BUSINESS ANALYTICS Master of Science

ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Big Data Content Analytics

Audioset Analysis

Sound Classification with Neural Networks

Project team

Kotoulas Evangelos – f2821816

Kyritsis Georgios – f2821818

Voudris Athanasios – f2821828

September 2019

Contents

1. Introduction

Directly or indirectly, we are always in contact with audio. Our brain is continuously processing and understanding audio data and giving us information about the environment. A simple example can be our conversations with people which we do daily. This speech is discerned by the other person to carry on the discussions. Even when we think we are in a quiet environment, we tend to catch much more subtle sounds, like the rustling of leaves or the splatter of rain. This is the extent of our connection with audio. [1]

There are devices built which help us catch these sounds and represent it in computer readable format. Examples of these formats are: [1]

- wav (Waveform Audio File) format
- mp3 (MPEG-1 Audio Layer 3) format
- WMA (Windows Media Audio) format

If we give a thought on what an audio looks like, it is nothing but a wave like format of data, where the amplitude of audio change with respect to time. [1]

In addition to the common use of speech-to-text algorithms, analysis of audio data can provide important knowledge. From sounds made by mechanical devices that pre-cursor an approaching failure to monitoring the dynamics of a crowd of people, audio data can provide important insights. [2]

With the evolution of artificial intelligence and machine learning, artificial neural networks and deep learning have been actively employed to identify sounds and understand audio data. Using neural networks, the aim of this project is to set up an analysis of the audio model that would be able to identify sounds. The project was developed in Python, using Keras on the Tensorflow backend.

2. Business Idea and Description of Project Mission

Audio analytics is the process of compressing data and packaging the data into single format called audio. Audio Analytics refers to the extraction of meaning and information from audio signals for Analysis. There are two ways to represent the audio analytics:

- Sound Representation and
- Raw Sound Files.

Audio file format is a format for storing digital audio data on a system. There are three main audio format:

- Uncompressed audio format
- Lossless compressed audio format and
- Lossy compressed audio format. [3], [4]

Application Area of Audio Analytics

The audio is the file format that was used to transfer the data from one place to another. Audio analytics is used to check whether given audio data is available in proper format or in similar format that sender sends. The Application of audio analytics are many:

- Surveillance application: Surveillance application is based on approach for systematic choice of audio classes for detection of crimes done in society. A surveillance application is based on audio Analytics framework is the only way to detect suspicious kinds of activity. The application is also used to send some important information to surveillance at some crisis situation urgently.
- Detection of Threats: The audio mechanism is used to identify the thread that take place between sender and receiver.
- Tele-monitoring System: New technology have camera with the facilities to record the audio also. Audio Analytics may provide effective detection of screams, breaking glass, gun sound, explosions, calling for help sound etc. Combination of audio Analytics and video Analytics in single monitoring system result as a good threat detection efficiency.
- Mobile Networking System: The Mobile networking system is used to talk or transfer information from one place to another place. Sometimes due to some network problems the audio sound is not working properly at that time Audio Analytics is used to find the information that not send properly due to some problems. [3]
- 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.
- Industrial uses such as predictive maintenance. [5]

Having all these in mind, it is apparent that there is a need to classify sounds with high accuracy and at scale.

The mission of the project is to try and build a model that can with a good accuracy predict, based on recorded audio data, the classification of audio data. To build this application we will use neural networks, which have been extensively investigated in academic literature as a good approach with high accuracy ratios in such classification problems.

Audio data is analogue sound wave that coded in digital form. [6]

Audio signal classification system analyzes the input audio signal and creates a label that describes the signal at the output. These are used to characterize both music and speech signals. The categorization can be done on the basis of pitch, music content, music tempo and rhythm. The signal classifier analyzes the content of the audio format thereby extracting information about the content from the audio data. This is also called audio content analysis, which extends to retrieval of content information from signals. [7]

3. Data

3.1. Dataset Description

The dataset that was finally chosen for this specific assignment was found in the following link:

https://research.google.com/audioset/index.html

AudioSet consists of an expanding ontology of 527 audio event classes and a collection of 2,084,320 human-labeled 10-second sound clips drawn from YouTube videos. The ontology is specified as a hierarchical graph of event categories, covering a wide range of human and animal sounds, musical instruments and genres, and common everyday environmental sounds. [8]

In total, there are 2.1 million annotated videos, 5.8 thousand hours of audio and 527 classes (labels) of annotated sounds.

The ontology is released as a JSON file containing the following fields for each category:

- ID: The Knowledge Graph Machine ID (MID) best matching the sound or source, used as the primary identifier for the class. In Knowledge Graph, many MIDs refer to specific objects (e.g., /m/0gy1t2s, “Bicycle bell” or /m/02y 763, “Sliding door”); when used in the ontology, they are understood to mean “the sound readily associated with the specified object”. For instance, in the case of “Sliding door”, the sound made by someone crashing into a closed door would be emanating from the door; however, it would not be the characteristic short burst of bearing noise that suggests “Sliding door”, so it should instead be labelled “Smash” or “Thump”.

- Display name: A brief one or two word name for the sound, sometimes with a small number of comma-separated alternatives (e.g., “Burst, pop”), and sometimes with parenthesized disambiguation (e.g. “Fill (with liquid)”). Display names are intended to be unambiguous even when shown without their ancestor nodes.

- Description: A longer description, typically one or two sentences, used to provide more explanation of the meaning and limits of the class. In many cases these are based on Wikipedia or WordNet descriptions (with appropriate citation URIs), adapted to emphasize their specific use for audio events.

The dataset consists of csv files describing, for each segment, the YouTube video ID, start time, end time, and one or more labels.

The dataset is divided into three disjoint sets: a balanced evaluation set, a balanced training set, and an unbalanced training set.

Evaluation - eval_segments.csv

20,383 segments from distinct videos, providing at least 59 examples for each of the 527 sound classes that are used. Because of label co-occurrence, many classes have more examples.

Balanced train - balanced_train_segments.csv

22,176 segments from distinct videos chosen with the same criteria: providing at least 59 examples per class with the fewest number of total segments.

Unbalanced train - unbalanced_train_segments.csv

2,042,985 segments from distinct videos, representing the remainder of the dataset.

Each csv file has a three-line header with each line starting with "#", and with the first two lines indicating the creation time and general statistics:

# Segments csv created Sun Mar 5 10:54:25 2017

# num_ytids=20371, num_segs=20371, num_unique_labels=527, num_positive_labels=51804

Each subsequent line has columns defined by the third header line:

# YTID, start_seconds, end_seconds, positive_labels

For example:

-0RWZT-miFs, 420.000, 430.000, "/m/03v3yw,/m/0k4j"

means that for the YouTube video -0RWZT-miFs, for the 10 second chunk from t=420 sec to t=430 sec, annotators confirmed the presence of sound classes /m/03v3yw ("Keys jangling") and /m/0k4j ("Car").

The total size of the features is 2.4 gigabytes. They are stored in 12,228 TensorFlow record files, sharded by the first two characters of the YouTube video ID, and packaged as a tar.gz file.

The labels are stored as integer indices. They are mapped to sound classes via class_labels_indices.csv. The first line defines the column names:

index,mid,display_name

Subsequent lines describe the mapping for each class. For example:

0,/m/09x0r,"Speech"

which means that "labels" with value 0 indicate segments labelled with "Speech".

3.2 Data downloading and preparation

The files were downloaded using the code from a user in GitHub. [9] The code is written in JavaScript programming language and was run through a Windows terminal. The code was reading the csv file line by line and for each line, was keeping the URL of the YouTube video, then it was downloading the whole video and finally keeping only the part that is between the corresponding time periods (e.g. **-0RWZT-miFs, 420.000, 430.000**). The output was an audio file which had the URL as name, in order to be unique. The whole process takes a long time, so for this reason multiple terminals in multiple PCs were running simultaneously in order to download the files, each terminal running for different row numbers of the csv files. Several weeks were needed for the most files from the balanced and evaluation csv datasets to be downloaded. Also, a few thousand files were downloaded from the unbalanced csv file.

This whole process faced multiple errors and some files could not be downloaded. A list with the files that were downloaded was needed in order to be able to continue with the analysis. Firstly, a database was created in MySQL, an open-source relational database management system, with three empty tables (for the balanced, evaluation and unbalanced data). Then, with a few lines of code in R, a programming language and free software environment for statistical computing, graphics and data analysis, the csv files were read, manipulated and transferred to the corresponding three tables in MySQL. A new column was created for the three tables with the name ‘file_downloaded’. This column was filled with 0’s which meant that none of these files were downloaded.

Then, after having all the data from the csv files in the tables in MySQL database, three Python scripts were created, one for each csv file. These scripts make a connection with the database in MySQL and then check the folder with the downloaded files. Then they assign the value ‘1’ to the field ‘file_downloaded’ in the tables of the database. The final step is extracting the tables from MySQL to csv files for further analysis. So, the files balanced.csv, evaluation.csv and unblanaced.csv were created, which contained among the initial values, the file names, the labels and Boolean values for ‘file_downloaded’ column.

4. Exploratory Data Analysis

4.1. EDA Technology

Exploratory Data Analysis (EDA) is an approach to analyse datasets to summarize their main characteristics mostly by using data visualization methods.

Exploratory Data Analysis is one of the important steps in the data analysis process. Here, the focus is on making sense of the data in hand — things like formulating the correct questions to ask to your dataset, how to manipulate the data sources to get the required answers, and others. [10]

EDA, as the whole project was implemented using python. During the past years, python despite that is an interpreted high-level programming language for general-purpose programming, it has also been used as a scientific language. There are many libraries that offer the capability of scientific computations and serve purposes such as Data management and Deep Learning.

In this assignment, the scientific packages Pandas and NumPy were used. NumPy is a third-party library, which further extends the native mathematics capabilities and is frequently used in scientific scripting language to aid in problems of data processing and manipulation. Its name stands for “Numerical Python”. Similar to NumPy, Pandas is one of the most widely used Python libraries in data science. Pandas is another library that due to its intuitive syntax and high-performance matrix computation capabilities offers Matrix and Vector manipulations for Machine Learning purposes. It provides high-performance, easy to use structures and data analysis tools.

Also, the package Keras has been used. Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Keras allows for easy and fast prototyping (through user friendliness, modularity, and extensibility), supports both convolutional networks and recurrent networks, as well as combinations of the two and runs seamlessly on CPU and GPU. The core data structure of Keras is a model, a way to organize layers. The simplest type of model is the Sequential model, a linear stack of layers. [11]

“TensorFlow" is an open-source software library for dataflow programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks.

The package “OS” has been utilized as well. The OS module in Python provides a way of using operating system dependent functionality. The functions that the OS module provides allows you to interface with the underlying operating system that Python is running on. [12]

Finally, Librosa has been employed. It is a python package for music and audio analysis. It provides the building blocks necessary to create music information retrieval systems.

4.2. EDA Process

The following paragraphs will demonstrate how to apply Deep Learning techniques to the classification of sounds. When given an audio sample in a computer readable format (such as a .mp3 file) of a few seconds duration, we want to be able to determine if it contains one of the target sounds with a corresponding Classification Accuracy score. [13]

At first, the balanced csv file is loaded and the dataframe is called balanced. The csv file includes the following columns:

- url (i.e. –PJHxphWEs)
- start_time (i.e. 30.0)
- end_time (i.e. 40.0)
- labels (i.e. /m/09x0r,/t/dd00088)
- file_downloaded (i.e. 1)
- filename (i.e. __pjhxphwes.mp3)

Following that, a temporary dataframe, which is called temp, is created including the labels of each file. The labels are loaded in a dataframe named labels and the dictionary has the following format:

'/g/122z_qxw': 'Firecracker',

'/m/011k_j': 'Timpani',

'/m/01280g': 'Wild animals',

'/m/012f08': 'Motor vehicle (road)',

'/m/012n7d': 'Ambulance (siren)',

'/m/012ndj': 'Fire engine, fire truck (siren)' etc.

All the indexes (i.e. /g/122z_qxw) are replaced in the temp dataframe with their labels (i.e. Firecracker). The columns of the temp dataframe have been renamed as Label1, Label2,..., Label 12 and a new column has been added for the total number of labels per file. Finally, the dataframes balanced and temp are concatenated and the final balanced dataframe includes the following columns:

- url (i.e. –PJHxphWEs)
- start_time (i.e. 30.0)
- end_time (i.e. 40.0)
- file_downloaded (i.e. 1)

- filename (i.e. __pjhxphwes.mp3)
- Label1 (i.e. Speech), Label2 (i.e. Gush), Label3 (i.e. None), Label4, Label5, Label6, Label7, Label8, Label9, Label10, Label11, Label12
- Number_Labels (i.e. 3)

Finally, the total number of each label is counted in the balanced csv file. For example, the label Music is observed 5553 times in the 22176 files in the balanced csv file.

The same process is followed for the evaluation csv file.

4.3. Preparation of the train dataset

A new column called ‘labels’ is added in the balanced dataframe. This column refers to the names of the labels which are included in every mp3 file.

4.4. Create the train dataset

The train dataset (train_data) is created consisting of two columns: the filename (i.e. __pjhxphwes.mp3) and the labels (i.e. Speech|Gush).

Regarding the data cleaning and transformation, the corrupted mp3 files were removed developing and using the ‘parser’ function. In total, 20 mp3 files in the train dataset out of the 19938 mp3 files were found to be corrupted and consequently they have been removed from the train dataset.

In parallel, a dataframe named df is created including the filenames (i.e. __pjhxphwes.mp3) and the number of labels (i.e. 2 labels).

4.5. Preparation of the validation dataset

The same process, as in the preparation of the train dataset, has been adopted for the preparation of the validation dataset. The validation dataset (validation_data) is created consisting of two columns: the filename (i.e. __4gqaraeje.mp3) and the labels (i.e. Domestic animals, pets|Squeak|Dog|Animal).

Regarding the data cleaning and transformation, the corrupted mp3 files were removed developing and using the ‘parser’ function. In total, 8 mp3 files in the validation dataset out of the 18680 mp3 files were found to be corrupted and consequently they have been removed from the validation dataset.

In parallel, a dataframe named df2 is created including the filenames (i.e. __4gqaraeje.mp3) and the number of labels (i.e. 4 labels).

4.6. EDA Results

From a visual perspective it is tricky to visualise both the similarities of sounds of the same classes and the differences between some of the classes. As an example we present below the waveforms of 4 files. Both files _unpp3fewse and btahd1y_41o are sounds generated by a drill, while 0_rnk6mnmca and _wfsdr9xbvo are recordings of an air condition.

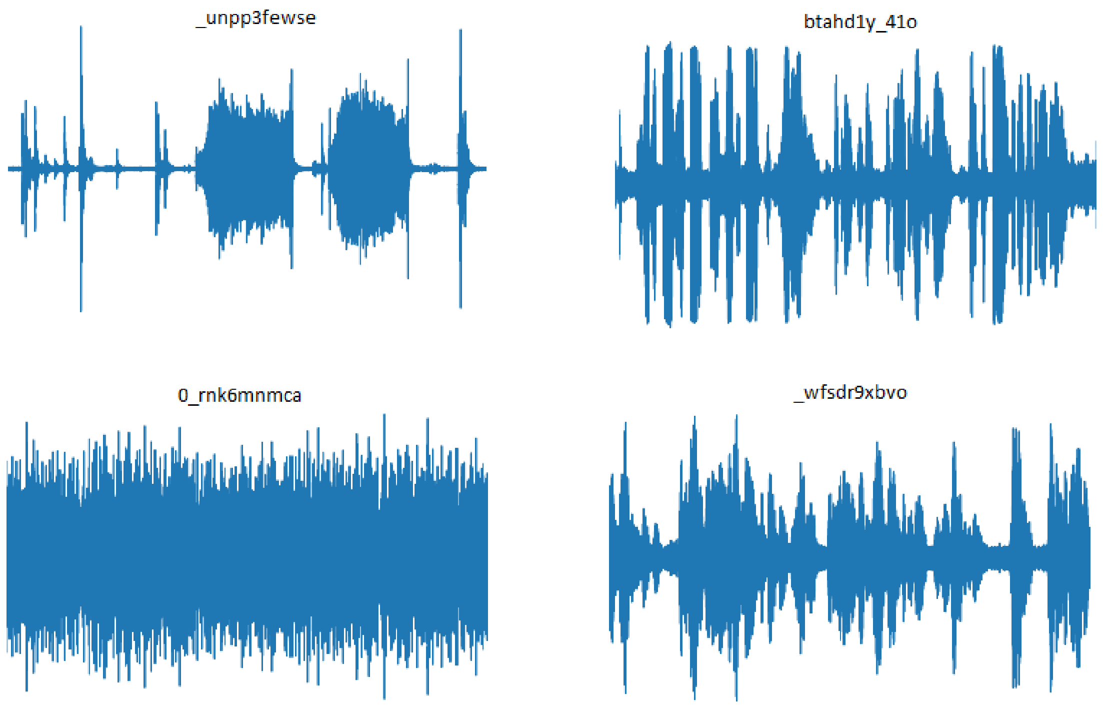


Figure 1. Waveforms for drill (_unpp3fewse and btahd1y_41o) and air-condition (0_rnk6mnmca and _wfsdr9xbvo)

For the above presented images we have used the embedded function of the Librosa library waveplot, but as it can be observed the waveforms of the sounds do not depict a clear distinctive characteristic for classification purposes.

4.7. Extract features

The next step is to extract the features which will be needed to train our model. To do this, a visual representation of each of the audio samples will be created which will allow us to identify features for classification, using the same techniques used to classify images with high accuracy.

Spectrograms are a useful technique for visualising the spectrum of frequencies of a sound and how they vary during a very short period of time. A spectrogram uses a linear spaced frequency scale (so each frequency bin is spaced an equal number of Hertz apart).

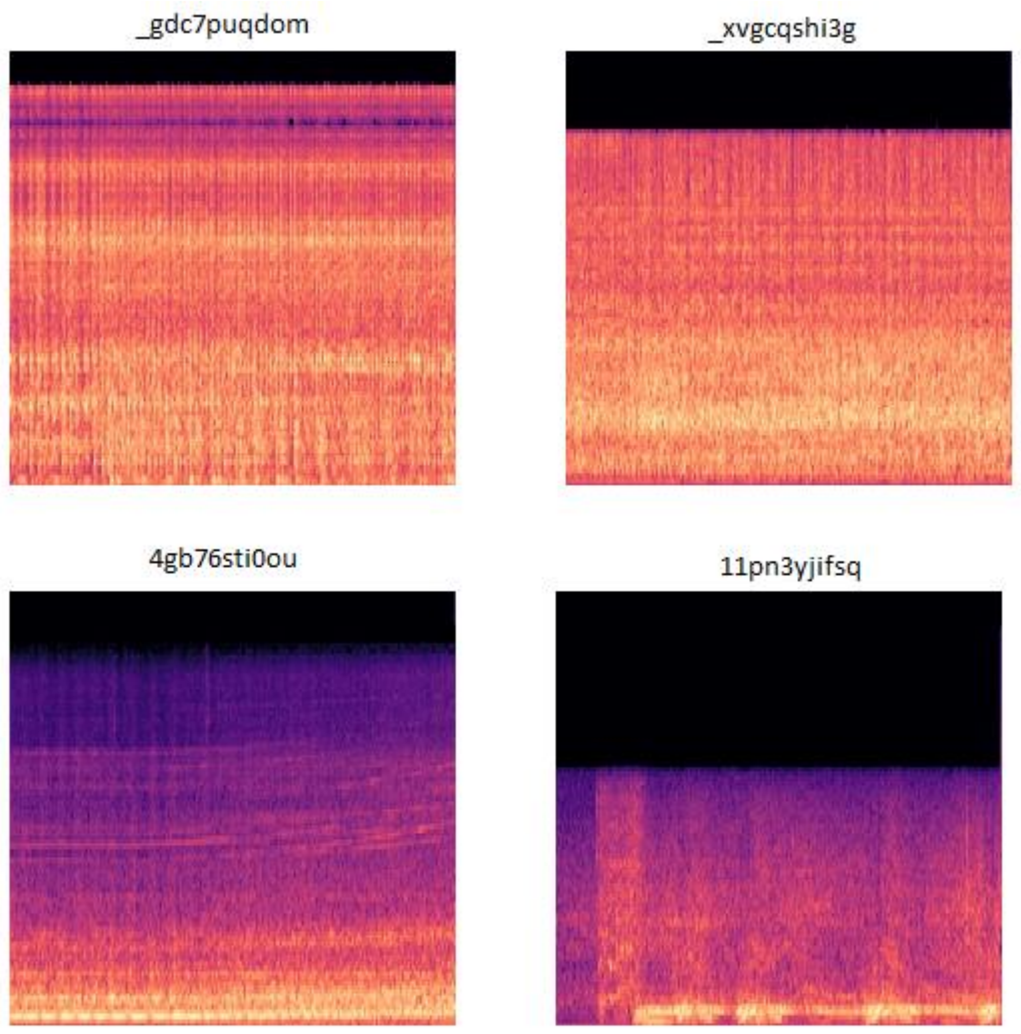


Figure 2. A spectrogram showing the energy in different frequency bands changing over time

As it is shown in the pictures above, characteristic features of each class are more homogeneous than before. The two presented pairs of files, (_gdc7puqdom, _xvgcqshi3g) and (4gb76sti0ou,11pn3yjifsq) belong to similar classes Engine and Accelerating, revving, vroom but with the use of spectrograms we could make a distinction between the two groups with much more ease.

For much of the preprocessing Librosa's load() function is used, which by default converts the sampling rate to 22.05 KHz, normalise the data so the bit-depth values range between -1 and 1 and flattens the audio channels into mono. [14]

4.8. Converting the data and labels then splitting the dataset

In order for the training process of the first model to start, the data need to be handled accordingly. The files from the balanced csv were used as train dataset and the files from the evaluation csv were used as validation dataset. All files were parsed with a custom function that extracts the MFCC of the audio files using the librosa library and saves the generated values in a dataframe. The result for each file is 40 sequential numbers. These numbers represent the audio file through time. The function faced some errors and a few files did not transform, so it was a must that these files should be removed from the dataset. After finding these files, they were removed from the folder and the corresponding rows in the train dataset and train labels dataset were completely removed. Then, an array was created from the dataframe with the mfcc values with proper shape, for example (19918,40) for the train dataset which means 19918 files with 40 values as input per file and (19918,527) shape for the labels of the files. That means that for each file, 527 columns are created filled with 0's and 1's where 1 means that these labels represent the specific file. Of course, there may exist many 1's in each row. The same process was done for the validation dataset. Now the data were ready for model training.

For the image classification approach we converted both the training (19918 files) and validation dataset (18680) to spectrogram images and store them individually. Its image has a shape of (64,64,3) and as with our previous approach can belong in one or several classes.

5. Methodology and Model Development

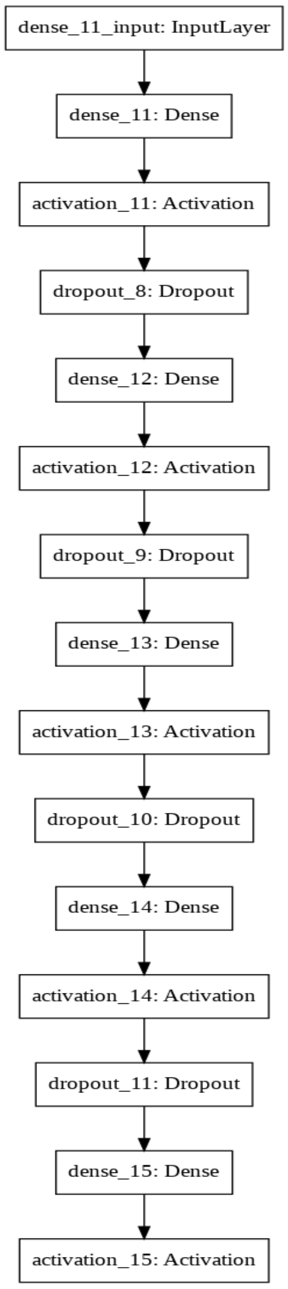
The model specification started by defining the type of model we want to build. There are two types of models available in Keras: the Sequential model and the Model class used with functional API. The chosen type for our case was the sequential.

At every layer, we use “Dense” which means that the units are fully connected. Within the hidden-layers, we use the ReLu function as the activation function. ReLu is probably the most widely used activation function as it incorporates significant assets compared to others including, extremely fast computation due to its mathematical simplicity and avoidance of vanishing gradients. Another technique that was used for better results was the drop-out technique on each layer.

5.1. Building the first model

The next step will be to build and train a Deep Neural Network with these data sets and make predictions.

Below, the first model construction is provided. The input shape of the first layer of the model is (40,) for the 40 different values for each file. As a final layer an activation layer was chosen with sigmoid function.


dense_11_input: InputLayer
dense_11: Dense
activation_11: Activation
dropout_8: Dropout
dense_12: Dense
activation_12: Activation
dropout_9: Dropout
dense_13: Dense
activation_13: Activation
dropout_10: Dropout
dense_14: Dense
activation_14: Activation
dropout_11: Dropout
dense_15: Dense
activation_15: Activation

The model started training the data from the balanced csv with the MFCC values. It was initially configured to run for 50 epochs with training and validation data. It was also configured to stop the training if the validation loss did not improve for 4 sequential epochs. The model training after 12 epochs stopped early as no better validation loss was observed.

For the second model we extracted 128 MFCC values from all audio files and each file was loaded in Librosa library with the 'kaiser_best' parameter, which means it was loaded with the best quality. So, a similar model was constructed with input shape = (128,), which is the number of values for each file. Now, the metric 'val_top_k_categorical_accuracy' was chosen for the model to stop training, when the max value was observed, again with patience = 4.

The final model with this method was constructed with more layers. The model looks like this:

```
Layer (type)                 Output Shape              Param #   
=================================================================
dense_92 (Dense)             (None, 1024)              132096
_________________________________________________________________
activation_122 (Activation)  (None, 1024)              0
_________________________________________________________________
dropout_92 (Dropout)         (None, 1024)              0
_________________________________________________________________
dense_93 (Dense)             (None, 1024)              1049600
_________________________________________________________________
activation_123 (Activation)  (None, 1024)              0
_________________________________________________________________
dropout_93 (Dropout)         (None, 1024)              0
_________________________________________________________________
dense_94 (Dense)             (None, 512)               524800
_________________________________________________________________
activation_124 (Activation)  (None, 512)               0
_________________________________________________________________
dropout_94 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_95 (Dense)             (None, 512)               262656
_________________________________________________________________
activation_125 (Activation)  (None, 512)               0
_________________________________________________________________
dropout_95 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_96 (Dense)             (None, 256)               131328
_________________________________________________________________
activation_126 (Activation)  (None, 256)               0
_________________________________________________________________
dropout_96 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_97 (Dense)             (None, 256)               65792
_________________________________________________________________
activation_127 (Activation)  (None, 256)               0
_________________________________________________________________
dropout_97 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_98 (Dense)             (None, 128)               32896
_________________________________________________________________
activation_128 (Activation)  (None, 128)               0
_________________________________________________________________
dropout_98 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_99 (Dense)             (None, 527)               67983
_________________________________________________________________
activation_129 (Activation)  (None, 527)               0
=================================================================
Total params: 2,267,151
Trainable params: 2,267,151
Non-trainable params: 0
```

The monitored metric was again the ‘val_top_k_categorical_accuracy’ with patience = 5, but for this model we tried something different. The input data for this model was the majority of the balanced and validation data (37500) and for the validation and test data we used 545 data for each set (almost 1.5% of the whole data).

5.2. Building the second model

Here a Convolutional Neural Network (CNN) will be used. CNN’s typically make good classifiers and perform particular well with image classification tasks due to their feature extraction and classification parts.

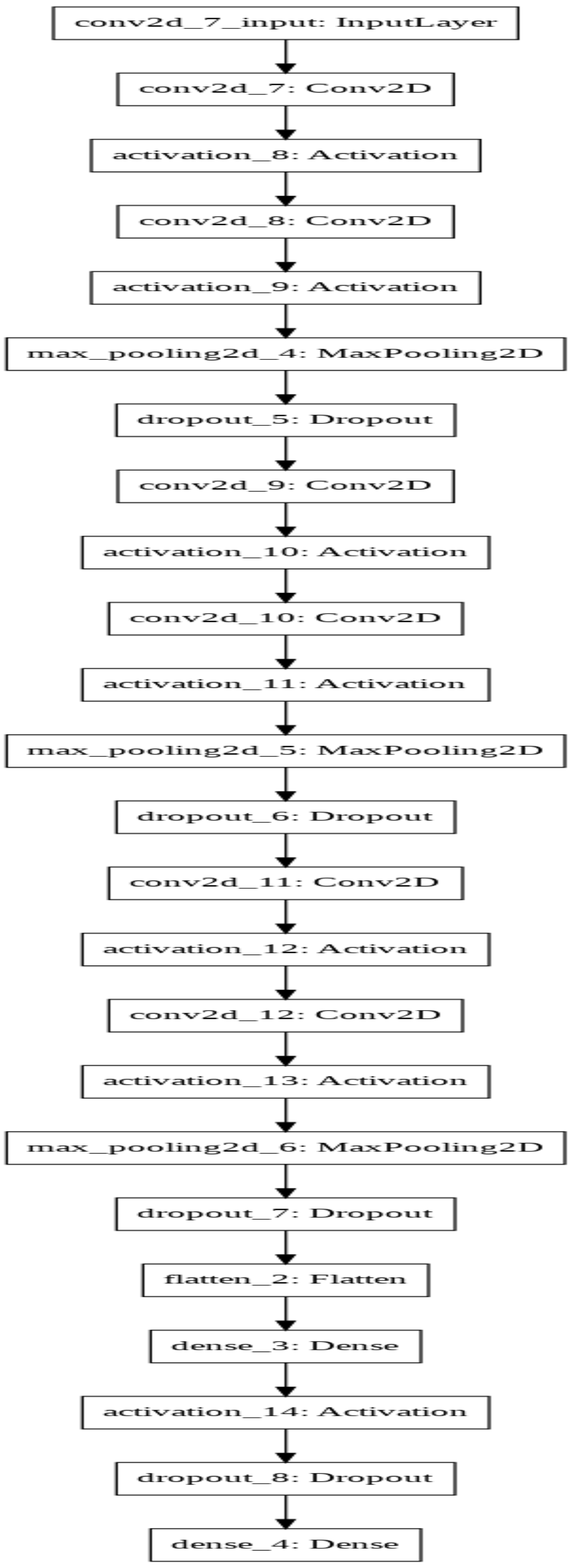
A sequential model will be employed, starting with a simple model architecture, consisting of four Conv2D convolution layers, with our final output layer being a dense layer. Our output layer will have 10 nodes (num_labels) which matches the number of possible classifications.

The model will be trained. As training a CNN can take a significant amount of time, we will start with a low number of epochs and a low batch size. If we can see from the output that the model is converging, we will increase both numbers.

A sequential model will be employed with the following architecture. We shall use 6 convolutional layers of which each successive layer will have an increasing filter density in order to increase the likelihood of extracting the features of each image. We also use a number of pooling and dropout layers so that we could both increase computational efficiency and prevent overfitting.

Below we have a visual representation of the general model architecture we have selected for our image classification approach.

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_7 (Conv2D)            (None, 64, 64, 32)        896       
_________________________________________________________________
activation_8 (Activation)    (None, 64, 64, 32)        0         
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 62, 62, 64)        18496     
_________________________________________________________________
activation_9 (Activation)    (None, 62, 62, 64)        0         
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 31, 31, 64)        0         
_________________________________________________________________
dropout_5 (Dropout)          (None, 31, 31, 64)        0         
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 31, 31, 64)        36928     
_________________________________________________________________
activation_10 (Activation)   (None, 31, 31, 64)        0         
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 29, 29, 64)        36928     
_________________________________________________________________
activation_11 (Activation)   (None, 29, 29, 64)        0         
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 14, 14, 64)        0         
_________________________________________________________________
dropout_6 (Dropout)          (None, 14, 14, 64)        0         
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 14, 14, 128)       73856     
_________________________________________________________________
activation_12 (Activation)   (None, 14, 14, 128)       0         
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 12, 12, 128)       147584    
_________________________________________________________________
activation_13 (Activation)   (None, 12, 12, 128)       0         
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 6, 6, 128)         0         
_________________________________________________________________
dropout_7 (Dropout)          (None, 6, 6, 128)         0         
_________________________________________________________________
flatten_2 (Flatten)          (None, 4608)              0         
_________________________________________________________________
dense_3 (Dense)              (None, 512)               2359808   
_________________________________________________________________
activation_14 (Activation)   (None, 512)               0         
_________________________________________________________________
dropout_8 (Dropout)          (None, 512)               0         
_________________________________________________________________
dense_4 (Dense)              (None, 527)               270351    
=================================================================
Total params: 2,944,847
Trainable params: 2,944,847
Non-trainable params: 0
```



The following will review the accuracy of the model on both the training and test data sets.

5.3. Model Compiling

To compile the model, the configuration should be done on this step.

As far as optimizer, the chosen one was the Adaptive Moment Estimation, i.e. Adam Optimizer. The Adam algorithm is an extension of Stochastic Gradient Descent, i.e. SGD that has seen broader adoption in deep learning applications as it converges really fast and does not suffer from common problems among other optimizers such as Vanishing Learning rate.

The models use as loss function the binary crossentropy. The final output of these models is 527 different probabilities between 0 and 1, for each different file. This is a necessity as we face a multi-class multi-label classification problem, so independent probabilities for each label are needed in order to be able to come to conclusions. As evaluation metrics for the models, accuracy and top k categorical accuracy metrics were used. Accuracy alone cannot be a wise choice and the reason is obvious. There are 527 different labels that the model needs to predict, with different probabilities for each label. The vast majority of the labels are 0's, which means the model will predict correctly almost all of the labels with 0's. As a result, an almost perfect accuracy exists. So, another way to validate the results needs to be used. Top k categorical accuracy computes the accuracy for the first 5 labels that the model predicts for each file. But that assumes that all files have at least 5 labels, which is not true.

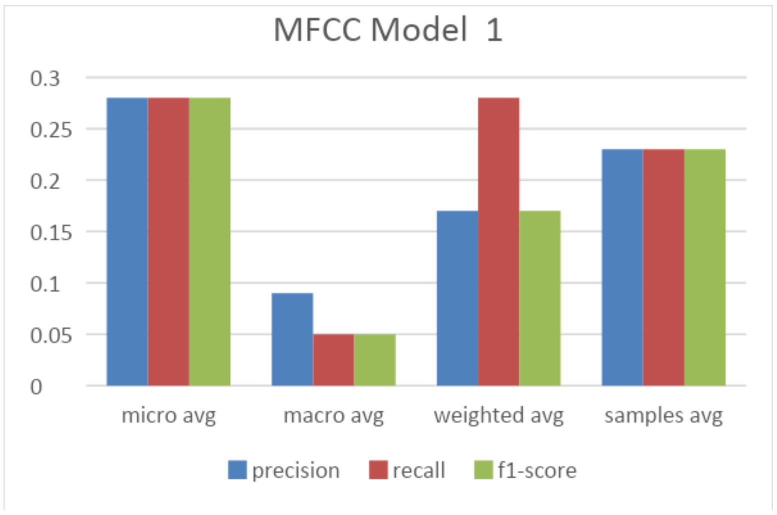
Our second approach bears a number of similarities with our first NN. Both models use the binary crossentropy as a loss function and sigmoid as the final activation layer. In terms of an optimiser RMSProp has been choosen so that we can compare the differences in performance. Additionally we have employed the method of EarlyStopping, by using as a condition the minimum validation accuracy, in order to avoid unnecessary repetitions.

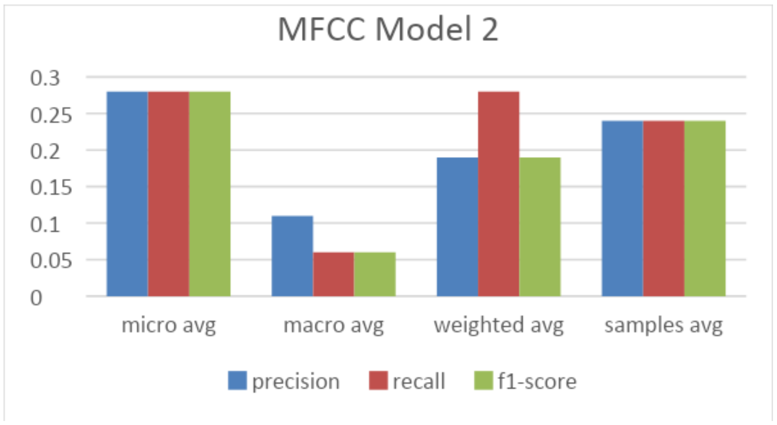
6. Results

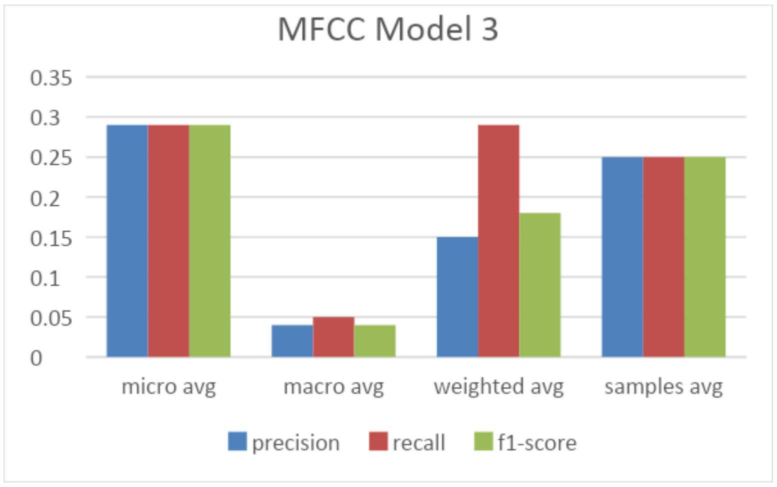
Our first trained model indeed obtained a Training accuracy of 99.56% and a Testing accuracy of 99.53%. The top k categorical accuracy is 25.59% and 21.19% accordingly. But in order to show the realistic result, another way was used. First of all, all predictions were saved in a dataframe. This dataframe has a column with the name of the mp3 file, 527 columns with the different labels having as values the predicted probabilities and one last column with the number of labels that particular file has. Then, with a custom made 'for loop', the predicted probabilities turned to 0's and 1's. The logic behind this is simple. The loop sorts the probabilities for each file and selects the top k probabilities and turns them to 1's and all other predictions are turned to 0's, where k is the number of labels each file has. The result is a dataframe with 0's and 1's for each file.

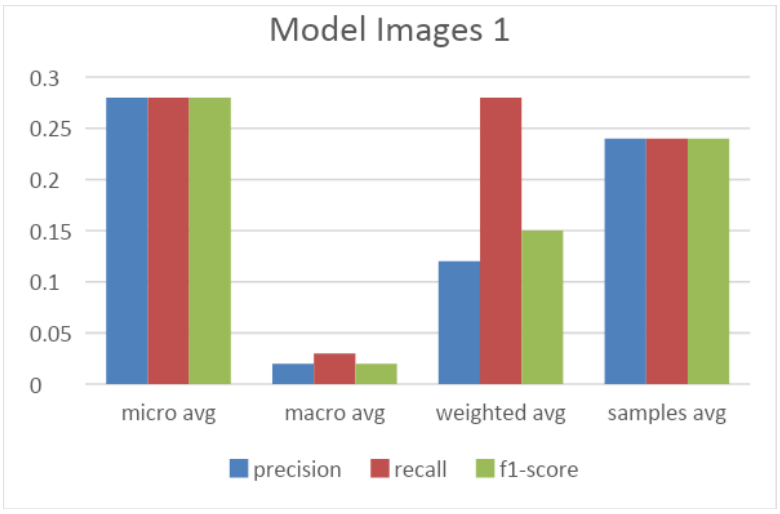
Now things get simpler, as the only thing that needs to be done is to compare the results with the initial values from the train label dataset. The dataframes are transformed to arrays and the function 'multilabel_confusion_matrix' is used. Then a classification_report()[15] of the confusion matrices is shown. This report is consisted of 4 different metrics as columns (precision, recall, f1 score and support) and with 527 rows, each row representing a different label. The numbers show that the model needs more training with new input data or better layer construction. The most interesting metric in this case is the recall, as it shows how well the correct labels are predicted (True positives). There are many labels that are never predicted. However, the model can recognise pretty well speech and music, this could be attributed to the mass representation they have on the training dataset.

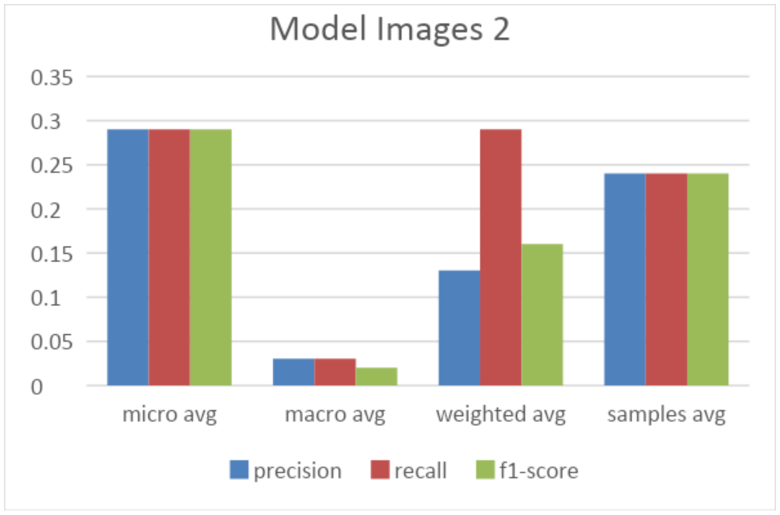
The reported averages include macro average (averaging the unweighted mean per label), weighted average (averaging the support-weighted mean per label), sample average (only for multilabel classification) and micro average (averaging the total true positives, false negatives and false positives) it is only shown for multi-label or multi-class with a subset of classes because it is accuracy otherwise.

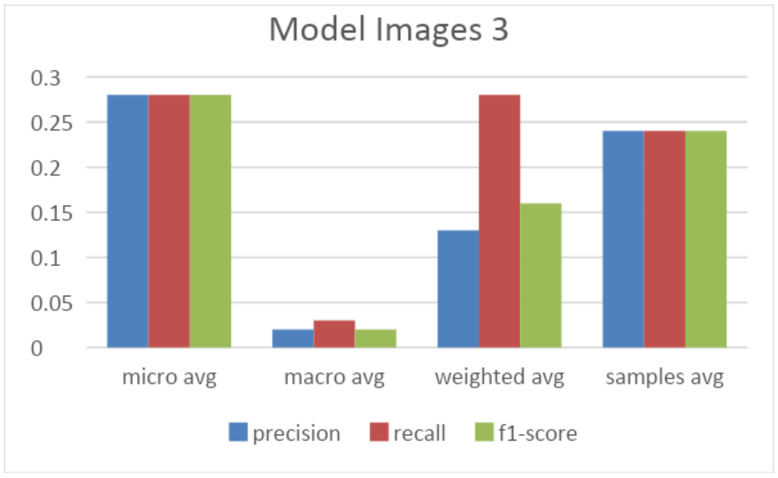

MFCC Model 1
0.3
0.25
0.2
0.15
0.1
0.05
0
micro avg
macro avg
weighted avg
samples avg
precision
recall
f1-score


MFCC Model 2
0.3
0.25
0.2
0.15
0.1
0.05
0
micro avg
macro avg
weighted avg
samples avg
precision
recall
f1-score


MFCC Model 3
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
micro avg
macro avg
weighted avg
samples avg
precision
recall
f1-score







However despite the presented accuracy measure metrics we should note that these results are far from satisfactory. We should note that the output is sparse multi-label table, meaning we have only a few positive labels (1) and a majority of negative labels (0), this will result in an overflatted Keras accuracy metric due to the correctly predicted negative labels.

After researching the issue we have concluded that there are two different approaches that can be followed in order to overcome the misleading metric results. The first option includes using oversampling/undersampling, meaning that we could either generate new samples in the classes which are under-represented or reduce the number of samples in the targeted classes which represent a majority in our dataset. The second and selected approach includes using class weights, meaning that we tell the model to "pay more attention" to samples from under-represented classes.

Unfortunately we were not able to find any officially documented approach to calculate and implement the class weights methodology for multilabel classification problems. This is the reason we have used a user defined function (UDF) in order to calculate the appropriate weights for its class. Furthermore, in order to apply the calculated class weights we have additionally used a UDF which would calculate the adjusted binary crossentropy function by using the class weights as an input.

However, upon using the aforementioned stated methodology we have encountered computational restrictions, in both cases when using our machines and Colab, which would require unacceptable time for the whole implementation.

7. Members / Roles

This team was formed as a result of previous good collaboration experience among the three team members during the master.

After the decision on the dataset to work with, the first face-to-face team meeting took place, where we agreed on our work plan and roles. We decided that we are going to undertake some distinct tasks depending on our project roles and attend weekly meetings in parallel. During these meetings, everyone was responsible of keeping updated each team member regarding his/her progress and share any difficulties or thoughts about the project. In some cases, we decided to work together as a group despite our distinct roles in order everyone to contribute in and follow every part of the assignment and get the relevant knowledge in practice. The project roles were assigned based on each member's experience.

Below the team members with their relevant background are presented:

- Data Engineer (Kotoulas Evangelos)

Evangelos works as a Data Analyst in Analytics team of Propulsion Analytics.

He has studied Economics (BSc) in National and Kapodistrian University of Athens and has a 3-year working experience as an Accountant Assistant in ELTONE S.A. (Dorian Inn Hotel).

His contribution to the project was mainly the downloading and preparation of the data for analysis and helped in EDA and NN construction.

- Data Scientist (Kyritsis Georgios)

George works as a Business Intelligence Engineer at Stor Services Ltd (Storfund).

He holds a Bachelor degree in Finance and Banking from University of Piraeus and has worked as an Assistant Accountant in Eurotel Princess S.A.

His contribution to the project was mainly to research, implement and review the performance of the Neural Networks.

- Business Analyst (Voudris Athanasios)

Thanos works as Business Analyst in Retail Operations department of OPAP S.A.

He holds a 5-year diploma (equivalent to Master) in Civil Engineering from National Technical University of Athens and a Master of Science from Imperial College. He, also, has 2-year working experience as Engineering Consultant in the United Kingdom.

His role in the current project was to analyse Business Case and interpret the results of the neural network analysis.

8. Time Plan

After identifying the project needs, we decided to proceed in scheduling a project plan from the end of the semester lectures until the project deadline. That period was about 8 weeks. We also had to define the major project phases and then tried to make a relevant plan that is presented below:

- Week 1: Business Need Identification
- Week 2: Dataset Search and Exploration
- Week 3: Methodology Searching
- Week 4: Methodology Searching and Model Building
- Week 5: Model Building
- Week 6: Results Interpretation
- Week 7: Report
- Week 8: Presentation

9. Conclusions

This project is a multi-class multi-label classification problem for audio files, provided by Google. Each file is a 10-second .mp3 file with more than 1 labels. The total number of different labels is 527 and each file may have up to 12 different labels. All files were downloaded from YouTube using a script file. To face this classification problem various methods have been tried, using Neural Networks from the Keras library in Python programming language. A first method was to extract the MFCC features from all files available. Taking these features as input (a 1-D table with sequent float values), a sequential model has been constructed with multiple layers. The second method was transforming all files to spectrogram images and implement image classification with CNNs. The model takes as input the dimensions of the images. For both methods the binary crossentropy and sigmoid function have been used. The result of both models is 527 independent probabilities for each file from 0 to 1. Various metrics have been used for validation, because accuracy alone is not an accurate measure. The initial results seem perfect (accuracy>95%) but are misleading. The problem is that the models correctly predict the hundreds true negative labels (0) and that the dataset is pretty unbalanced, meaning that some labels (Music, Speech) appear up to ten times more often than the rest. An approach for this issue is to use label weights in the models. However, the resources for training such a model were not available. In the future we intend to perform more extensive experiments with more data sets and methods. We also intend to perform a comparative experimental study of problem adaptation methods.

10. References

[1]: https://www.analyticsvidhya.com/blog/2017/08/audio-voice-processing-deep-learning/

[2]: https://www.datasciencecentral.com/profiles/blogs/a-quick-start-to-automatic-audio-data-analysis

[3]: https://www.researchgate.net/publication/297683425_Big_Data_Analytics_Challenges_And_Applications_For_Text_Audio_Video_And_Social_Media_Data

[4]:
Radhakrishnan, Ajay Divakaran and Paris Smaragdis, ―AUDIO ANALYSIS FOR SURVEILLANCE APPLICATIONS‖, 2005 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, October 16-19, 2005, New Paltz, NY.

[5]: https://medium.com/@mikesmales/sound-classification-using-deep-learning-8bc2aa1990b7

[6]: https://www.quora.com/What-is-audio-data

[7]: https://pdfs.semanticscholar.org/3b05/d8762acf58a2a55e3520a7ccb84d673f8f7d.pdf

[8]: https://research.google.com/audioset/index.html

[9]: https://github.com/gkoniaris/youtube-downloader

[10]: https://towardsdatascience.com/journey-to-the-center-of-multi-label-classification-384c40229bff

[11]: https://keras.io/

[12]: https://www.pythonforbeginners.com/os/pythons-os-module

[13]: https://medium.com/@mikesmales/sound-classification-using-deep-learning-8bc2aa1990b7

[14]: https://medium.com/@mikesmales/sound-classification-using-deep-learning-8bc2aa1990b7

[15]: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html