data/UrbanSound/UrbanSound8K/audio/'
metadata=pd.read_csv('../UrbanSoundDatasetsample/metadata/UrbanSound8K.csv')
features=[]
#Iteratethrougheachsoundfileandextractthefeatures
forindex,rowinmetadata.iterrows():
file_name=os.path.join(os.path.abspath(fulldatasetpath),'fold'+str(row["fold"])+'/',str(ro
class_label = row["class_name"]data=extract_features(file_name)
features.append([data,class_label])
#ConvertintoaPandadataframe
featuresdf=pd.DataFrame(features,columns=['feature','class_label'])
print('Finishedfeatureextractionfrom',len(featuresdf),'files')
Finishedfeatureextractionfrom8732files
fromsklearn.preprocessingimportLabelEncoderfromkeras.utilsimportto_categorical
#Convertfeaturesandcorrespondingclassificationlabelsintonumpyarrays
X=np.array(featuresdf.feature.tolist())
y=np.array(featuresdf.class_label.tolist())
# Encode the classification labels
le=LabelEncoder()
yy=to_categorical(le.fit_transform(y))
#splitthedataset
fromsklearn.model_selectionimporttrain_test_split
x_train,x_test,y_train,y_test=train_test_split(X,yy,test_size=0.2,random_state=42)
```

A. Convolutional Neural Network (CNN) model architecture

We will modify our model to be a Convolutional Neural Network (CNN) again using Keras and a Tensorflow backend.

Again we will use a sequential model, starting with a simple model architecture, consisting of four Conv2D convolution layers, with our final output layer being a dense layer.

The convolution layers are designed for feature detection. It works by sliding a filter window over the input and performing a matrix multiplication and storing the result in a feature map. This operation is known as a convolution.

The filter parameter specifies the number of nodes in each layer. Each layer will increase in size from 16, 32, 64 to 128, while the kernel_size parameter specifies the size of the kernel window which in this case is 2 resulting in a 2x2 filter matrix.

The first layer will receive the input shape of (40, 174, 1) where 40 is the number of MFCC's 174 is the number of frames taking padding into account and the 1 signifying that the audio is mono.

The activation function we will be using for our convolutional layers is ReLU which is the same as our previous model. We will use a smaller Dropout value of 20% on our convolutional layers.

Each convolutional layer has an associated pooling layer of MaxPooling2D type with the final convolutional layer having a GlobalAveragePooling2D type. The pooling layer is do reduce the dimensionality of the model (by reducing the parameters and subsquent computation require- ments) which serves to shorten the training time and reduce overfitting. The Max Pooling type takes the maximum size for each window and the Global Average Pooling type takes the average which is suitable for feeding into our dense output layer.

Our output layer will have 10 nodes (num_labels) which matches the number of possible classifi- cations. The activation is for our output layer is softmax. Softmax makes the output sum up to 1 so the output can be interpreted as probabilities. The model will then make its prediction based on which option has the highest probability.

```
importnumpyasnp

fromkeras.modelsimportSequential

fromkeras.layersimportDense,Dropout,Activation,Flatten

fromkeras.layersimportConvolution2D,Conv2D,MaxPooling2D,GlobalAveragePooling2Dfromkeras.optimizersimportAdam

fromkeras.utilsimportnp_utilsfromsklearnimportmetrics

num_rows=40

num_columns=174

num_channels=1

x_train=x_train.reshape(x_train.shape[0],num_rows,num_columns,num_channels)x_test=x_test.reshape(x_test.shape[0],num_rows,num_columns,num_channels)

num_labels=yy.shape[1]filter_size=2

#Construct model

model=Sequential()

model.add(Conv2D(filters=16,kernel_size=2,input_shape=(num_rows,num_columns,num_channels),model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.2))

model.add(Conv2D(filters=32,kernel_size=2,activation='relu')) model.add(MaxPooling2D(pool_size=2))

model.add(Dropout(0.2))

model.add(Conv2D(filters=64,kernel_size=2,activation='relu')) model.add(MaxPooling2D(pool_size=2))

model.add(Dropout(0.2))

model.add(Conv2D(filters=128,kernel_size=2,activation='relu')) model.add(MaxPooling2D(pool_size=2))

model.add(Dropout(0.2))model.add(GlobalAveragePooling2D())

model.add(Dense(num_labels,activation='softmax'))
```

B. Compiling Model

Forcompilingourmodel,wewillusethesamethreeparametersasthepreviousmodel:

```
#Compilethemodel

model.compile(loss='categorical_crossentropy',metrics=['accuracy'],optimizer='adam')

#Display model architecture summary

model.summary()

#Calculatepre-trainingaccuracy

score=model.evaluate(x_test,y_test,verbose=1)accuracy=100*score[1]

print("Pre-trainingaccuracy:%.4f%%"%accuracy)
```

```
Layer(type)    OutputShape       Param#
=================================================================
  conv2d_11(Conv2D)                 (None    3    173,16)        80
                                  ,       9,
  max_pooling2d_11(MaxPooling       (None    1    86,16)         0
                                  ,       9,
  dropout_17(Dropout)               (None    1    86,16)         0
                                  ,       9,
  conv2d_12(Conv2D)                 (None    1    85,32)         2080
                                  ,       8,
  max_pooling2d_12(MaxPooling       (None    9    42,32)         0
                                  ,       ,
  dropout_18(Dropout)               (None    9    42,32)         0
                                  ,       ,
  conv2d_13(Conv2D)                 (None    8    41,64)         8256
                                  ,       ,
```

max_pooling2d_13(MaxPooling	(None, 4, 20,64)	0
dropout_19(Dropout)	(None, 4, 20,64)	0
conv2d_14(Conv2D)	(None, 3, 19,128)	32896
max_pooling2d_14(MaxPooling	(None, 1, 9,128)	0
dropout_20(Dropout)	(None, 1, 9,128)	0
global_average_pooling2d_1	((None, 128)	0
dense_13(Dense)	(None, 10)	1290

```
=================================================================
Totalparams:44,602
Trainableparams:44,602
Non-trainableparams:0
_________________________________________________________________
1747/1747[==============================]-9s5ms/step
Pre-trainingaccuracy:12.0206%
```

C. Training

Here we will train the model.As training a CNN can take a sigificant amount of time, we willstart with a low number of epochs and a low batch size. If we can see from the output that themodelisconverging,wewillincreasebothnumbers.

```
fromkeras.callbacksimportModelCheckpoint
fromdatetimeimportdatetime
#num_epochs=12
#num_batch_size= 128
num_epochs=72
num_batch_size=256
checkpointer=ModelCheckpoint(filepath='saved_models/weights.best.basic_cnn.hdf5',
verbose=1,save_best_only=True)
start=datetime.now()
model.fit(x_train,y_train,batch_size=num_batch_size,epochs=num_epochs,validation_data=(x_tes
duration = datetime.now() - startprint("Trainingcompletedintime:",duration)

Trainon6985samples,validateon1747samples

Epoch00070:val_lossdidnotimprovefrom0.27239Epoch71/72

6985/6985[==============================]-14289s2s/step-loss:0.1203-acc:0.9581-val_l

Epoch00071:val_lossdidnotimprovefrom0.27239Epoch72/72

6985/6985[==============================]-92s13ms/step-loss:0.1147-acc:0.9596-val_lo

Epoch00072:val_lossdidnotimprovefrom0.27239

Trainingcompletedintime:8:57:38.203486
```

D. Test the model

Herewewillreviewtheaccuracyofthemodelonboththetrainingandtestdatasets.

```
#Evaluatingthemodelonthetrainingandtestingset

score = model.evaluate(x_train, y_train, verbose=0)

print("TrainingAccuracy:",score[1])

score=model.evaluate(x_test,y_test,verbose=0)print("TestingAccuracy:",score[1])

TrainingAccuracy:0.9819613457408733

TestingAccuracy:0.9192902116210514
```

TheTrainingandTestingaccuracyscoresarebothhighandanincreaseonourinitialmodel.Trainingaccuracyhasincreasedby~6%andTestingaccuracyhasincreasedby~4%.

There is a marginal increase in the difference between the Training and Test scores (~6% com-pared to ~5% previously) though the difference remains low so the model has not suffered fromoverfitting.

E. Predictions

Herewewillmodifyyourpreviousmethodfortestingthemodelspredictionsonaspecifiedaudio

.wavfile.

```
defprint_prediction(file_name):

prediction_feature=extract_features(file_name)

prediction_feature=prediction_feature.reshape(1,num_rows,num_columns,num_channels)

predicted_vector=model.predict_classes(prediction_feature)

predicted_class = le.inverse_transform(predicted_vector)

print("Thepredictedclassis:",predicted_class[0],'\n')

predicted_proba_vector=model.predict_proba(prediction_feature)

predicted_proba=predicted_proba_vector[0]

foriinrange(len(predicted_proba)):

category=le.inverse_transform(np.array([i]))
```

print(category[0],"\t\t:",format(predicted_proba[i],'.32f'))

F. Validation

• Test with sample data

As before we will verify the predictions using a subsection of the sampleaudiofilesweexploredinthefirstnotebook.Weexpectthebulkofthesetobeclassifiedcorrectly.

#Class:AirConditioner

filename='../UrbanSoundDatasetsample/audio/100852-0-0-0.wav' print_prediction(filename)

Thepredictedclassis:air_conditioner

# Class: Drilling

filename='../UrbanSoundDatasetsample/audio/103199-4-0-0.wav' print_prediction(filename)

Thepredictedclassis:drilling

#Class:Streetmusic

filename='../UrbanSoundDatasetsample/audio/101848-9-0-0.wav' print_prediction(filename)

Thepredictedclassis:street_music

#Class:CarHorn

filename='../UrbanSoundDatasetsample/audio/100648-1-0-0.wav' print_prediction(filename)

Thepredictedclassis:drilling

• Observations

We can see that the model performs well.

Interestingly, car horn was again incorrectly classifed but this time as drilling - though the per class confidence shows it was a close decision between car horn with 26% confidence and drilling at 34% confidence.

G. Other Audio

Againwewillfurthervalidateourmodelusingasampleofvariouscopyrightfreesoundsthatwenotpartofeitherourtest ortrainingdata.

filename='../Evaluationaudio/dog_bark_1.wav' print_prediction(filename)

Thepredictedclassis:dog_bark

filename='../Evaluationaudio/drilling_1.wav' print_prediction(filename)

Thepredictedclassis:jackhammer

filename='../Evaluationaudio/gun_shot_1.wav' print_prediction(filename)

Thepredictedclassis:gun_shot

IV. Results

1. Model evaluation and validation

During the model development phase the validation data was used to evaluate the model. The final model architecture and hyperparameters were chosen because they performed the best among the tried combinations. This architecture is described in detail in section 3.

As we can see from the validation work in the previous section, to verify the robustness of the final model, a test was conducted using copyright free sounds from sourced from the internet. The following observations are based on the results of the test:

▪ The classifier performs well with new data.

▪ Misclassification does occur but seems to be between classes that are relatively similar such as Drilling and Jackhammer.

2. Justification

The final model achieved a classification accuracy of 92% on the testing data which exceeded my expectations given the benchmark was 68%.

Model	**ClassificationAccuracy**
CNN	92%
MLP	88%
BenchmarkSVM_rbf	68%

The final solution performs well when presented with a .wav file with a duration of a few seconds and returns a reliable classification.

However, we do not know how the model would perform on Real-time audio. We do not know whether it would be able to perform the classification in a timely manner so audio frames are not skipped or the classification would be heavily affected by latency.

Also, we do not know how the classifier would perform in a real world setting. Our study makes no attempt to determine the effect of factors such as noise, echos, volume and salience level of the sample.

V. Conclusion

1. Freeform visualization

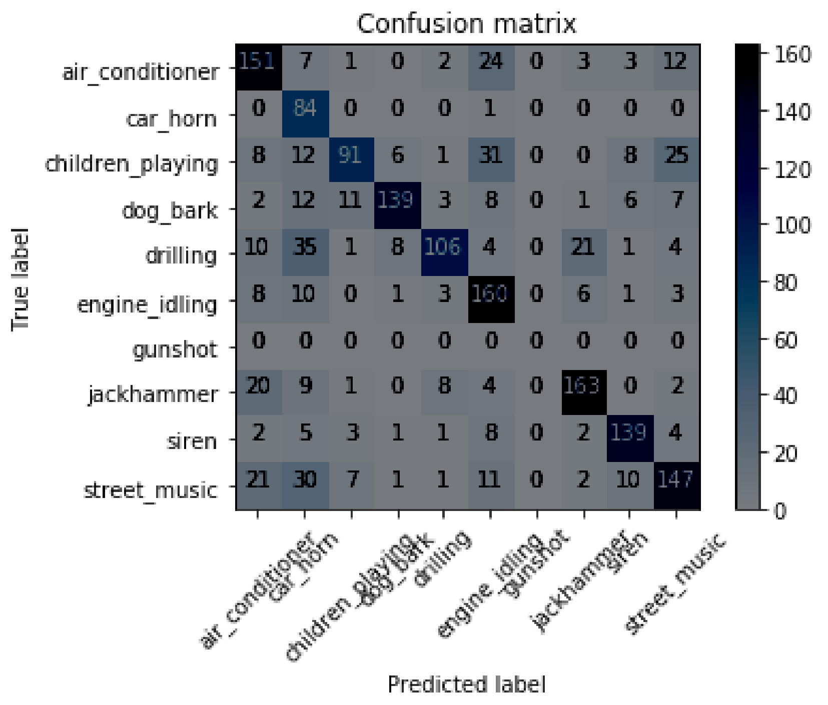
It was previously noted in our data exploration, that it is difficult to visualise the difference between some of the classes. In particular, the following sub-groups are similar in shape:

▪ Repetitive sounds for air conditioner, drilling, engine idling and jackhammer.

▪ Sharp peaks for dog barking and gun shot.

▪ Similar pattern for children playing and street music.

Using a confusion matrix we will examine if the final model also struggled to differentiate between these classes.



The Confusion Matrix tells a different story. Here we can see that our model struggles the most with the following sub-groups:

- ▪ air conditioner, jackhammer and street music.
- ▪ car horn, drilling, and street music.
- ▪ air conditioner, children playing and engine idling.
- ▪ jackhammer and drilling.
- ▪ air conditioner, car horn, children playing and street music.

This shows us that the problem is more nuanced than our initial assessment and gives some in-sights into the features that the CNN is extracting to make it's classifications. For example, street music is one of the commonly classified classes and could be to a wide variety of different samples within the class.

2. Reflection

The process used for this project can be summarised with the following steps:

1. The initial problem was defined and relevant public dataset was located.
2. The data was explored and analysed.
3. Data was preprocessed and features were extracted.
4. An initial model was trained and evaluated.
5. A further model was trained and refined.
6. The final model was evaluated.

From the initial exploration of the data in step 2, I envisaged that the preprocessing work in step 3 would be incredibly time consuming. However, this was actually relatively easy thanks to the Python tool Librosa. I also thought that the feature extraction would be a lot trickier but again Librosa shortened the effort required immensely.

MFCC's we extracted in step 3 perform much better than I had expected. However, we had to revisit the extraction process when we transitioned to using a CNN as our model. I did consider revisiting our MLP model to see how it performed with the updated feature extraction technique, but unfortunately there was not enough time for this.

Overall, the model performed better than planned. One observation we made during step 2 is that the dataset is slightly unbalanced with 2 out of the 10 classes having roughly a 40% sample size of the other 8. However, it is unclear whether this is significant enough to have caused any issues.

3. Improvement

If we were to continue with this project there are a number of additional areas that could be explored:

▪ As previously mentioned, test the models performance with Real-time audio.

▪ Train the model for real world data.

This would likely involve augmenting the training data in various ways such as:

• Adding a variety of different background sounds.

• Adjusting the volume levels of the target sound or adding echos.

• Changing the starting position of the recording sample, e.g. the shape of a dog bark.

▪ Experiment to see if per-class accuracy is affected by using training data of different durations.

▪ Experiment with other techniques for feature extraction such as different forms of Spectro- grams.

VI. References

[1] Justin Salamon, Christopher Jacoby and Juan Pablo Bello. Urban Sound Datasets. https://urbansounddataset.weebly.com/urbansound8k.html

[2] Mel-frequency cepstrum Wikipedia page https://en.wikipedia.org/wiki/Mel-frequency_cepstrum

[3] J. Salamon, C. Jacoby, and J. P. Bello. A dataset and taxonomy for urban sound research. http://www.justinsalamon.com/uploads/4/3/9/4/4394963/salamon_urbansound_acmmm14.pdf

[4] Manik Soni AI which classifies Sounds:Code:Python. https://hackernoon.com/ai-which-classifies-sounds-code-python-6a07a204381032

[5] Manash Kumar Mandal Building a Dead Simple Speech Recognition Engine using ConvNet in Keras. https://blog.manash.me/building-a-dead-simple-word-recognition-engine-using-convnet-in-keras-25e72c19c12b

[6] Eijaz Allibhai Building a Convolutional Neural Network (CNN) in Keras. https://towardsdatascience.com/building-a-convolutional-neural-network-cnn-in-keras-329fbbadc5f5

[7] Daphne Cornelisse An intuitive guide to Convolutional Neural Networks. https://medium.freecodecamp.org/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050

[8] Urban Sound Classification - Part 2: sample rate conversion, Librosa. https://towardsdatascience.com/urban-sound-classification-part-2-sample-rate-conversion-librosa-ba7bc88f209a

[9] wavsource.com THE Source for Free Sound Files and Reviews. http://www.wavsource.com/

[10] soundbible.com. http://soundbible.com//

[11] D. Bogdanov, N. Wack, E. G´omez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, J. Zapata, and X. Serra. ESSENTIA: an open-source library for sound and music analysis. In 21st ACM Int. Conf. on Multimedia, pages 855–858, 2013.

[12] A. L. Brown, J. Kang, and T. Gjestland. Towards standardization in soundscape preference assessment.Applied Acoustics, 72(6):387–392, 2011.

[13] L.-H. Cai, L. Lu, A. Hanjalic, H.-J. Zhang, and L.-H. Cai. A flexible framework for key audio effects detection and auditory context inference. IEEE TASLP, 14(3):1026–1039, 2006.

[14] S. Chaudhuri and B. Raj. Unsupervised hierarchical structure induction for deeper semantic analysis of audio. In IEEE ICASSP, pages 833–837, 2013.

[15] S. Chu, S. Narayanan, and C.-C. Kuo. Environmental sound recognition with time-frequency audio features.IEEE TASLP, 17(6):1142–1158, 2009.

[16] C. V. Cotton and D. P. W. Ellis. Spectral vs. spectro-temporal features for acoustic event detection. In IEEE WASPAA’11, pages 69–72, 2011.

[17] D. P. W. Ellis, X. Zeng, and J. H. McDermott. Classifying soundtracks with audio texture features. In IEEE ICASSP, pages 5880–5883, 2011.

[18] D. Giannoulis, D. Stowell, E. Benetos, M. Rossignol, M. Lagrange, and M. D. Plumbley. A database and challenge for acoustic scene classification and event detection. In 21st EUSIPCO, 2013.

[19] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: an update. SIGKDD Explorations Newsletter, 11(1):10–18, 2009.

[20] T. Heittola, A. Mesaros, A. Eronen, and T. Virtanen. Audio context recognition using audio event histograms. In 18th EUSIPCO, pages 1272–1276, 2010.

[21] T. Heittola, A. Mesaros, A. Eronen, and T. Virtanen. Context-dependent sound event detection. EURASIP JASMP, 2013(1), 2013.

[22] S. R. Payne, W. J. Davies, and M. D. Adams. Research into the practical and policy applications of soundscape concepts and techniques in urban areas. DEFRA, HMSO, London, UK, 2009.

[23] R. Radhakrishnan, A. Divakaran, and P. Smaragdis. Audio analysis for surveillance applications. In IEEE WASPAA’05, pages 158–161, 2005.

[24] R. M. Schafer. The Soundscape: Our Sonic Environment and the Tuning of the World. Destiny Books, 1993.

[25] D. Steele, J. D. Krijnders, and C. Guastavino. The sensor city initiative: cognitive sensors for soundscape transformations. In GIS Ostrava, pages 1–8, 2013.

[26] M. Xu, C. Xu, L. Duan, J. S. Jin, and S. Luo. Audio keywords generation for sports video analysis. ACM TOMCCAP, 4(2):1–23, 2008

[27] S Ntalampiras, I Potamitis, N Fakotakis, in IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP ’09. On acoustic surveillance of hazardous situations (IEEE Computer Society, Washington, DC, USA, 2009), pp. 165–168

[28] S Chu, S Narayanan, CCJ Kuo, Environmental sound recognition with time-frequency audio features. IEEE Trans. Audio Speech Lang. Process. 17(6), 1142–1158 (2009)

[29] T Heittola, A Mesaros, A Eronen, T Virtanen, in 18th European Signal Processing Conference. Audio context recognition using audio event histograms (Aalborg, Denmark, 2010), pp. 1272–1276

[30] M Shah, B Mears, C Chakrabarti, A Spanias, in 2012 IEEE International Conference on Emerging Signal Processing Applications (ESPA). Lifelogging: archival and retrieval of continuously recorded audio using wearable devices (IEEE Computer Society, Las Vegas, NV, USA, 2012), pp. 99–102

[31] G Wichern, J Xue, H Thornburg, B Mechtley, A Spanias, Segmentation, indexing, and retrieval for environmental and natural sounds. IEEE Trans. Audio Speech Lang. Process. 18(3), 688–707 (2010)

[32] AS Bregman, Auditory Scene Analysis. (MIT Press, Cambridge MA, 1990)

[33] M Bar, The proactive brain: using analogies and associations to generate predictions. Trends Cogn. Sci. 11(7), 280–289 (2007)

[34] A Oliva, A Torralba, The role of context in object recognition. Trends Cogn. Sci. 11(12), 520–527 (2007)

[35] M Niessen, L van Maanen, T Andringa, in IEEE International Conference on Semantic Computing. Disambiguating sounds through context (IEEE Computer Society, Santa Clara, CA, USA, 2008), pp. 88–95

[36] C Clavel, T Ehrette, G Richard, in IEEE International Conference on Multimedia and Expo. Events detection for an audio-based surveillance system (IEEE Computer Society, Los Alamitos, CA, USA, 2005), pp. 1306–1309

[37] H Wu, J Mendel, Classification of battlefield ground vehicles using acoustic features and fuzzy logic rule-based classifiers. IEEE Trans. Fuzzy Syst. 15, 56–72 (2007)

[38] L Atlas, G Bernard, S Narayanan, Applications of time-frequency analysis to signals from manufacturing and machine monitoring sensors. Proc. IEEE. 84(9), 1319–1329 (1996)

[39] S Fagerlund, Bird species recognition using support vector machines. EURASIP J. Appl. Signal Process. 2007, 64–64 (2007)
[40] F Kraft, R Malkin, T Schaaf, A Waibel, in Proceedings of Interspeech. Temporal ICA for classification of acoustic events in a kitchen environment (International Speech Communication Association, Lisboa, Portugal, 2005), pp. 2689–2692
[41] J Chen, AH Kam, J Zhang, N Liu, L Shue, in Pervasive Computing. Bathroom activity monitoring based on sound (Springer, Berlin, 2005), pp. 47–61
[42] A Temko, C Nadeu, Classification of acoustic events using SVM-based clustering schemes. Pattern Recognit. 39(4), 682–694 (2006)
[43] TH Dat, H Li, in IEEE International Conference on Acoustics, Speech and Signal Processing. Probabilistic distance SVM with Hellinger-exponential kernel for sound event classification (IEEE Computer Society, Prague,Czech Republic, 2011), pp. 2272–2275
[44] R Stiefelhagen, R Bowers, J(eds) Fiscus, Multimodal Technologies for Perception of Humans: International Evaluation Workshops CLEAR 2007 and RT 2007. (Springer, Berlin Germany, 2008)
[45] X Zhou, X Zhuang, M Liu, H Tang, M Hasegawa-Johnson, T Huang, in Multimodal Technologies for Perception of Humans: International Evaluation Workshops CLEAR 2007 and RT 2007. HMM-based acoustic event detection with AdaBoost feature selection (Springer, Berlin, Germany, 2008), pp. 345–353
[46] X Zhuang, X Zhou, MA Hasegawa-Johnson, TS Huang, Real-world acoustic event detection. Pattern Recognit. Lett. (Pattern Recognition of Non-Speech Audio). 31(12), 1543–1551 (2010)
[47] A Mesaros, T Heittola, A Eronen, T Virtanen, in 18th European Signal Processing Conference. Acoustic event detection in real-life recordings (Aalborg, Denmark, 2010), pp. 1267–1271
[48] M Akbacak, JHL Hansen, Environmental sniffing: noise knowledge estimation for robust speech systems. IEEE Trans. Audio Speech Lang. Process. 15(2), 465–477 (2007)
[49] A Mesaros, H Heittola, A Klapuri, in 19th European Signal Processing Conference. Latent semantic analysis in sound event detection (Barcelona, Spain, 2011), pp. 1307-1311
[50] A Eronen, V Peltonen, J Tuomi, A Klapuri, S Fagerlund, T Sorsa, G Lorho, J Huopaniemi, Audio-based context recognition. IEEE Trans. Audio Speech Lang. Process. 14, 321–329 (2006)
[51] JJ Aucouturier, B Defreville, F Pacher, The bag-of-frames approach to audio pattern recognition: a sufficient model for urban soundscapes but not for polyphonic music. J. Acoust. Soc. Am. 122(2), 881–891 (2007)
[52] R Cai, L Lu, A Hanjalic, Co-clustering for auditory scene categorization. IEEE Trans. Multimed. 10(4), 596–606 (2008)
[53] L Lie, A Hanjalic, Text-like segmentation of general audio for content-based retrieval. IEEE Trans. Multimed. 11(4), 658–669 (2009)
[54] LR Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition. Proc. IEEE. 77(2), 257–286 (1989)Heittola et al. EURASIP Journal on Audio, Speech, and Music Processing 2013, 2013:1 Page 13 of 13 http://asmp.eurasipjournals.com/content/2013/1/1
[55] D Reynolds, R Rose, Robust text-independent speaker identification using Gaussian mixture speaker models. IEEE Trans. Speech Audio Process. 3, 72–83 (1995)
[56] GD Forney, The Viterbi algorithm. Proc. IEEE. 61(3), 268–278 (1973)
[57] M Ryynanen, A Klapuri, in ¨ Proceedings of the 2005 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics. Polyphonic music transcription using note event modeling (IEEE Computer Society, New York, NY, USA, 2005), pp. 319–322
[58] A Temko, C Nadeu, D Macho, R Malkin, C Zieger, M Omologo, in Computers in the Human Interaction Loop, ed. by AH Waibel, R Stiefelhagen. Acoustic event detection and classification (Springer, New York, 2009), pp. 61–73
[59] M Grootel, T Andringa, J Krijnders, in Proceedings of the NAG/DAGA Meeting 2009. DARES-G1: database of annotated real-world everyday sounds (Rotterdam, Netherlands, 2009), pp. 996–999
[60] T Heittola, A Mesaros, T Virtanen, A Eronen, in Workshop on Machine Listening in Multisource Environments, CHiME2011. Sound event detection in multisource environments using source separation (Florence, Italy, 2011), pp. 36–40
[61] S. Chu, S. Narayanan, and C.-C.J. Kuo, “Environmental sound recognition with time-frequency audio features,” IEEE TASLP, vol. 17, no. 6, pp. 1142–1158, 2009.
[62] R. Radhakrishnan, A. Divakaran, and P. Smaragdis, “Audio analysis for surveillance applications,” in IEEE WASPAA’05, 2005, pp. 158–161.
[63] M. Xu, C. Xu, L. Duan, J. S. Jin, and S. Luo, “Audio keywords generation for sports video analysis,” ACM TOMCCAP, vol. 4, no. 2, pp. 1–23, 2008.
[64] Y.-F. Ma, L. Lu, H.-J. Zhang, and M. Li, “A user attention model for video summarization,” in 10th ACM Int. Conf. On Multimedia, 2002, pp. 533–542.

[65] D. Steele, J. D. Krijnders, and C. Guastavino, "The sensor city initiative: cognitive sensors for soundscape transformations," in GIS Ostrava, 2013, pp. 1–8.

[66] D. P. W. Ellis and K. Lee, "Minimal-impact audio-based personal archives," in 1st ACM workshop on Continuous archival and retrieval of personal experiences, New York, NY, USA, Oct. 2004, pp. 39–47.

[67] T. Heittola, A. Mesaros, A. Eronen, and T. Virtanen, "Audio context recognition using audio event histograms," in 18th EUSIPCO, 2010, pp. 1272–1276.

[68] S. Chaudhuri and B. Raj, "Unsupervised hierarchical structure induction for deeper semantic analysis of audio," in IEEE ICASSP, 2013, pp. 833–837.

[69] L.-H. Cai, L. Lu, A. Hanjalic, H.-J. Zhang, and L.-H Cai, "A flexible framework for key audio effects detection and auditory context inference," IEEE TASLP, vol. 14, no. 3, pp. 1026–1039, 2006.

[70] T. Heittola, A. Mesaros, A. Eronen, and T. Virtanen, "Context dependent sound event detection," EURASIP JASMP, vol. 2013, no. 1, 2013.

[71] T. Ganchev, N. Fakotakis, and G. Kokkinakis, "Comparative evaluation of various MFCC implementations on the speaker verification task," in 10th Int. Conf. on Speech and Computer, Patras, Greece, Oct. 2005, vol. 1, pp. 191–194.

[72] C. V. Cotton and D. P. W. Ellis, "Spectral vs. spectro-temporal features for acoustic event detection," in IEEE WASPAA'11, 2011, pp. 69–72.

[73] D. Stowell and M. D. Plumbley, "Automatic large-scale classification of bird sounds is strongly improved by unsupervised feature learning," PeerJ, vol. 2, pp. e488, Jul. 2014.

[74] S. Dieleman and B. Schrauwen, "Multiscale approaches to music audio feature learning," in 14th Int. Soc. for Music Info. Retrieval Conf., Curitiba, Brazil, Nov. 2013.

[75] P. Hamel, S. Lemieux, Y. Bengio, and D. Eck, "Temporal pooling and multiscale learning for automatic annotation and ranking of music audio.," in 12th Int. Soc. for Music Info. Retrieval Conf., Miami, USA, Oct. 2011, pp. 729–734.

[76] Y. Vaizman, B. McFee, and G. Lanckriet, "Codebook-based audio feature representation for music information retrieval," IEEE Transactions on Audio, Speech, and Language Processing, vol. 22, no. 10, pp. 1483–1493, Oct. 2014.

[77] E. Amid, A. Mesaros, K. J Palomaki, J. Laaksonen, and M. Kurimo, "Unsupervised feature extraction for multimedia event detection and ranking using audio content," in IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), Florence, Italy, May 2014, pp. 5939–5943.

[78] A. Coates and A. Y. Ng, "Learning feature representations with K-means," in Neural Networks: Tricks of the Trade, pp. 561–580. Springer, 2012.

[79] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in 22nd ACM International Conference on Multimedia (ACM-MM'14), Orlando, FL, USA, Nov. 2014.

[80] D. Bogdanov, N. Wack, E. Gomez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, J. Zapata, and X. Serra, "ESSENTIA: an audio analysis library for music information retrieval," in 14th Int. Soc. for Music Info. Retrieval Conf., Curitiba, Brazil, Nov. 2013, pp. 493–498.

[81] S. Lloyd, "Least squares quantization in PCM," IEEE Trans. on Information Theory, vol. 28, no. 2, pp. 129–137,1982.

[82] I.S. Dhillon and D.M. Modha, "Concept decompositions for large sparse text data using clustering," Machine Learning, vol. 42, no. 1, pp. 143–175, 2001.

[83] L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.

[84] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.

[85] T. Fawcett, "An introduction to ROC analysis," Pattern Recognition Letters, vol. 27, no. 8, pp. 861–874, 2006.

[86] D. P. W. Ellis, X. Zeng, and J. H. McDermott, "Classifying soundtracks with audio texture features," in IEEE ICASSP, 2011, pp. 5880–5883