"Alexandru Ioan Cuza" University of Iaşi

Master of computational optimization

FACULTY OF COMPUTER SCIENCE

Advanced Software Engineering Techniques 2019 Project

- State of the art -

Audio Tagging

Automatically recognize sounds and apply tags of varying natures

Proposed by: Cojocaru Gabriel-Codrin
Dinu Sergiu Andrei
Luncaşu Bogdan Cristian
Racoviţă Mădălina-Alina
Vîntur Cristian

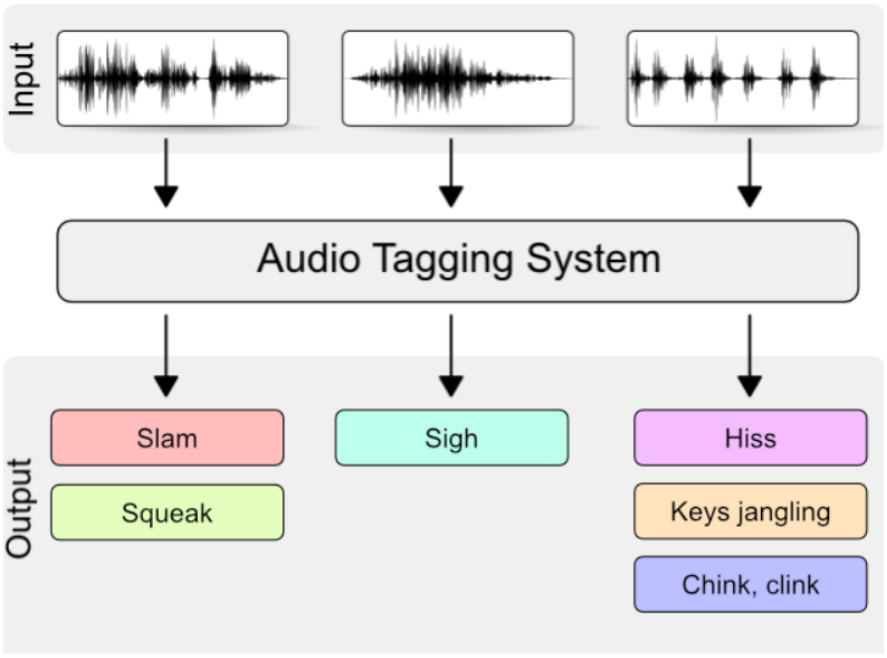
Scientific coordinators: PhD Associate Professor Adrian Iftene
PhD Associate Professor Mihaela Elena Breaban

Contents

I. Introduction

I.1. Competition's description

One year ago, **Freesound** and **Google's Machine Perception** hosted an audio tagging competition challenging Kagglers to build a general-purpose auto tagging system. This year they're back and taking the challenge to the next level with multi-label audio tagging, doubled number of audio categories, and a noisier than ever training set. [1]



Here's the background: **Some sounds are distinct and instantly recognizable, like a baby's laugh or the strum of a guitar**. Other sounds are difficult to pinpoint. If you close your eyes, could you tell the difference between the sound of a chainsaw and the sound of a blender?

Because of **the vastness of sounds we experience**, *no reliable automatic general-purpose audio tagging systems exist.* A significant amount of manual effort goes into tasks like annotating sound collections and providing captions for non-speech events in audiovisual content.

To tackle this problem, Freesound (an initiative by MTG-UPF that maintains a collaborative database with over 400,000 Creative Commons Licensed sounds) and Google Research's Machine Perception Team (creators of AudioSet, a large-scale dataset of manually annotated audio events with over 500 classes) have teamed up to develop the dataset for this new competition.

To win this competition, Kagglers will develop an **algorithm to tag audio data automatically** using a diverse vocabulary of 80 categories.

If successful, these systems could be used for several applications, ranging from **automatic labelling of sound collections** to **the development of systems that automatically tag video content** or recognize sound events happening in real time.

I.2. Motivation

Current machine learning techniques require large and varied datasets in order to provide good performance and generalization. However, manually labelling a dataset is time-consuming, which limits its size. Websites like Freesound or Flickr host large volumes of user-contributed audio and metadata, and labels can be inferred automatically from the metadata and/or making predictions with pre-trained models. Nevertheless, these automatically inferred labels might include a substantial level of label noise.

The main research question addressed in this competition is **how to adequately exploit a small amount of reliable, manually-labeled data, and a larger quantity of noisy web audio data in a multi-label audio tagging task with a large vocabulary setting**. In addition, since the data comes from different sources, the task encourages domain adaptation approaches to deal with a potential domain mismatch.

I.3. Timeline

- **June 3, 2019 11:59 PM UTC** - Entry deadline. You must accept the competition rules before this date in order to compete.
- **June 3, 2019 11:59 PM UTC** - Team Merger deadline. This is the last day participants may join or merge teams.
- **June 11, 2019 11:59 AM UTC** - Final submission deadline.

II. Dataset description

II.1. Audio Dataset

The dataset used in this challenge is called **FSDKaggle2019**, and it employs audio clips from the following sources:

- Freesound Dataset (**FSD**): a dataset being collected at the **MTG-UPF** based on **Freesound** content organized with the AudioSet Ontology
- The soundtracks of a pool of Flickr videos taken from the **Yahoo Flickr Creative Commons 100M dataset (YFCC)**

The audio data is labeled using a vocabulary of 80 labels from **Google's AudioSet Ontology** [2], covering diverse topics: Guitar and other Musical instruments, Percussion, Water, Digestive, Respiratory sounds, Human voice, Human locomotion, Hands, Human group actions, Insect, Domestic animals, Glass, Liquid, Motor vehicle (road), Mechanisms, Doors, and a variety of Domestic sounds.

II.2. Ground Truth Labels

The ground truth labels are provided at the clip-level, and express the presence of a sound category in the audio clip, hence can be considered weak labels or tags. Audio clips have variable lengths (roughly from 0.3 to 30s, see more details below).

The audio content from **FSD** has been **manually labeled** by humans following a data labeling process using the **Freesound Annotator platform**. Most labels have inter-annotator agreement but not all of them. More details about the data labeling process and the Freesound Annotator can be found in [3].

The **YFCC soundtracks** were labeled using automated heuristics applied to the audio content and metadata of the original Flickr clips. Hence, a substantial amount of label noise can be expected. The label noise can vary widely in amount and type depending on the category, including in- and out-of-vocabulary noises. More information about some of the types of label noise that can be encountered is available in [4].

II.3. Format and License

All clips are provided as uncompressed PCM 16 bit, 44.1 kHz, mono audio files. All clips used in this competition are released under Creative Commons (CC) licenses, some of them requiring attribution to their original authors and some forbidding further commercial reuse. In order to be able to comply with the CC licenses terms, a full list of audio clips with their associated licenses and a reference to the original content (in Freesound or Flickr) will be published at the end of the competition. Until then, the provided audio files can only be used for the sole purpose of participating in the competition.

II.4. Train set

The train set is meant to be for system development. The idea is to limit the supervision provided (i.e., the manually-labeled data), thus promoting approaches to deal with label noise. The train set is composed of two subsets as follows:

II.4.1. Curated subset

The curated subset is a small set of manually-labeled data from FSD.

- **Number of clips/class**: 75 except in a few cases (where there are less)
- **Total number of clips**: 4970
- **Avge number of labels/clip**: 1.2
- **Total duration**: 10.5 hours

The duration of the audio clips ranges from 0.3 to 30s due to the diversity of the sound categories and the preferences of Freesound users when recording/uploading sounds. It can happen that a few of these audio clips present additional acoustic material beyond the provided ground truth label(s).

II.4.2. Noisy subset

The noisy subset is a larger set of noisy web audio data from Flickr videos taken from the **YFCC dataset** [6].

- **Number of clips/class**: 300
- **Total number of clips**: 19815
- **Avge number of labels/clip**: 1.2

- **Total duration**: 80 hours

The duration of the audio clips ranges from 1s to 15s, with the vast majority lasting 15s.

Considering the numbers above, per-class data distribution available for training is, for most of the classes, 300 clips from the noisy subset and 75 clips from the curated subset, which means 80

II.5. Test set

The test set is used for system evaluation and consists of manually-labeled data from FSD. Since most of the train data come from YFCC, some acoustic domain mismatch between the train and test set can be expected. All the acoustic material present in the test set is labeled, except human error, considering the vocabulary of 80 classes used in the competition.

The test set is split into two subsets, for the public and private leaderboards. In this competition, the submission is to be made through Kaggle Kernels. Only the test subset corresponding to the public leaderboard is provided (without ground truth).

II.6. Files

The structure of the data	
train_curated.csv	ground truth labels for the curated subset of the training audio files (see Data Fields below)
train_noisy.csv	ground truth labels for the noisy subset of the training audio files (see Data Fields below)
sample_submission.csv	a sample submission file in the correct format, including the correct sorting of the sound categories; it contains the list of audio files found in the test.zip folder (corresponding to the public leaderboard)
train_curated.zip	a folder containing the audio (.wav) training files of the curated subset
train_noisy.zip	a folder containing the audio (.wav) training files of the noisy subset
test.zip	a folder containing the audio (.wav) test files for the public leaderboard

II.7. Columns

Each row of the train_curated.csv and train_noisy.csv files contains the following information:

- **fname**: the audio file name, eg, 0006ae4e.wav
- **labels**: the audio classification label(s) (ground truth). Note that the number of labels per clip can be one, eg, Bark or more, eg, "Walk_and_footsteps,Slam".

II.8. Dataset problems

Detected corrupted files in the curated train set The following 5 audio files in the curated train set have a wrong label, due to a bug in the file renaming process: **f76181c4.wav**, **77b925c2.wav**, **6a1f682a.wav**, **c7db12aa.wav**, **7752cc8a.wav**

The audio file **1d44b0bd.wav** in the curated train set was found to be corrupted (contains no signal) due to an error in format conversion.

III. Sound classification predictive models

III.1. Models brief description from different Kaggle approaches

Kaggle [9] is an online community of data scientists and machine learners. Since Kaggle allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges, it **can be used also as a source of inspiration when solving different tasks machine learning related**. We extracted some of the predictive models that we have found in the public kernels that were attached to the subject competition.

III.1.1. Bidirectional LSTM for audio labeling with Keras

Long short-term memory (LSTM) is an artificial **recurrent neural network (RNN)** architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as **unsegmented, connected handwriting recognition or speech recognition**. **LSTM** in its core, *preserves information from inputs that has already passed through it using the hidden state.*

Unidirectional LSTM only **preserves information of the past** because the only inputs it has seen are from the past. Using **bidirectional** will **run your inputs in two ways, one from past to future and one from future to past** and what differs this approach from unidirectional is that **in the LSTM that runs backwards you preserve information from the future** and using the two hidden states combined you are able in any point in time to preserve information from both past and future.

What they are suited for is a very complicated question but BiLSTMs show very good results as they can understand context better, but this can be explain through a simple example. Lets

say we try to predict the next word in a sentence, on a high level what a unidirectional LSTM will see is:

```
The boys went to ....
```

And will try to predict the next word only by this context, with bidirectional LSTM you will be able to see information further down the road for example

```
# Forward LSTM:

The boys went to ...

# Backward LSTM:

... and then they got out of the pool
```

You can see that using the information from the future it could be easier for the network to understand what the next word is.

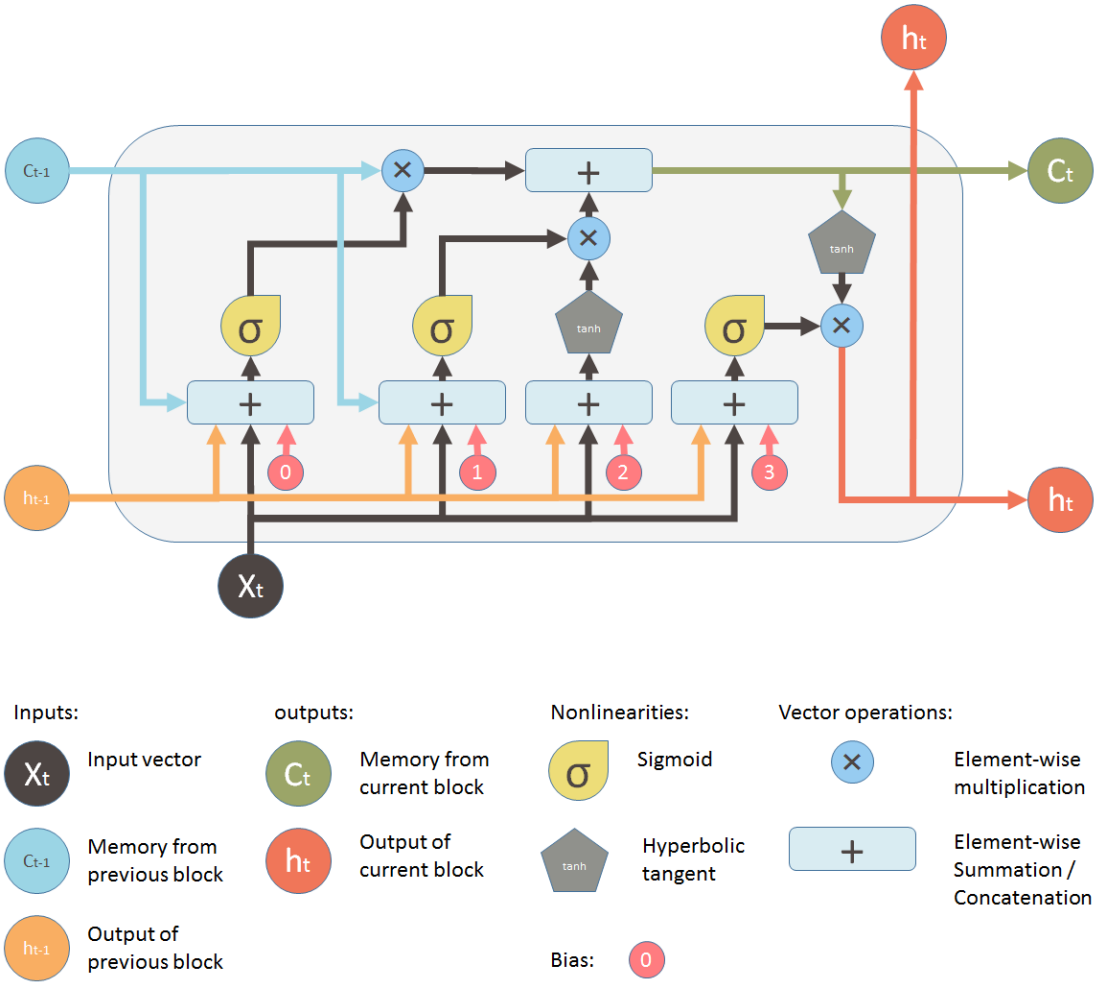


Fig. III.1.1.a: *Overview LSTM*

In a public Kaggle kernel [8] it was used the following architecture for the **Freesound classification problem** inspired from an article named *Convolutional RNN: an Enhanced Model for Extracting Features from Sequential Data* [11].

```
# Neural network model
input_shape = (636,128)
optimizer = Adam(0.005, beta_1=0.1, beta_2=0.001, amsgrad=True)
n_classes = 80

model = Sequential()
model.add(Bidirectional(CuDNNLSTM(256, return_sequences=True),
          input_shape=input_shape))
model.add(Attention(636))
model.add(Dropout(0.2))
model.add(Dense(400))
model.add(ELU())
model.add(Dropout(0.2))
model.add(Dense(n_classes, activation='softmax'))
model.compile(loss='categorical_crossentropy',
              optimizer=optimizer,
              metrics=['acc'])
```

Another likely model that develops a bird sound classification problem and uses bidirectional LSTM neural networks had the following architecture:

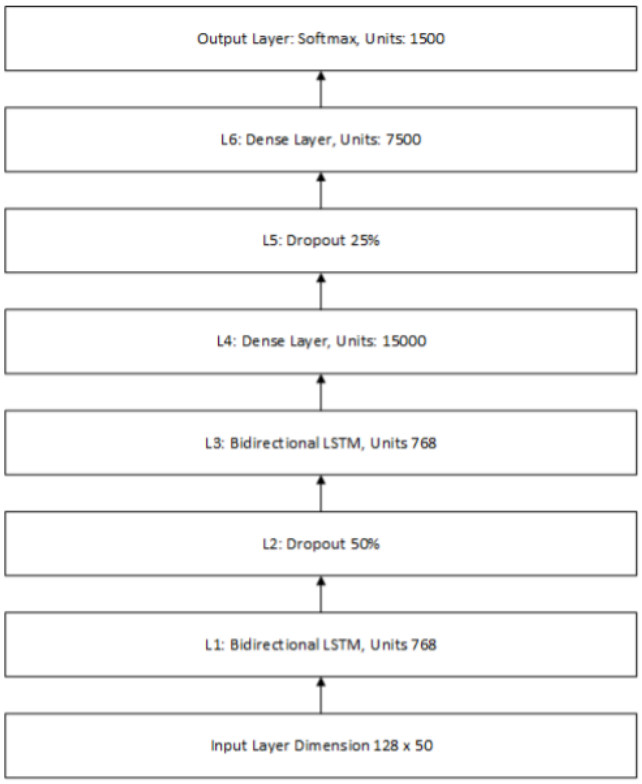


Fig. III.1.1: *Architecture bidirectional LSTM. [10]*

III.2. Evaluation metric

The task consists of **predicting the audio labels (tags) for every test clip**. Some test clips bear one label while others bear several labels. The predictions are to be done at the clip level, i.e., no start/end timestamps for the sound events are required.

The primary competition metric will be **label-weighted label-ranking average precision** [7](lwlrap, pronounced "Lol wrap"). This **measures the average precision of retrieving a ranked list of relevant labels** for each test clip (i.e., the system ranks all the available labels, then the precisions of the ranked lists down to each true label are averaged). This is a generalization of the mean reciprocal rank measure for the case where there can be multiple true labels per test item. The novel "label-weighted" part means that the overall score is the average over all the labels in the test set, where each label receives equal weight (by contrast, plain lrap gives each test item equal weight, thereby discounting the contribution of individual labels when they appear on the same item as multiple other labels).

The formula for label-ranking average precision (LRAP) is as follows:

$$LRAP(y, \widehat{f}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} \frac{1}{\|y_i\|_0} \sum_{j:y_{ij}=1} \frac{|\mathcal{L}_{ij}|}{rank_{ij}} \tag{1}$$

```
def calculate_overall_lwlrap_sklearn(truth, scores):
    """Calculate the overall lwlrap using sklearn.metrics.lrap."""
    # sklearn doesn't correctly apply weighting to samples with no labels
    # so just skip them.
    sample_weight = np.sum(truth > 0, axis=1)
    nonzero_weight_sample_indices = np.flatnonzero(sample_weight > 0)
    overall_lwlrap = label_ranking_average_precision_score(
        truth[nonzero_weight_sample_indices, :] > 0,
        scores[nonzero_weight_sample_indices, :],
        sample_weight=sample_weight[nonzero_weight_sample_indices])
    return overall_lwlrap
```

IV. State of the art

IV.1. Problem description

The environmental sound classification problem can come in many different shapes, from having to classify short audio files with a label from a specified set to having to tag live audio streaming with one or multiple labels.

Automatic environmental sound classification or tagging is a growing area of research. Work done in this area is comparatively scarce with work done in related audio fields such as speech and music. Applications are numerous and include multimedia indexing and retrieval, environmental sounds subtitles, assisting deaf individuals or even monitoring illegal deforestation. A very interesting project regarding this last use case is created by a non-profit organization called Delta Analytics. They helped build a system where a lot of old mobile phones are attached to trees in rain forests which listen to chainsaw noises. Their role is to identify when a chainsaw is being used and alert rangers who can stop illegal deforestation [13].

Because of the recent success in image classification, the question that raises is: can we bring techniques used in image classification to sound classification by representing sound in various image formats?

IV.2. Short audio files classification

IV.2.1. Problem statement

Classify environmental sounds with focus on identification of particular urban sounds. Given an audio sample of a few seconds duration determine if it contains one of the target urban sounds [12].

IV.2.2. Dataset description

The dataset is called Urbansound8K and contains 8732 sound excerpts (under 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

By taking a closer look to the dataset, the following observation can be made: it's tricky to visualize the difference between some sounds, especially the continuous ones like jackhammer and engine idling.

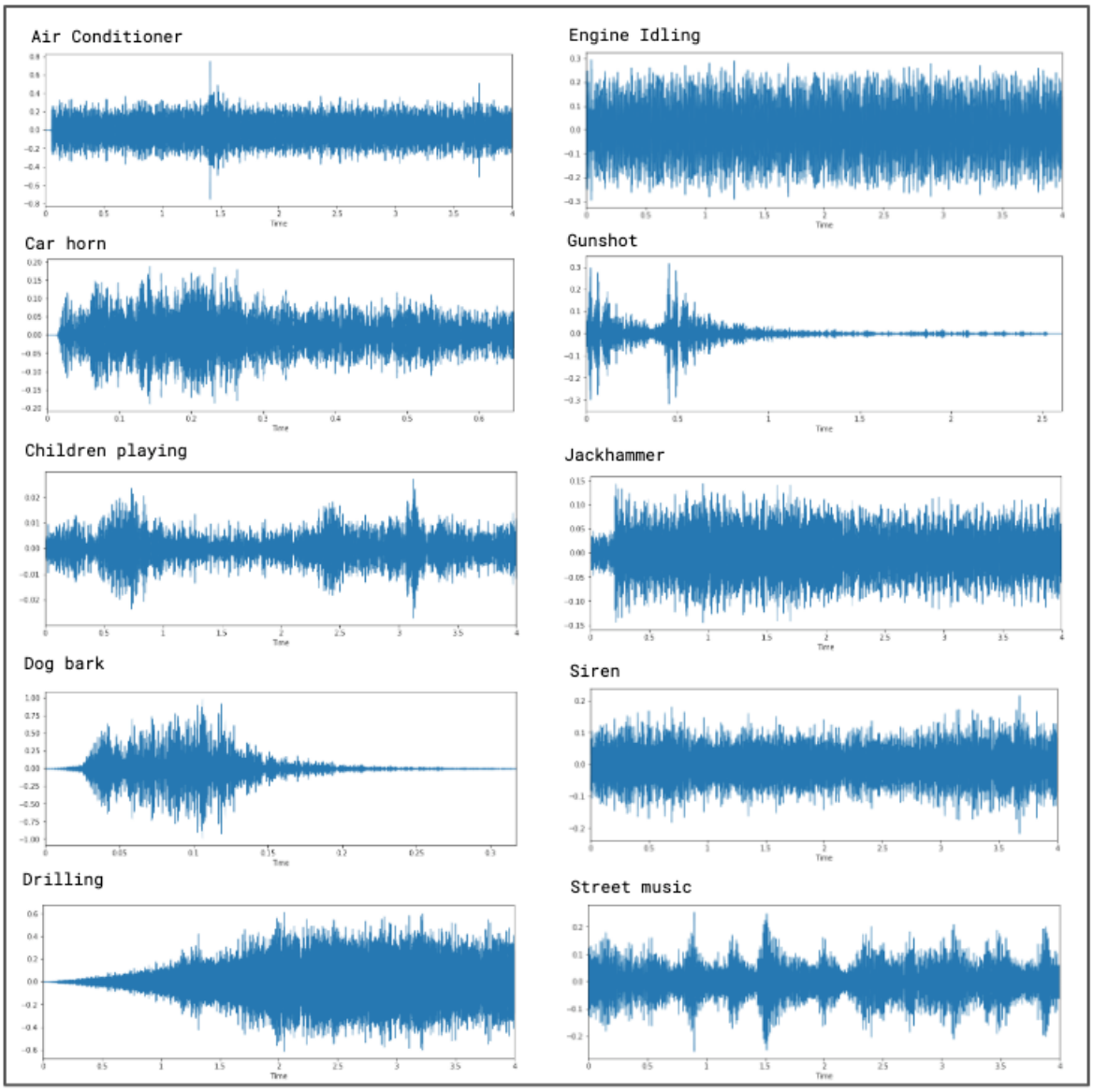


Fig. : *Visualization of sound amplitude with respect to time*

A deeper dive into the data show that: most samples have 2 audio channels with a few having just 1 channel, sample rate varies from 8kHz to 48kHZ and bit depth also varies from 4bit to 32bit which leads to requiring a data normalization step.

IV.2.3. Proposed solution

The proposed solution involves mapping the audio files, after the normalization step, to a visual representation. Spectrograms are a useful technique for visualizing the spectrum of frequencies of a sound and how they vary during a very short period of time. A similar technique known as mel-spectrogram will be used for this case. The main difference is that a spectrogram uses a linear spaced frequency scale (so each frequency bin is spaced an equal number of Hertz apart), whereas a mel-spectrogram uses a quasi-logarithmic spaced frequency scale, which is more similar to how the human auditory system processes sounds.

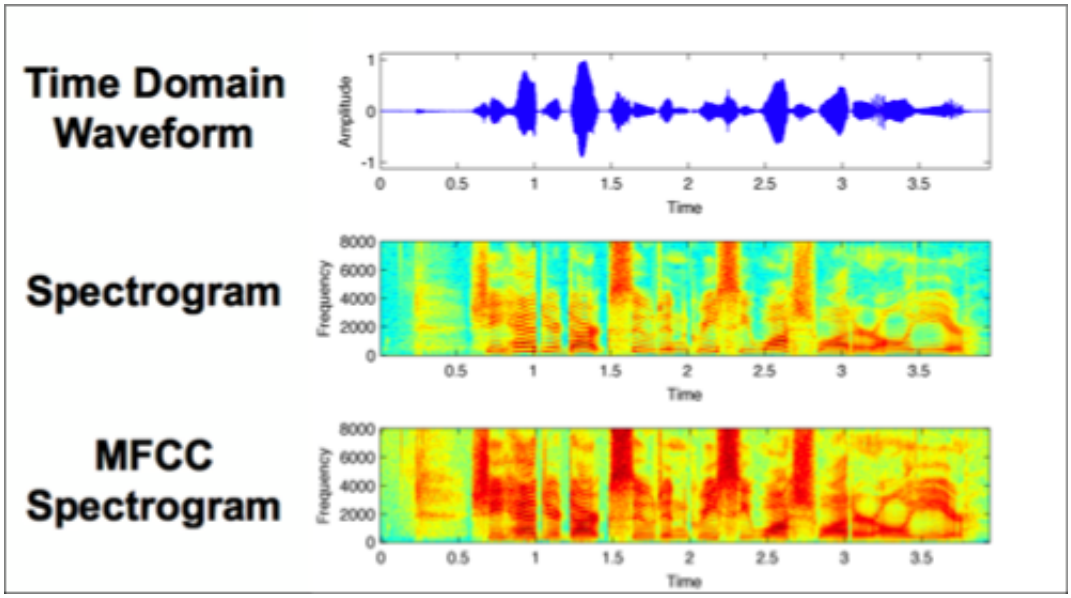


Fig. : *Spectrogram and mel-spectrogram*

The problem will be solved by using deep learning on the visual representation of the audio file. In this case a Convolutional Neural Network (CNN) will be used. The model used is a sequantial one consisting of 4 Conv2D convolution layers with the final output being a dense layer. The output layer will have 10 nodes which matches the number of possible classifications.

IV.2.4 Results

The trained model obtained a Training accuracy of 98.19% and a Testing accuracy of 91.92%.

IV.3. Tagging longer audio files

By providing a solution to solving the short audio files tagging problem we can extend it to work on longer audio files. The idea behind this extension is to divide the audio into smaller parts, transform these parts into mel-spectrograms and feed them into a more complex network that will be able to provide the correct label(s).

IV.4. State of the art solution

State of the art solution to solving this problem is deep learning. By dividing the audio and transforming it into mel-spectrograms [14] we can use some of the success of image classification and bring it to audio classification. Just applying image classification on mel-spectrograms doesn't work because sound in form of mel-spectrograms doesn't have the same properties as an image but by adapting the network to the new context we can obtain a solution to solving this problem.

IV.5. Relevant articles

- AUDIO TAGGING WITH NOISY LABELS AND MINIMAL SUPERVISION - *https://arxiv.org/pdf/1906.02975.pdf*
- Convolutional RNN: an Enhanced Model for Extracting Features from Sequential Data - *https://arxiv.org/pdf/1602.05875v3.pdf*

IV.6. Relevant links

- Kaggle competition 1st place solution - *https://github.com/lRomul/argus-freesound*
- Kaggle competition 2nd place solution - *https://www.kaggle.com/c/freesound-audio-tagging-2019/discussion/97815*
- Kaggle competition 3rd place solution - *https://www.kaggle.com/c/freesound-audio-tagging-2019/discussion/97926*

V. Application prototype

V.1. Use cases

- Semi-automatized System:
  - can use reinforcement learning for improving tagging accuracy
- Signals that may help deaf users in:
  - dangerous situations (outdoor activities/ fire alarms)
  - daily activities
  - movie captioning
    - different details like gun shooting in background may help deaf users having a better entertainment experience while watching movies
- video indexing for search engines
- security (e.g. detecting breaking in sounds, hearth attack/ people suffering, etc.)
- analyzing audience behavior(excitement, boredom, etc.)

V.2. Model integration

Based on the business needs the following approaches may be applied:

Use model on a remote server and expose it via:

- remote REST API
  - can process full media file
  - disadvantage: harder to infer in a live stream
- remote streaming data (e.g. sending data over websockets)

Export the model to be used on an end device

- for tensorflow, the models can be converted to either tflite for Android or coreML for iOS

V.3. Application flow

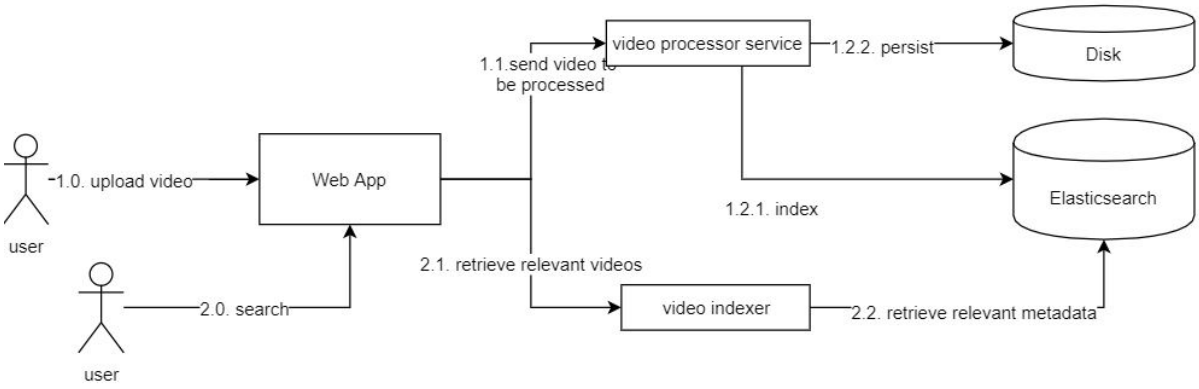


Fig. V.3.1: *Application flow overview*

Video processing flow

1. Upload video via web app
2. Send video to processor service
3. Index extracted metadata into ElasticSearch
4. Persist video in a blob/ on disk

Video indexer flow

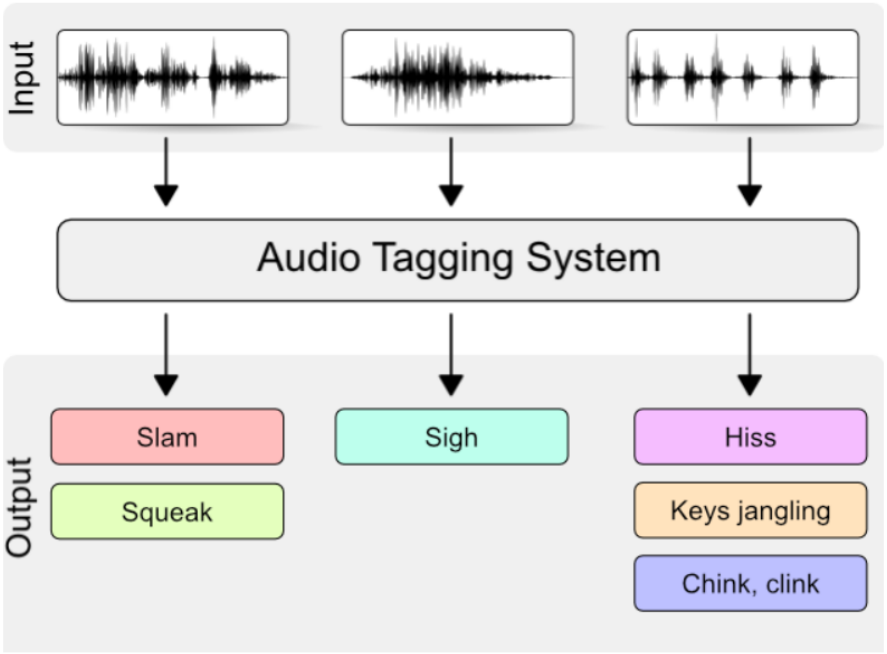
1. Search for videos using certain keywords
2. Retrieve relevant metadata from ElasticSearch
3. Construct relevant list of videos that will be send to the user

VI. Risk assessment

Risk type	Risk details	Likelihood	Severity	Mitigation measures
Requirements	Requirements will be added in the middle of the project	Probable	Moderate	Brainstorm as many variants of the application as possible and keep the best one
Schedule	Not all components will be done in time	Possible	Serious	Evaluate how difficult each component will be and choose only those that can be finished
Performance	The classification model will perform poorly	Possible	Serious / Major	Read from various sources of what other did
Organizational	Bad distribution of tasks. Two people doing the same thing independently	Unlikely	Moderate	Organizing tasks on *trello* and occasional meetings to discuss the current state of the project
System architecture	Choose the wrong architecture for the application	Possible	Serious	Read other architectures used in solving the same problem and create something similar
Resources	Personal laptops not powerful enough to train the model	Possible / Probable	Major	Consider using cloud solutions to train the model

VII. Technical report of Clink application

VII.1. Problem presentation



Some sounds are distinct and instantly recognizable, like a baby's laugh or the strum of a guitar. Other sounds are difficult to pinpoint. If you close your eyes, could you tell the difference between the sound of a chainsaw and the sound of a blender?

Because of **the vastness of sounds we experience**, *no reliable automatic general-purpose audio tagging systems exist.* A significant amount of manual effort goes into tasks like annotating sound collections and providing captions for non-speech events in audiovisual content.

To tackle this problem, Freesound (an initiative by MTG-UPF that maintains a collaborative database with over 400,000 Creative Commons Licensed sounds) and Google Research's Machine Perception Team (creators of AudioSet, a large-scale dataset of manually annotated audio events with over 500 classes) have teamed up to develop the dataset for a new Kaggle

competition.

To win this competition, Kagglers will develop an **algorithm to tag audio data automatically** using a diverse vocabulary of 80 categories.

If successful, these systems could be used for several applications, ranging from **automatic labelling of sound collections** to **the development of systems that automatically tag video content** or recognize sound events happening in real time.

VII.1.1. Our motivation

Current machine learning techniques require large and varied datasets in order to provide good performance and generalization. However, manually labelling a dataset is time-consuming, which limits its size. The main research question addressed in this competition is **how to adequately exploit a small amount of reliable, manually-labeled data, and a larger quantity of noisy web audio data in a multi-label audio tagging task with a large vocabulary setting**. In addition, since the data comes from different sources, the task encourages domain adaptation approaches to deal with a potential domain mismatch.

VII.2. The summary of the state-of-the-art

The environmental sound classification problem can come in many different shapes, from having to classify short audio files with a label from a specified set to having to tag live audio streaming with one or multiple labels.

Automatic environmental sound classification or tagging is a growing area of research. Work done in this area is comparatively scarce with work done in related audio fields such as speech and music. Applications are numerous and include multimedia indexing and retrieval, environmental sounds subtitles, assisting deaf individuals or even monitoring illegal deforestation. A very interesting project regarding this last use case is created by a non-profit organization called Delta Analytics. They helped build a system where a lot of old mobile phones are attached to trees in rain forests which listen to chainsaw noises. Their role is to identify when a chainsaw is being used and alert rangers who can stop illegal deforestation [13].

Because of the recent success in image classification, the question that raises is: can we bring techniques used in image classification to sound classification by representing sound in various image formats?

State of the art solution to solving this problem is deep learning. By dividing the audio and transforming it into spectrograms we can use some of the success of image classification and bring it to audio classification. Just applying image classification on spectrograms doesn't

work because sound in form of spectrograms doesn't have the same properties as an image but by adapting the network to the new context we can obtain a solution to solving this problem.

VII.3. Proposed solution - CLINK application

The **Clink application** is an **audio indexer** who's interface is attached bellow, that **allows the upload of an audio file and returns the tags corresponding to the sound extracted in that audio file**. As well this application **allows the filtration by tag name of a set of audio files persisted in a local database**. By doing this, in the user interface will be displayed a couple of audio files if any corresponding to that tag.

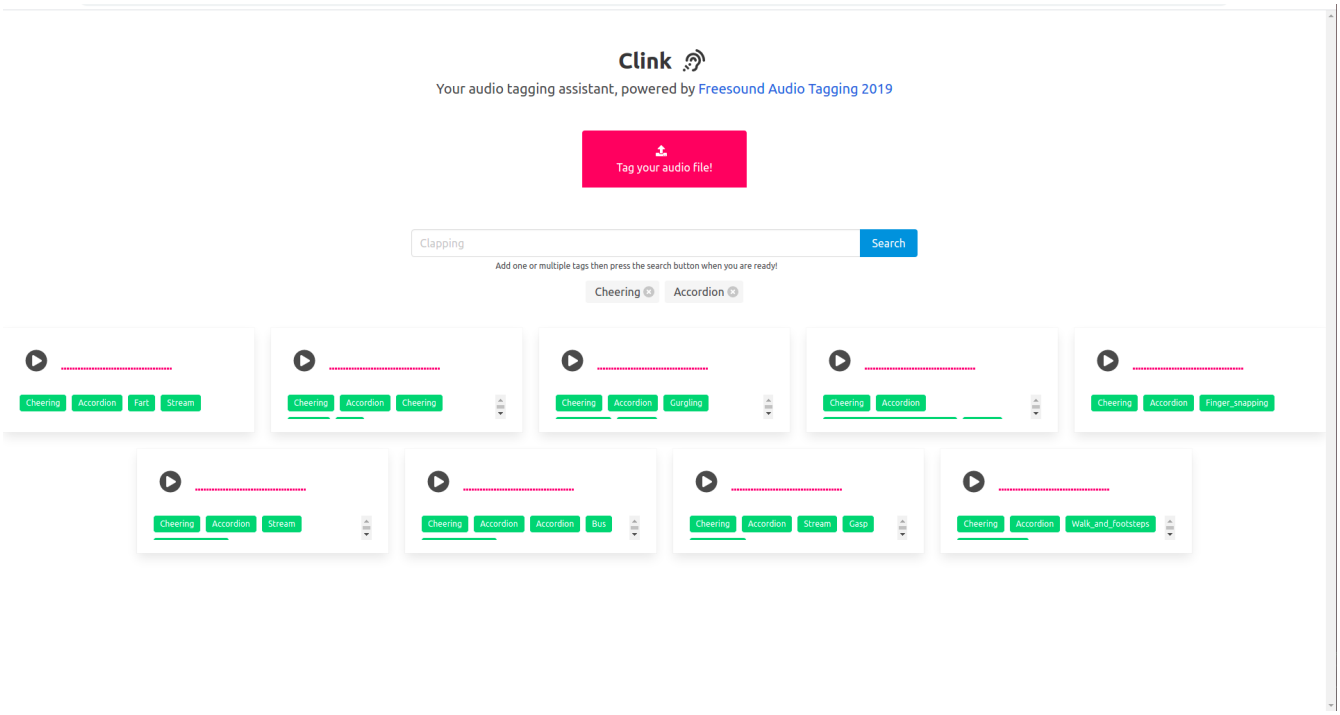


Fig. VII.3.1.: *User uploads an audio file*

To be more specific, one of the principal use cases of the Clink application is illustrated in the following use-case diagram. First of all the user uploads from its personal computer an audio file.The content of the audio file is sent encoded Base64 to the back-end server. The back-end server will decode the content of the file and will call a function from the ML component which is responsible with spliting the audio file in fragments of the same size. Each audio fragment is going to be processed by translating the audio sequences into spectograms. From here the pre-trained machine learning model is going to be called. When the execution is finished, a list of tags will be returned, list that is going to be collected by the front-end server as a response and after that rendered in a user-friendly manner into the interface.

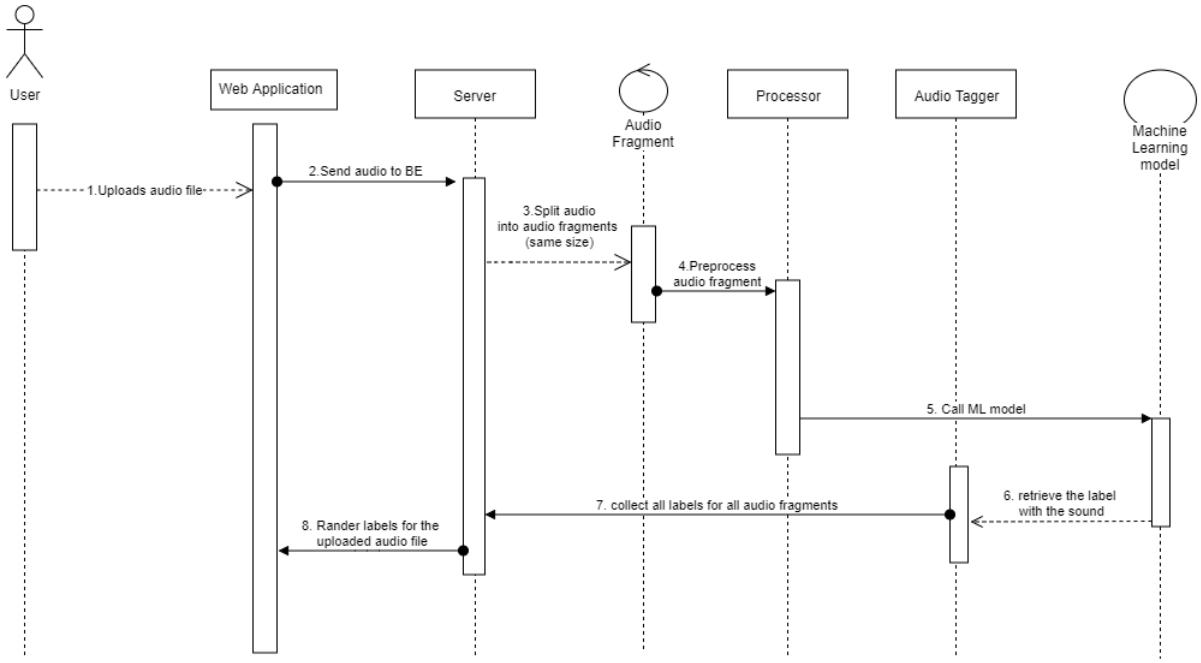


Fig. VII.3.2.: *Clink application*

The response for a search call for the **'Bark'** tag for example will look like this:


```
{
    "metadata": [
        {
            "id": 1,
            "description": null,
            "labels": [
                "Bark"
            ]
        },
        {
            "id": 2,
            "description": "description",
            "labels": [
                "Bark"
            ]
        }
    ]
}
```


Fig. VII.3.3.: *Api reponse for 'Bark' tag*

Regarding **technical implementation details**, the front-ned was developed in nodejs and the backend in **Python 3.7** using a **Flask API** along with an **ORM** developed in **SQLAlchemy**. The database in which the data is stored was **Postgres**.

The database was necessary for persisting the information; for instance, since the application allows the searching based on a specific tag name, it is required to have in the database a list of audio file names mapped to their corresponding tags. This was done using association tables, which illustrate relationships of many-to-many. The audio files content incoded into base64 is as well persisted into the databes, along with all the labels that can be detected by the ML

component.

To get into more details concerning the **Machine Learning implementation**, ***the proposed solution involves mapping the audio files, after the normalization step, to a visual representation***. Spectrograms are a useful technique for visualizing the spectrum of frequencies of a sound and how they vary during a very short period of time. A similar technique known as mel-spectrogram can be used for this case. The main difference is that a spectrogram uses a linear spaced frequency scale (so each frequency bin is spaced an equal number of Hertz apart), whereas a mel-spectrogram uses a quasi-logarithmic spaced frequency scale, which is more similar to how the human auditory system processes sounds. **We used spectograms, which we further divided into windows with the help of which we contructed our train datasets**.

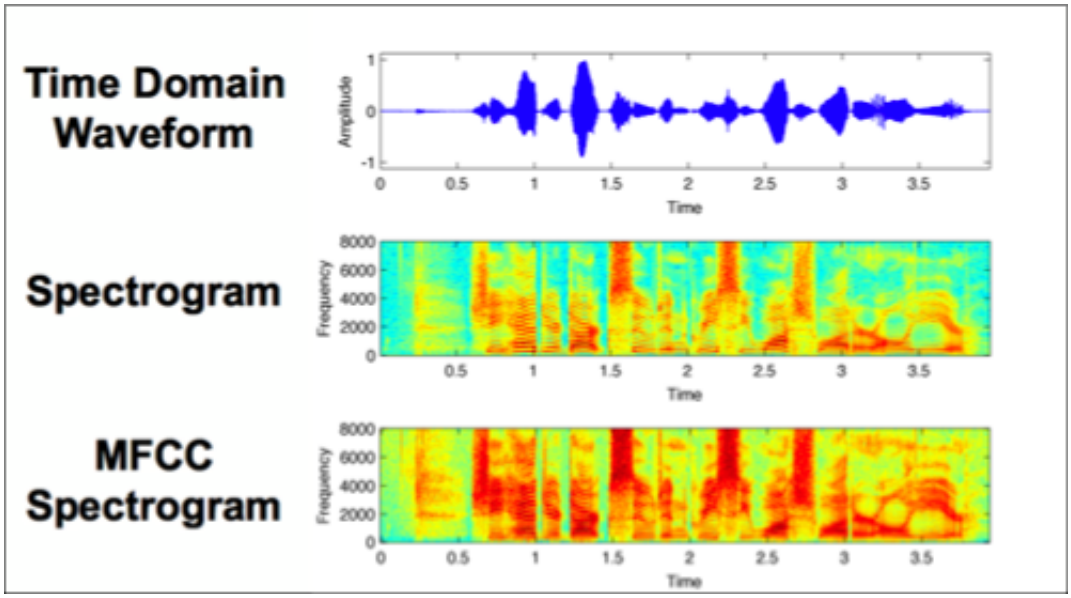


Fig. VII.3.4.: *Spectrogram and mel-spectrogram*

VII.4. Results / Evaluation

For the classification of a given sound as belonging to into one or more classes we managed to have different attempts with several models of pre-trained neural networks. The models we used in our experiments are: VGG16, VGG19, ResNet50, Resnet50V2, InceptionV3, Inception-ResNetV2, MobileNet and MobileNetV2. The best results were obtained by VGG16, VGG19 and ResNet50. In the next phase, we will apply a genetic algorithm to choose the subset of the models mentioned above in order to optimize the accuracy of the model set. The result of our model will consist in the average of the results of the models in the subset mentioned above.

ResNet-50 is a convolutional neural network that is trained on more than a million images from the ImageNet database. The network is 50 layers deep and can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images.

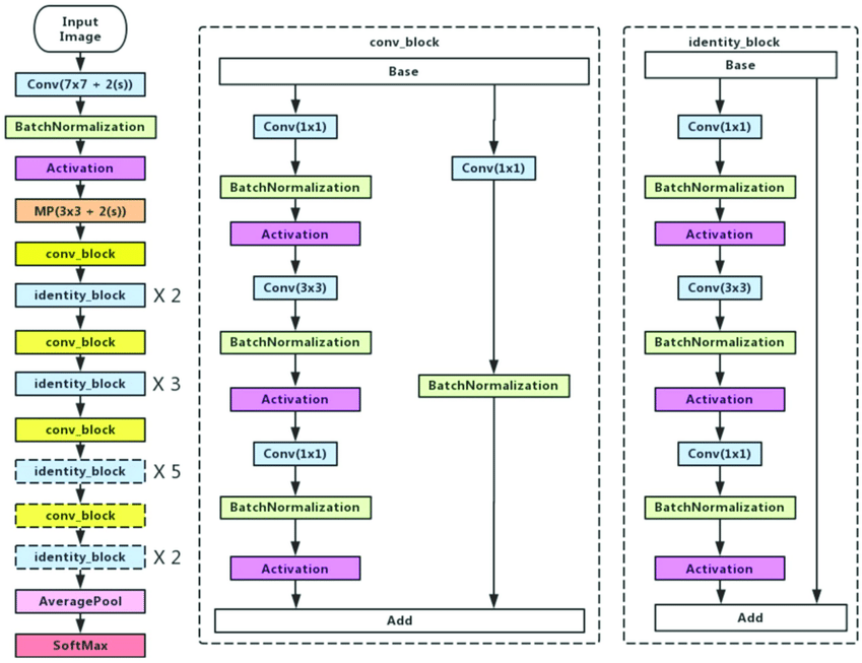


Fig. VII.4.1.: *Resnet50 architecture*

VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". The model achieves 92.7% top-5 test accuracy in ImageNet, previously mentioned, which is a dataset of over 14 million images belonging to 1000 classes. This is why we thought that by using this pretrained model with transfer learning techniques the accuracy will be increased comparing with the previous models. The architecture of this neural network is illustrated bellow.

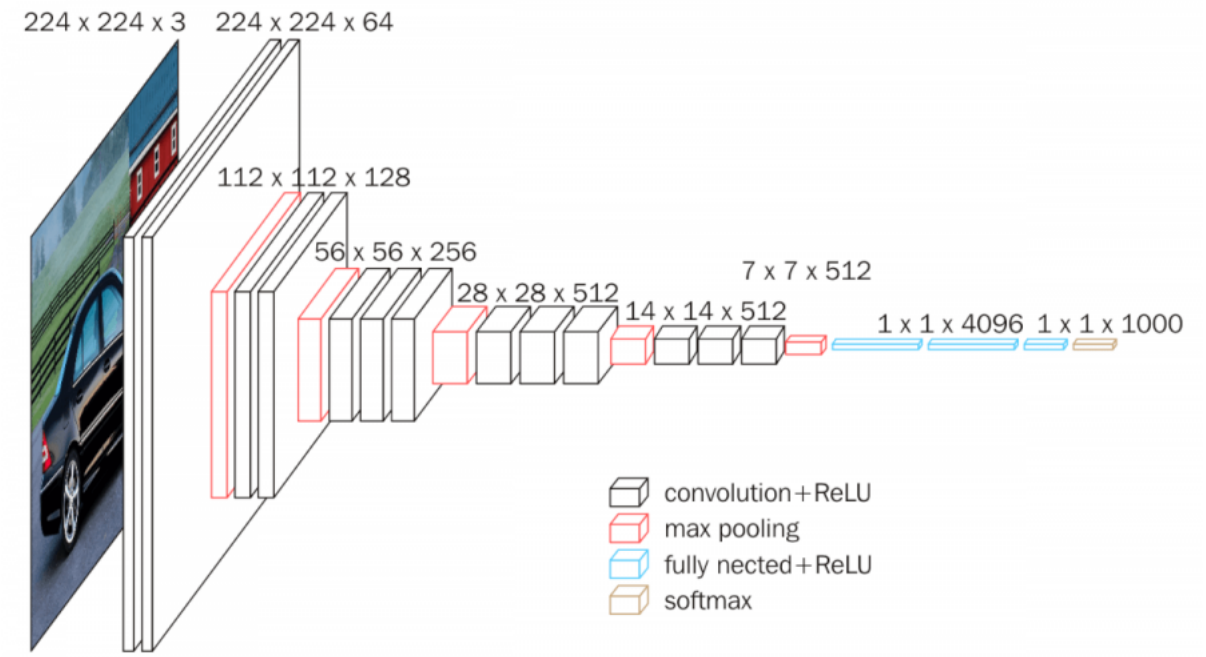


Fig. VII.4.2.: *VGG16 architecture*

It is a certain fact that each models comes with its advantages and disadvantages, these can be seen in the next results section of this technical report.

VII.5. Comparison with other solutions

The results obtained using the models mentioned in the above section are the following:

Results obtained using pre-trained models				
Model Name	**Train loss**	**Train acc.**	**Test loss**	**Test acc.**
VGG16	0.0034	0,8140	0,0083	0,5731
VGG19	0.0032	0,8222	0,0096	0,5287
ResNet50	0.0017	0,9181	0,0090	0,5545
ResNet50V2	0.0028	0,8578	0,0101	0,4770
InceptionV3	0.0050	0,7188	0,0114	0,4230
InceptionResNetV2	0.0034	0,8159	0,0106	0,4833
MobileNet	0.0032	0,8341	0,0087	0,5396
MobileNetV2	0,0035	0,8068	0,0106	0,4677

VII.6. Future work

Regarding the future work, the following ideas could be implemented:

- trying further models to improve the accuracy
- improving the interface to be more complex, along with the login on the back-end side
- implementing the usecase in which a user uploads a video file
- extending the current augio engine to have usecases in security (e.g. detecting breaking in sounds, hearth attack/ people suffering, etc.)
- using the application for analyzing audience behavior(excitement, boredom, etc.)

VII.7. Conclusions

Clink application solved the problem of audio file classification by using deep learning as a consistent base from which the Machine Learning component was built. By dividing the audio and transforming it into spectrograms we used some of the success of image classification and brought it to audio classification. The competition was thought-provoking and it was a great opportunity for us to invest time into this part of audio processing, audio normalisation and image classification. Even though the aplication is nou in a Beta version, we hope we are going to come with further improvements.

VII.8. Links

[1]The Github repository can be accessed at:

[1]Permission required

https://github.com/gabrielxzc/moc1-aset-project

[2]**The Trello Board can be accessed at:**

https://trello.com/b/phl7r8hL/moc1-aset-project

The Kaggle Competition can be accessed at:

https://www.kaggle.com/c/freesound-audio-tagging-2019

[2]Permission required

Bibliography

[1] **Competition's overview from Kaggle platform**: *https://www.kaggle.com/c/freesound-audio-tagging-2019/overview*

[2] **Google's AudioSet Ontology**: *https://research.google.com/audioset/ontology/index.html*

[3] Eduardo Fonseca, Jordi Pons, Xavier Favory, Frederic Font, Dmitry Bogdanov, Andres Ferraro, Sergio Oramas, Alastair Porter, and Xavier Serra. **"Freesound Datasets: A Platform for the Creation of Open Audio Datasets."** In Proceedings of the International Conference on Music Information Retrieval, 2017. [PDF]

[4] Eduardo Fonseca, Manoj Plakal, Daniel P. W. Ellis, Frederic Font, Xavier Favory, and Xavier Serra. **"Learning Sound Event Classifiers from Web Audio with Noisy Labels."** In Proceedings of the International Conference on Acoustics, Speech and Signal Processing, 2019. [PDF]

[5] Frederic Font, Gerard Roma, and Xavier Serra. **"Freesound technical demo."** Proceedings of the 21st ACM international conference on Multimedia, 2013. https://freesound.org

[6] Bart Thomee, David A. Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li, YFCC100M: **The New Data in Multimedia Research**, Commun. ACM, 59(2):64–73, January 2016

[7] **LRAP Python**: *https://scikit-learn.org/stable/modules/model_evaluation.html#label-ranking-average-precision*

[8] **Bidirectional LSTM for audio labeling with Keras**: *https://www.kaggle.com/carlolepelaars/bidirectional-lstm-for-audio-labeling-with-keras#Birectional-LSTM-model-for-audio-labeling-with-Keras*

[9] **About Kaggle**: *https://en.wikipedia.org/wiki/Kaggle*

[10] **Bird sound classification using a bidirectional LSTM**, Lukas Muller and Mario Marti: *http://ceur-ws.org/Vol-2125/paper_134.pdf*

[11] **Convolutional RNN: an Enhanced Model for Extracting Features from Sequential Data**: *https://arxiv.org/pdf/1602.05875v3.pdf*

[12] **"Classifying Urban sounds using Deep Learning"**, Mike Smales: *https://github.com/mikesmales/Udacity-ML-Capstone/blob/master/Report/Report.pdf*

[13] **"Data Science for Good: Stopping Illegal Deforestat"**, Sara Hooker, Sean McPherson: *https://mlconf.com/sessions/data-science-for-good-stopping-illegal-deforestat-2/*

[14] Eduardo Fonseca, Manoj Plakal, Frederic Font, Daniel P. W. Ellis, Xavier Serra. **"AUDIO TAGGING WITH NOISY LABELS AND MINIMAL SUPERVISION"** in Detection and Classification of Acoustic Scenes and Events 2019 [PDF]