Tampere University

Daniel Pato de la Torre

DETECTION OF MOBILE PHONE INTERFERENCE IN ENVIRONMENTAL AUDIO RECORDINGS

Tampere University of Technology
Bachelor's Degree Programme in Signal Processing
May 2020

ABSTRACT

Daniel Pato de la Torre: Detection of Mobile Interference in Environmental Audio Recordings
Tampere University of Technology
Bachelor's Degree Programme in Signal Processing
Examiner: Annamaria Mesaros
May 2020

The era of machine learning has been beginning to be an engine for the development and creation of applications for a few years and the public is not aware that machine learning is on most of the technological devices on the market. Nowadays, this technology is attracting a lot of attention to the researches and it is giving results that were not possible before: the development of vehicles without drivers to recognize a traffic signal, text detection for translation or the recognition of voice and sounds etc. This kind of techniques are made possible by machine learning. Machine learning consists in teaching computers to do what is natural for people: learn through examples. Therefore, it is necessary to have available a large amount of data to provide these examples.

The purpose of this thesis is to develop an application that detects the interference produced by mobile phones in audio recordings through a deep learning architecture, known as feedforward neural networks (FNN), which is used in many machine learning methods. These neural networks will carry out the necessary learning to analyze an acoustic signal and differentiate whether a test audio example contains interference.

To perform this learning, first the sound file is represented in the frequency domain through the mel spectrogram. Deep neural networks (DNN), use a layered structure of units to extract characteristics of the given sound representation input with an increased abstraction in each layer. This increases the ability of the network to efficiently learn the highly complex relationship between sound representation and target sounds.

In this thesis, we will classify interference and no interference categories by constructing a simple model of these samples as inputs and the binary classification at the output.

Keywords: machine learning, neural networks, interference, frequency domain, deep learning, mel spectogram, model, network, binary classification.

PREFACE

This thesis was written at the University of Technology in Tampere, Finland, where the teachers, especially my tutor Annamaria Mesaros and Toni Heittola, who have taught and guided with great care. It has really been a personal challenge for me, since this thesis has been written in another country and in another language than mine. I appreciate my friends from Erasmus, with whom I have always had great moments and we have supported each other.

I especially want to thank my friends and family, who have helped me even though I was many kilometers away.

In Tampere, Finland, on 2 May 2020.

Daniel Pato de la Torre

CONTENTS

LIST OF FIGURES IV

LIST OF SYMBOLS AND ABBREVIATIONS V

1. INTRODUCTION. 6
   1.1 Objectives 7
   1.2 Structure of the Thesis 7
2. BACKGROUND 8
   2.1 Audio Classification 8
   2.2 Audio features 8
   2.3 Machine learning 10
   2.4 Feed-Forward Neural networks 12
3. METHOD 15
   3.1 Dataset 15
   3.2 Computational requirements 15
   3.3 Experimental setup 17
4. CONCLUSION 22

APPENDIX 23
REFERENCES 29

LIST OF FIGURES

Figure 1. *Block diagram for mel spectrogram calculation. 9*
Figure 2. *Mel-spectogram from an-audio file which has interference……10*
Figure 3. *Function sigmoid graph. 12*
Figure 4. *Input layers (left), hidden layer (mid) and output layer (right) from a basic neural networks 13*
Figure 5. *Library used in the practical. 15*
Figure 6. *Version of the necessary applications used in the code.....…….……....15*
Figure 7. *Confusion matrix with n_mels=40, epochs=200 and batch_size=32 ..... 18*
Figure 8. *Confusion matrix with n_mels=40, epochs=100 and batch_size=32. ..... 18*
Figure 9. *Confusion matrix with n_mels=40, epochs=100 and batch_size=64……...18*
Figure 10. *Confusion matrix with n_mels=40, epochs=100 and 200 and batch_size=32 and 64…….....18*
Figure 11. *Confusion matrix with n_mels=128, epochs=100 and 200 and batch_size=32 and 64…….....19*
Figure 12. *Accuracy of confusion matrices in percentage. 19*
Figure 13. *Accuracy of models using different structures and their confusion matrices..... 20*

LIST OF SYMBOLS AND ABBREVIATIONS

AI	Artificial Intelligence
CNN	Convolutional Neural Network
CSV	Comma-Separated Values
DCASE	Detection and Classification of Acoustic Scenes and Events
DCT	Discrete Cosine Transform
DL	Deep Learning
DNN	Deep Neural Network
FFT	Fast Fourier Transform
FNN	Feedforward Neural Network
IEEE	Institute of Electrical and Electronics Engineers
MFCC	Mel Frequency Cepstral Coefficients
ML	Machine Learning
STFT	Short-time Fourier Transform
TAU	Tampere University
TUT	Tampere University of Technology
URL	Uniform Resource Locator

1. INTRODUCTION

In a world where technology advances to the rhythm of a blink and the devices are increasingly smaller, powerful and accessible to all, a better integration of these devices into the daily life of people is sought. One of these technologies is that allows understanding and recognizing events or situations where there are no input parameters, but databases. This technology is known as machine learning.

In order to understand and get to know this current technology better, there are several practical examples that explain what it involves. For example, the prediction of traffic in cities, in which it is observed when and where traffic jams would happen, or the facial recognition used by many of the applications that we use today and that we publish on our social networks. In addition, in modern cars, automatic braking is used when it is detected that a person or object is going to collide against the vehicle.

In the case of the audio branch and engineering, there are various projects, research and applications that also use machine learning. The basic concept is the analysis of the audio file and the classification of sounds. For example, in the given audio fragment, the sounds from the audio file can be separated, detected and classified.

Nowadays we use mobile phones for everything: call, send a message, take a photo or video, record an audio etc. But, when we use the mobile phone, other mobile phones can cause interference. It usually happens when we record in public places such as public transport, where reception conditions increase the transmission power making these interferences inevitable. The most common for environmental audio databases is that these interferences are annotated and discarded manually, thus making it a slow and tedious process.

For that reason, the machine learning techniques can be used to resolve this problem automatically by doing the detection and the classification of interferences in audio recordings.

1.1 Objectives

The main objective of this thesis is the detection of interferences in environmental audio recordings to make a classification of all audible mobile phone interference (if it has an interference or not) and automate the process by using machine learning techniques.

1.2 Structure of the Thesis

The structure of this thesis is as follows:

- Introduction: The current chapter. It deals with the content of the document in an introductory way to put the reader in context. Also, it contains the objectives and purposes of the thesis.
- Background: How to calculate mel spectrogram from an audio file and why mel energies are suitable for the problem. Next the differences between machine learning and deep learning techniques, an introduction on neural networks and the use of feed-forward neural networks for classification.
- Method: Experimental setup, results and analysis of results.
- Conclusions: Development of ideas and solutions after carrying out the experiment.

2. BACKGROUND

2.1 Audio classification

The supervised classification task is the key concept for sound identification. A model is created for each class based on many different examples. In our case, there will be two classes, *No Interference and Interference*. When the model receives a test example, the system we use for classification will analyze the signal and assign it to the most appropriate class of the two. In the training phase, the model that is created is based on the characteristics extracted from the training data [2].

There are many applications for classification of environmental audio, as well as applications focused on people such as speaker identification, speaker gender classification or speaker classification have been developed. Also, in the musical field there are applications such as music artist and genre classification [8]. The DCASE challenge (Detection and Classification of Acoustic Scenes and Events) consists of different tasks that approach a challenge on the recognition of the environmental sounds and classification of the events that happen in the scene. [5]

2.2 Audio features

When it comes to sound representation, there is a time domain where one can display the waveform of the audio signal, the amplitude and period, among other aspects. However, frequency domain features can provide a better representation based on the frequencies involved in the audio signal [1]. Therefore, it is necessary to use frequency domain because the frequency components of a sound are differentiated, and it is possible to analyze and discriminate between different sounds in frequency.

The most used spectral domain feature representation is the mel spectrogram which we use as spectral representation. The block diagram for calculating the mel spectrum is presented in Figure 1 [9].

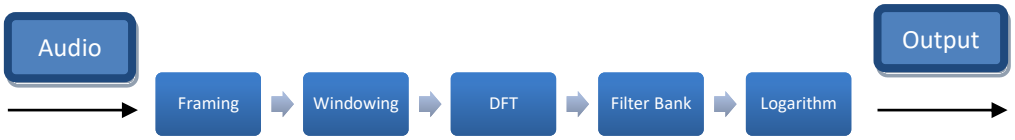

Figure 1 Block diagram for mel spectrogram calculation

Feature extraction consists of the following steps: frame blocking, windowing and frequency domain representation.

The process starts with an audio fragment in the time domain. The next step is to split the signal into short time segments (frames). The size of the analysis frame is chosen based on the application and it usually is between 20 and 100 ms depending on the analysed signal. After this step, each time frame signal is multiplied by a window function. This process is called windowing. There are several types of windows, such as the Hamming, Hanning or Blackman window. In this process, windows are useful because the process eliminates the sharp edges that cause broadband noise. Overlapping the analysis segments is a suitable procedure to make up for the attenuated parts of the input. Next, we proceed to apply the Fourier transform to each frame. The Fourier transform is employed because it shows the signal as a weighted sum of sinusoidal components and short windows are used because signal is considered stationary in such short time [12]. In practice the STFT is a succession of FFTs of windowed data frames. The final step is applying a mel filterbank to transform the spectrum to mel scale.

This scale was created to interpret the pitch in the auditory system in similar way with human perception [11]. The experiment concluded that pitch perception is linear up to 1000 Hz, from there, the perceptual scale becomes logarithmic.

The bank of filters applied to the transform response is typically 40 triangular filters, but more can also be used [4]. Each filter has response 1 at its center frequency, and response 0 at center frequency of adjacent filters. These filters on the mel scale are smaller in bandwidth at low frequencies than at high frequencies, where their bandwidth increases.

The spectrogram is the spectral representation of a signal as time progresses. The spectrogram on linear frequency scale is transformed to spectrogram on mel scale. Logarithm of the energies on mel scale is used in order to mimic perception of loudness by the human auditory system.

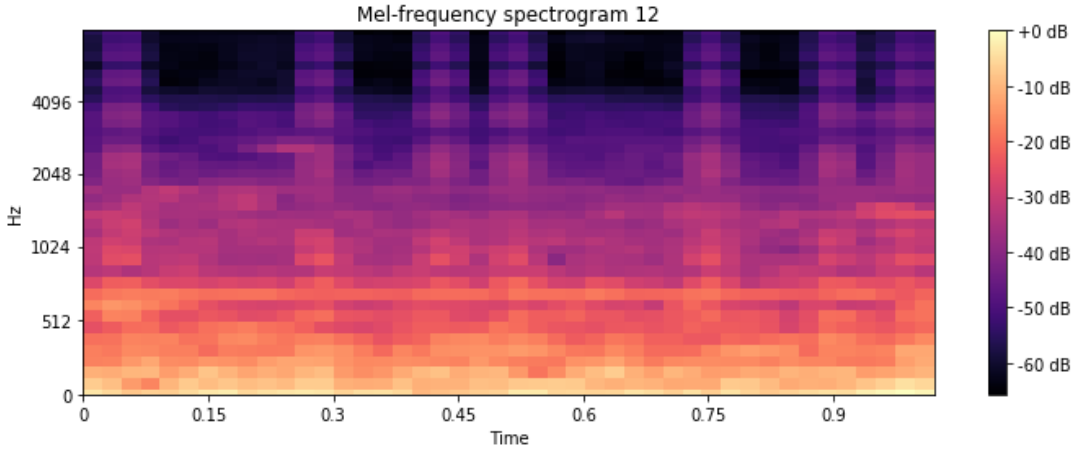

Figure 2 Mel-spectogram from an audio file which has INTERFERENCE

2.3 Machine learning

Artificial intelligence emerged in the 1950s and became a very popular phenomenon due to the great evolution of the computational capacity it had at that time. Alan Turing, considered the father of computing, in his article [6] argued the intelligence of a machine if it acted like a human being. Since this event, the evolution of artificial intelligence has been remarkable and has grown exponentially. Thanks to AI, more advanced algorithms and AI sub-genres such as machine learning and deep learning have emerged and are now used with massive amounts of data and for countless applications.

In [10], there is a formal definition for machine learning (ML) explaining that it is the applied form of statistics that discerns complex mathematical functions where the confidence intervals of these functions do not matter so much. To explain better what ML is, there are three things that it needs:

- Inputs data points: For example, the audio files that we are going to use for this project.
- Expected output: In our case, determine if the files that we introduce as input have interferences or not.
- Algorithm measure of learning: How to evaluate the model with the expected output and the inputs. The most common measure for classification is the confusion matrix or accuracy.

This technology involves the creation of a model that is trained using an algorithm with a database. By giving this model a different sample than what it has been trained with, it is possible to predict a result. There are several categories of ML, but in this thesis, we focus on supervised learning for classification. In supervised learning the system is given training data along with the desired outputs. According to [10] a supervised algorithm associates the input with the output, using a training set of inputs **x** and outputs **y**.

Deep learning (DL) is a specific subfield of ML and brings many possibilities for supervised learning. The advantage of these methods is that they can estimate very complex functions through neural networks by adding more layers and units within a layer. The name of ‘deep’ comes from the large consecutive layers of representations (hidden layers) that one can use.

This technology is primarily used in automatic learning methods, which are so-called neural network architectures, also known as deep neural networks. The term neural network comes from the field of neurology, precisely from the neural networks that form the brain. But it should not be confused and directly associated with the idea that deep learning models work in the same way as the brain, since they are only influenced by features similar to the behavior of neurons and therefore they are mathematical models that learn through data [13].

The main difference between machine learning and deep learning is in how the expected characteristics are extracted to make a classification and determine a result.

2.4 Feed-Forward Neural networks

The basic processing element of a neural network is a neuron (a mathematical function F_x that is intended to resemble a biological neuron) which has input values (X_1, X_2, X_3...) and will generate an output value (Y_1) through the neuron. The neuron performs a weighted summation of the input values according to the weight assigned to each input and works similar to a linear regression line. The result of the linear regression line (the output) is usually evaluated above and below the threshold. This process is aimed at binary classification [14].

In order to solve more complex problems, it is necessary to create a neural network composed of at least two neurons. Two or more neurons placed in a row are in one layer and have the same information as the input (input layer) and have one output layer. Layers between the input and output layers are called hidden layers. The set of all the neurons and their different layers add up to the linear combination of all of them and generate an equivalent linear regression that must be activated by the activation functions. Most neural network models use sigmoid function.

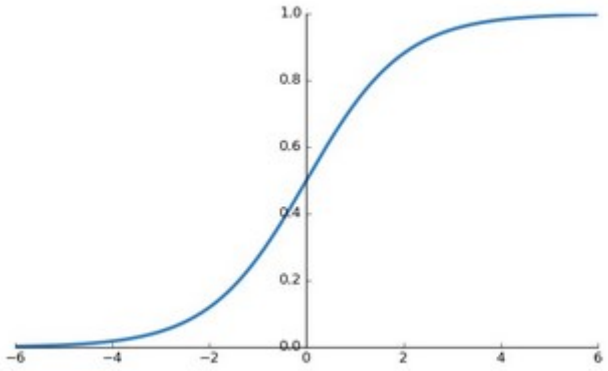

Figure 3 Function sigmoid graph

This activation function is used in most binary classification cases as this is a particular case of this situation. This function is used because it has a good behavior at the output as it is an exponential transition that separates the samples very well between 0 and 1.

$$f(x) = \frac{1}{1 + e^{-x}}$$

Non-linear functions must be added to distort the concatenation of linear regressions in order to obtain as output a linear regression of the whole set of neurons.

Feed-forward neural networks are the preferential model of deep learning and the model used in this project. The main distinction between these networks and other models is the absence of feedback, i.e. the data obtained at the output of the neural network model do not return to the input of the model as input data [7]. What this implies is that the information is only transmitted in one direction, passing since input layer through hidden layers towards output layer of the neural network.

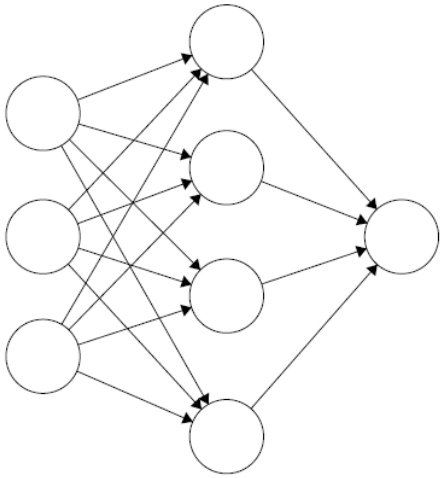

Figure 4 Input layers (left), hidden layer (mid) and output layer (right) from a basic neural network

3. METHOD

In this section, a report on the development of the application will be presented, explaining everything that has been used in the process, how it has been developed and its evaluation.

3.1 Dataset

The dataset which it used for this project has 1-second audio segments from 10 acoustic scenes (some of them are used) having mobile phone interference, in total 2510 sound files. The equipment used for recording are a binaural Soundman OKM II Classic/studio A3 electret in-ear microphone and a Zoom F8 audio recorder (48 kHz sampling and 24-bit resolution) [3]. Dataset is split into training and test sets with 70% data in the training

3.2 Computational requirements

In the previous sections we have introduced which resources are going to be used in this project. Here, we are going to deal with each of these resources in detail.

The application has been developed with the Python programming language, which provides various specific audio libraries. In the code developed, 4 essential libraries have been used, including libraries for the extraction of audio features and for creating models of neural networks:

- **Numpy**: It is a scientific numerical calculus library for Python. It is used in this project to create matrices and operations with them.
- **Pandas**: It is a library of science and data analysis software based on Numpy for Python. It is used to read the database of the audio segments that contain interference by transforming the data in CSV format into arrays to operate with them.
- **Librosa**: This is an audio and music analysis library for Python. The main purpose of using this library is the melspectrogram function that will be used to extract the characteristics of the audio segment being analyzed. It is also used to load the data from the audio segments to process the values and enter them into the mel spectrogram function.
- **Keras**: A deep learning library used for neural networks in Python. There are two important functions necessary for the creation of the neural network.

Sequential from Keras: It is a function that creates a neural model of a stack of layers that can be modified with parameters along with the Dense function.

Dense from Keras: It is a function that comes along with the previous one and modifies the parameters of each layer such as the number of nodes that will be used or the activation function that will be applied in each case.

```
 9 import numpy as np #number processing
10 import pandas as pd #load .csv files
11 import librosa as lr #mel spectogram
12 from keras.models import Sequential
13 from keras.layers import Dense
```

Figure 5 Libraries used in the practical implementation

Besides Keras (since the library's own functions are used), TensorFlow and Anaconda are used. While the first is an extensive open-source library for machine learning and especially deep neural networks (DL), the other is a well-known data science research platform in machine learning.

The corresponding versions of these applications that were used for this project are as follows:

Application	Version
Python	3.7.3
Anaconda Navigator	1.9.7
Keras	2.2.4
Tensorflow	1.13.1
Numpy	1.16.4
Pandas	0.24.2
Librosa	0.7.1

Figure 6 Versions of the neccesary applications used in the code

3.3 Experimental setup

In summary, we have a database containing one-second-long ambient audio segments where several audios contain mobile phone interference and other ones do not contain interference. The goal is to create a neural model that automates the process of classifying audio segments between interference and non-interference. For this purpose, a binary classification will be performed, being a value of 1 for those cases where there are interferences and a value of 0 for those cases without interferences. The dataset is divided in three subsets: **Train, Test and Evaluate.**

- Train: it corresponds to the training data for the neural model which is trained with the audio files identified as INTERFERENCE or NORMAL so that this model learns to differentiate them.
- Test: It contains different audios from those of the training set.
- Evaluate: Contains the ground truth for the test set.

Each audio file is represented in feature domain as a matrix with dimensions 40x44, where 40 is the number of the mel energies and 44 is the number of frames. The number of frames depends on frame length and file length, so it takes 44 frames for one audio file in our dataset.

First, the next step is obtaining mel energies from spectrogram. It is necessary loading all audio from TRAIN data to obtain mel energies values. Spectrogram function needs parameters as the window type, window length, mel filters numbers and the mel numbers. The window type used is Hanning window because it is used for better resolution in the frequency domain and for noise measurements, window length is 40ms because using shorter time it obtains more spectral information. As for the number of mel filters and number of mel energies we use 40 to obtain the mel spectrogram.

The previously initialized arrays are simply concatenated and saved in a new array whether the file contains interference or not.

The neural network includes 2 hidden layers of 50 and 20 neurons respectively and an output layer of one neuron with the sigmoid activation function as explained above and that will determine if the result of the model will contain a 1 or a 0.

In the training stage, a high number of epochs have been used for learning (200 epochs) and 32 batch size of data for each iteration. A second experiment was also performed where the number of iterations is 100 and batch size is 64.

The TESTING stage uses the same procedure as for the TRAINING stage only this time the trained model will test with the data from the TEST set to obtain a prediction of each time frame of each audio file and determine if the file will be INTERFERENCE or NORMAL. The criterion for choosing whether an audio is NORMAL or INTERFERENCE is the majority voting method. At this stage, we predict file by file and frame by frame. If each file has 44 frame length, the first frame is evaluated as a sigmoid function. If mel energies value of that frame is greater than or equal to 0.5, that frame will be considered 1, otherwise it will be considered 0. This process is performed until 44 frame length is reached. With the output matrix full of zeros and ones, at the end a count is made of the number of ones and zeros that the audio file has. If there are more zeros than ones, the label that the file will have will be 0 or NORMAL, on the contrary, it will be 1 or INTERFERENCE. This method is replicated for all audio files in the TEST set.

In the EVALUATION part the results obtained from the TESTING stage of each audio file will be compared with the ground truth in the EVALUATE set. This process will generate a result determining the accuracy of the model and the confusion matrix. As for the confusion matrix, this is a tool used in supervised learning, where the elements (in our case they are audio files) are divided into predictions and actual values of each class. The classes, as we know, are NORMAL and INTERFERENCE.

In order to calculate the percentage of the model's accuracy, the sum of the NORMAL / NORMAL and INTERFERENCE / INTERFERENCE values of the predicted and real values is done. The sum of this set is divided by the total number of audio files and multiplied by 100 to obtain the percentage of accuracy of the model.

For this case where **number of epochs are 200, batch size is 32 and number of mel energies are 40** (Figure 7), the model has **58.58 % accuracy**.

	ACTUAL		
PREDICTED		NORMAL	INTERFERENCE
	NORMAL	222	158
	INTERFERENCE	187	266

Figure 7 Confusion matrix with n_mels=40, epochs = 200 and batch_size = 32

To improve this result, the epochs and batch size variables have been changed. For the second case, **number of epochs are 100, batch size equal to 32** (Figure 8), where the model has **60.02 %.** Now it has been demonstrated that by reducing the number of iterations the percentage of accuracy increases due to an overfitting of the model.

	ACTUAL		
PREDICTED		NORMAL	INTERFERENCE
	NORMAL	149	231
	INTERFERENCE	102	351

Figure 8 Confusion matrix with n_mels=40, epochs = 100 and batch size = 32

For a third case, only the **batch size value was changed to 64** (Figure 9), and the model has **61.11 % accuracy.**

	ACTUAL		
PREDICTED		NORMAL	INTERFERENCE
	NORMAL	231	149
	INTERFERENCE	175	278

Figure 9 Confusion matrix with n_mels=40, epochs = 100 and batch size = 64

For the next experiment, **mel energies number was changed to 128** instead of 40 and the previous experiments were repeated (Figure 10 and 11).

Actual / Predicted	n_mels = 40		
Epochs=200 Batch=32		NORMAL	INTERFERENCE
	NORMAL	222	158
	INTERFERENCE	187	266
Epochs=100 Batch=32		NORMAL	INTERFERENCE
	NORMAL	149	231
	INTERFERENCE	102	351
Epochs=100 Batch=64		NORMAL	INTERFERENCE
	NORMAL	231	149
	INTERFERENCE	175	278

Figure 10 Confusion matrices using differents variables. n_mels:40, epochs:100 and 200, batch size:32 and 64

Actual / Predicted	n_mels = 128		
Epochs=200 Batch=32		NORMAL	INTERFERENCE
	NORMAL	71	309
	INTERFERENCE	28	425
Epochs=100 Batch=32		NORMAL	INTERFERENCE
	NORMAL	279	101
	INTERFERENCE	310	143
Epochs=100 Batch=64		NORMAL	INTERFERENCE
	NORMAL	148	232
	INTERFERENCE	72	381

Figure 11 Confusion matrices using differents variables. n_mels:128, epochs:100 and 200, batch size:32 and 64

Actual / Predicted	n_mels = 40	n_mels = 128
Epochs=200 Batch=32	58.58%	59.54 %
Epochs=100 Batch=32	60.02%	50.66%
Epochs=100 Batch=64	61.11%	63.51%

Figure 12 Accuracy of different experiments in percentage

If we increase the number of mel energies for the case where **epochs = 100** and **batch = 32**, the model suffers a drop in classification accuracy.

In addition, a few more network architectures were tested to try how the model works with other variables, so the structure of the neural network was changed. The original architecture is **Network 1**, the structure with more layers **Network 2** and the network with less layers **Network 3**. (Figure 12)
It was used for the different structures the values **n_mels=128, epochs = 200** and **batch = 64**.

Initially in **Network 1**:

- 4 layers.
- In the entry layer, 100 neurons were used.
- 2 hidden layers with 50 and 20 neurons.

Network 2:

- 8 layers.
- In the input layer, 500 neurons were used.
- In the 6 hidden layers, from 250 neurons to 10 neurons decreasing twice the number of neurons per layer.

Network 3:

- 3 layers.
- In the entry layer, 100 neurons were used.
- In the hidden layer, 50 neurons.

	NETWORK 1		NETWORK 2		NETWORK 3	
Actual / Predicted	**NORMAL**	**INTERF.**	**NORMAL**	**INTERF.**	**NORMAL**	**INTERF.**
NORMAL	71	309	130	250	284	96
INTERF.	28	425	84	369	304	149
Accuracy %	**59.54 %**		**59.90 %**		**51.98 %**	

Figure 13 Accuracy of models using different structures and their confusion matrices

4. CONCLUSION

The conclusions to be drawn from this experiment are diverse and varied.

Reviewing all the process that has been done in the experimental setup, the purpose was to build a neural network that analyzes, detects and classifies audio files in two output tags: NORMAL or INTERFERENCE.

A 40x44 dimension feature representation was used for each file, being 40 mel energies and 44 frames, extending the experiment to 128 mel energies to observe a difference of the model. In addition, for the calculation of the spectrogram, typical parameters such as the Hanning window and a small window length were used.

The neural network model should be reviewed and improved using other types of neural networks such as CNN and using another methodology to classify as the method used majority vote.

Firstly, the results at first sight determine that the model does not learn the data well enough (Underfitting). In particular **Network 2** (case where there are more hidden layers), while training the model in the Keras fit function, I have observed a drop from the parameter accuracy=0.8332 in epoch=175 to a value accuracy=0.6720 for epoch=176.

This probably means that learning should be stopped at epoch=175, however, the accuracy on the test data was not improved with this experiment as expected in theory.

In summary, it has been shown that by varying most of the elements involved in the neural network such as the number of mel energies and iterations, the accuracy of this deep learning model varies slightly without obtaining a reliable result. The results vary between 58% and 64% accuracy.

APPENDIX

This appendix includes the code to create a model to detect interferences and make a classification. The variables used in this code are: epochs=200, batch_size=32 and n_mels=40.

```
import numpy as np #number processing
import pandas as pd #load .csv files
import librosa as lr #mel spectogram
from keras.models import Sequential
from keras.layers import Dense

#Reading our dataset
data = pd.read_csv(r"**PATH**WHERE**IS**THE**FILE**\fold1_train.csv", sep = ';')
test = pd.read_csv(r"**PATH**WHERE**IS**THE**FILE**\fold1_test.csv")
evl = pd.read_csv(r"**PATH**WHERE**IS**THE**FILE**\fold1_evaluate.csv", sep = ';')

#Variables
t_files = data.values
event_label = data['event_label']
labels = event_label.values
data_dir = './TAU-urban-acoustic-errors-2019_BASEDATOS/audio/'
M = np.empty((1,44))
A = np.empty([40,44])
count = 0
```

```
#MEL SPECTOGRAM
for file in range (0, len(t_files), 1):
    audio, sfreq = lr.load(data_dir + t_files[file][0])
    D = np.abs(lr.stft(audio))**2
    mels = lr.feature.melspectrogram(S=D, sr=sfreq,
                                     win_length=0.04,
                                     power=1,
                                     window='Hanning',
                                     n_mels=40)


  #Change Event_label by number (NORMAL = 0 ; INTERFERENCE = 1)
    if event_label[file] == 'NORMAL':
        event_label[file] = 0
    else:
        event_label[file] = 1
    #Create array of zeros or ones (If it is NORMAL = Zeros // INTERFERENCE
= Ones)
    if labels[file] == 0:
        N = np.zeros((1,44))
    else:
        N = np.ones((1,44))
    #Concatenate outputs arrays(ZEROS or ONES)
    if count == 0:
        M = N
    else:
        M = np.concatenate((M,N),1)
```

```
    #Concatenate inputs arrays (mel energies)
    if count >= 1:
        A = np.concatenate((A,mels), 1)
    else:
        count = count + 1
        A = mels

    print ('\nTRAINING FILE NUMBER: ', file)

############# MODEL & TRAINING ###############

#Define Keras Model
model = Sequential()
model.add(Dense(100, input_dim=40, activation='relu')) #nodes, inputs, activation function
model.add(Dense(50, activation='relu'))
model.add(Dense(20, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
print (model.summary())

#Compile neural network
model.compile(loss='binary_crossentropy', # Cross-entropy
              optimizer='rmsprop', # Root Mean Square Propagation
              metrics=['accuracy']) # Accuracy performance metric

#Transpose matrixes
M=np.transpose(M)
A=np.transpose(A)
```

```
#Inputs and outputs
x_train = A #mel energies
y_train = M #labels

#Training neural network
model.fit(x_train, y_train,
        epochs=200, #epochs are iterations
        batch_size=32, #batch are data amount by each iteration
        validation_split=0.3)

############ TESTING ##################

#Variables
test_files = test.values
A1=np.empty([44,1])
pred = np.empty([877,44])
B1 = np.empty([833,1])
count_ones= 0
count_zeros= 0
cont = 0

#MEL SPECTROGRAM
for f in range (0, len(test_files), 1):
   #Load audio files
   audio, sfreq = lr.load(data_dir + test_files[f][0])
   D = np.abs(lr.stft(audio))**2
   S1 = lr.feature.melspectrogram(S=D, sr=sfreq,
                                  win_length=0.04,
                                  power=1,
                                  window='Hanning',
                                  n_mels=40)
```

```
    S1=np.transpose(S1)
    pred = model.predict(S1) #predict model by each file
    if cont >= 1:
        A1 = np.concatenate((A1,pred), 1)
    else:
        cont = cont + 1
        A1 = pred

    print ('\nTESTING FILE NUMBER: ', f)

A1 = np.transpose(A1)
for f_ile in range (0,len (test_files), 1):
    count_ones = 0
    count_zeros = 0
    for frame in range (0,len(S1),1):
        if A1[f_ile,frame]>=0.5:
            A1[f_ile,frame]=1
            count_ones = count_ones + 1
        else:
            A1[f_ile,frame]=0
            count_zeros = count_zeros + 1
    if count_ones >= count_zeros:
        label = 1
    else:
        label = 0
    B1[f_ile,0] = label
```

```
############ EVALUATION ##################

#Variables
evaluation_files = evl.values
labels = evl['event_label']
true_labels=labels.values
correct = 0

#Count numbers of true zeros and ones in evaluation file
for r in range (0, len(true_labels), 1):
    if true_labels[r] == 'NORMAL':
        true_labels[r] = 0
    else:
        true_labels[r] = 1
    if true_labels[r] == B1[r,0]:
        correct = correct + 1

print ('\nTRUE ZEROS AND ONES: ', correct)

accuracy = correct / len(true_labels) * 100

print ('\nACCURACY = ', accuracy)
B1 = np.transpose(B1)
trues = np.reshape(true_labels,833)
predictions = np.reshape(B1,833)
df_confusion = pd.crosstab(trues, predictions)
print(df_confusion)
```

REFERENCES

[1] Cakir, E. (2019). Deep Neural Networks for Sound Event Detection. (Tampere University Dissertations; Vol. 12) Tampere University. Date: 4/10/2019 [Online]. URL: https://tutcris.tut.fi/portal/files/17626487/cakir_12.pdf

[2] Annamaria Mesaros (2012). Singing Voice Recognition for Music Information Retrieval. Tampere University. Date: 8/10/2019 [Online]. URL: https://tutcris.tut.fi/portal/files/5346343/mesaros.pdf

[3] Mesaros, A., Heittola, T., Virtanen, T., “TAU Urban Acoustic Errors 2019”, Tampere University, 2019.

[4] Maka, Tomasz. (2015). Change Point Determination in Audio Data Using Auditory Features. International Journal of Electronics and Telecommunications. 61. 10.1515/eletel-2015-0024.

[5] Vázquez Baldovino, Miguel (2018). *Creación GUI para detección de eventos sonoros*. Proyecto Fin de Carrera / Trabajo Fin de Grado, E.T.S.I. y Sistemas de Telecomunicación (UPM), Madrid.

[6] A. M. Turing (1950) Computing Machinery and Intelligence. Mind 49: 433-460.

[7] Charte, David. (2017). Reducción de la dimensionalidad en problemas de clasificación con Deep Learning: Análisis y propuesta de herramienta en R. 10.13140/RG.2.2.16155.57123/1.

[8] Lee, H., Pham, P., Largman, Y., & Ng, A. Y. (2009). Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in neural information processing systems* (pp. 1096-1104).

[9] Lee, R. (2019). *Software Engineering Research, Management and Applications*. New York, Estados Unidos: Springer Publishing. (pp. 175-180).

[10] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. Amsterdam, Países Bajos: Amsterdam University Press. (pp. 95 - 182)

[11] Majeed, S. A., Husain, H., & Samad, S. A. (2015). Mel frequency Cepstral coefficients (MFCC) feature extraction enhancement in the application of speech recognition: A comparison study. *Journal of Theoretical & Applied Information Technology, 79(1).*

[12] J. B. Allen and L. R. Rabiner, ``A unified approach to short-time Fourier transform and synthesis,'' *Proc. IEEE*, vol. 65, pp. 1558-1564, Nov. 1977.

[13] Chollet, F. (2017). *Deep Learning with Python.* Manning. ISBN: 9781617294433

[14] Dot CSV.(2018, march 19). *¿Qué es una Red Neuronal? Parte 1 : La Neurona* | *DotCSV* [Video file]. Online: https://www.youtube.com/watch?v=MRIv2IwFTPg