POLITECNICO
MILANO 1863

Dipartimento di Elettronica, Informazione e Bioingegneria

Master Degree in Computer Science and Engineering

Closed and open set classification of real and AI synthesised speech

by:
Michelangelo Medori

matr.:
878025

Supervisor:
Fabio Antonacci

Co-supervisors:
Paolo Bestagini
Clara Borrelli

Academic Year
2019-2020

Abstract

Huge advancements in the development of artificial intelligence techniques have been made in the last decade, which have led to the diffusion and spread of computer generated multimedia content, consisting of images, audio and video, which is so realistic that it makes it difficult to be told apart from original content of the same nature.

While there are interesting applications to artificial intelligence generated content, it can also be used in dangerous and deceiving ways, for example as proof in a court of law. Hence it is more and more urgent to find automatic ways to distinguish artificial intelligence synthesised content from original content.

In this paper we take into account audio content, and in particular speech, which is obviously utterly delicate as it comes to forgery. We are going to deepen the previous research made in the field of bispectral analysis in order to create more general automatic methods to recognise real speakers from artificial intelligence synthesised speech.

The dataset of voices that we have used is very wide and heterogeneous, consisting of both real voices and voices synthesised using various different methods. We extracted the Bicoherence from all the speech recordings and performed some classifications (both multilabel classifications, which consist in distinguishing each class of voices from all the others, and binary classifications between real and fake voices) using various machine learning techniques, such as support vector machine, logistic regression and random forest.

In particular, once the bicoherences have been computed from the audio files, we performed the following tests. First of all we replicated the tests made on previous works extracting from the bicoherences a set of features which consist on mean, variance, skewness and kurtosis of both the modules and the phases of the bicoherences and trying to classify them performing simple multiclass and binary classifications using a support vector machine, a series of logistic regressors and a random forest.

Then we simulated an open set environment using a series of support vector machines, in order to test the model with data not yet seen in the training phase.

Moreover, we used a series of U-Nets to extract a new set of features and tried to classify them performing simple multi-label and binary classifications.

Finally, we concatenated the two set of features above and performed more classifications with them (in this case also with an open set environment) and we are going to show that with this method we obtained the best results.

We hope that these results can clarify better the role of bispectral analysis in distinguishing between real and fake speech recordings, and could lead to more research in the field of multimedia forensics.

Sommario

Nell'ultimo decennio sono stati fatti grandi passi avanti nello sviluppo di tecniche di intelligenza artificiale, che hanno permesso la diffusione in larga scala di contenuti multimediali generati artificialmente, come audio, immagini e video che sono talmente realistici da risultare praticamente impossibili da distinguere da contenuti multimediali originali dello stesso tipo. Nonostante ci siano molte interessanti applicazioni per questo materiale generato algoritmicamente, questo può essere usato anche in modo improprio e pericoloso, per esempio come prova in una corte di giustizia.

Per questo motivo, risulta sempre più urgente trovare modi automatici per distinguere contenuti multimediali generati da intelligenze artificiali da quelli originali.

In questa tesi prendiamo in considerazione contenuti audio, e ci occuperemo in particolare del parlato, che è ovviamente molto delicato nel momento in cui venga "falsificato". Andremo ad ampliare la ricerca fatta in lavori precedenti riguardo l'analisi bispettrale al fine di creare metodi automatici il più possibile generali per distinguere il parlato sintetico da quello registrato da parlatori veri.

Il dataset che abbiamo usato è molto ampio ed eterogeneo, e consiste sia di voci "vere" che di voci generate algoritmicamente con vari metodi di sintetizzazione. Abbiamo estratto le bicoerenze da tutte le registrazioni di parlato, e abbiamo provato fare delle classificazioni (sia classificazioni multi-label, che consistono nel distinguere ogni classe da tutte le altri, sia classificazioni binarie, per distinguere le voci vere da quelle finte) usando vari metodi di machine learning come support vector machine, logistic regression e random forest.

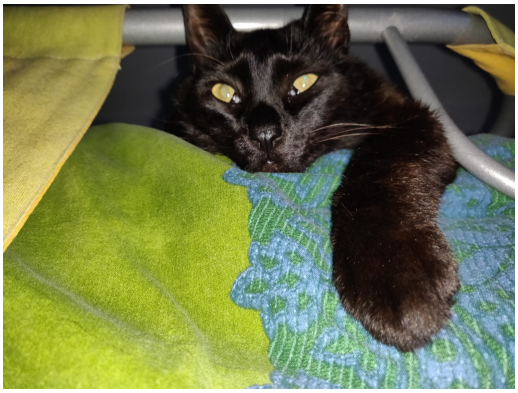
In particolare, una volta calcolate le bicoerenze, abbiamo eseguito i seguenti test: per prima cosa abbiamo replicato gli esperimenti fatti in lavori precedenti estraendo dei descrittori (che consistono nei primi quattro momenti statistici) sia dai moduli che dalle fasi delle bicoerenze e provando a classificarli per mezzo di semplici classificazioni (sia multi-label che binarie) usando una support vector machine, una serie di logistic regression e una Random forest; a seguire abbiamo simulato un ambiente open set usando una serie di support vector machine, al fine di testare il modello con dati mai visti nella fase di training; inoltre abbiamo usato delle reti neurali chiamate U-Net per estrarre un nuovo insieme di descrittori che abbiamo provato a classificare tramite semplici classificazioni multi-label e binarie; infine abbiamo concatenato i due insiemi di

descrittori di cui sopra e abbiamo provato a classificarli, sta volta simulando anche un ambiente open set, e mostreremo che in questo modo si ottengono i risultati migliori.

Ci auguriamo che il lavoro fatto possa chiarire meglio il ruolo che ha l'analisi bispettrale nel distinguere il parlato vero da quello sintetizzato da intelligenza artificiale, e possa aprire la strada ad altra ricerca nell'ambito dell'audio forense.

Ringraziamenti

Ogni cosa ha un inizio e una fine.
In questo caso la fine (ovvero i ringraziamenti) coincide con l'inizio (di questa tesi).
Inizierei subito col ringraziare Clara, Fabio e Paolo, che sono stati fondamentali per il compimento di questo lavoro, e dai quali ho imparato molto nel corso del mio ultimo anno di studi universitari.
Un altro importante ringraziamento va ai miei colleghi (i "Cremonesi"), ovvero Edo, Jackie, Malvo, Marzo, Molgo e Vago: senza di voi non sarei arrivato a questo punto.
Una menzione speciale va a Marta e Robin, che mi hanno sostenuto negli ultimi mesi, aiutandomi a superare delle difficoltà impossibili da trascurare.
Un ulteriore importante ringraziamento va alla mia famiglia, e in particolare a mia madre, che ha vissuto in prima persona il superamento di ogni ostacolo che ho dovuto affrontare lungo questo cammino.
Infine, a tutti coloro che non sono stati menzionati, va un ringraziamento per i bei momenti passati insieme...e anche per quelli brutti: dopotutto l'importante è riuscire ad andare avanti senza farsi abbattere.
Questo lavoro è dedicato a tutti coloro che sanno ascoltare.
Che la fine di questo capitolo sia l'inizio di uno migliore.
Nella foto sotto, il co-autore di questa tesi.



M.M.

In memoria di Alessio Nava
1994-2018

Contents

List of Figures

List of Tables

Glossary

AI artificial intelligence. xiii, xix, 22, 27, 28, 34, 73, 74

ASV automatic speaker verification. xiii, 23, 50

AUC area under curve. xiii, 19

CNN convolutional neural network. xiii, xxi, 1, 14, 31, 39, 45, 76

CSBVAS closed set bonafide versus all spoof. xiii, 45, 52, 53, 57, 58, 70, 76, 83, 88

CSBVSS closed set bonafide versus single spoof. xiii, 45, 52, 53, 76, 83, 88

CSM closed set multiclass. xiii, 45, 52, 55, 58, 60, 70, 76, 83, 88, 90

DFT discrete Fourier transform. xiii, 2, 28, 37

EER equal error rate. xiii, 32

ENF electric network frequency. xiii, 22

FFT fast Fourier transform. xiii, 2, 52

FN false negative. xiii, 17

FP false positive. xiii, 17

FPR false positive rate. xiii, 19, 49

GCD Greatest Common Divisor. xiii

GMM Gaussian mixture model. xiii, 26

GMM-UBM Gaussian mixture model - universal background model. xiii, 32

HMM hidden Markov model. xiii, 26, 51

ICA independent component analysis. xiii, 3, 5, 6, 44, 59

LA logical access. xiii, 50, 51

LPCCs linear prediction cepstrum coefficients. xiii, 2, 27, 32

LR logistic regression. xiii, xx, 1, 8, 11–13, 36, 45, 57, 58

LTS letter-to-sound. xiii, 24

MFCCs Mel-frequency cepstrum coefficients. xiii, 2, 22, 25–28, 32, 51, 52

MK mean-kurtosis. xiii, xx, xxi, 1, 3, 27, 35, 36, 38, 39, 43–45, 53, 54, 59–62, 66, 67, 70, 74, 83

MKU mean-kurtosis-UNET. xiii, xx, xxi, 1, 3, 27, 35, 36, 43–45, 53, 54, 59, 61, 66, 70, 74, 78, 83, 90

MSA morpho-syntactic analyser. xiii, 23, 24

MSE mean square error. xiii, xx, 42, 79, 80, 83

NLP natural language processing. xiii, 23, 24

NN neural network. xiii, 13, 51, 52

OSBT1 open set bonafide training version 1. xiii, 48, 52, 62, 64, 67

OSBT2 open set bonafide training version 2. xiii, 48, 52, 62, 64, 67

OSBT3 open set bonafide training version 3. xiii, 48, 52, 63, 64, 67, 70

OSST1 open set spoof training version 1. xiii, 48, 52, 65, 69

OSST2 open set spoof training version 2. xiii, 48, 52, 65, 69

OSST3 open set spoof training version 3. xiii, 48, 52, 65, 66, 69–71

PA physical access. xiii, 50, 51

PCA principal component analysis. xiii, 3, 4, 44, 59

PCs principal components. xiii, 4, 5

PG prosody generator. xiii, 24

PRNU photo response non uniformity. xiii, 21

PSOLA pitch synchronous overlap and add. xiii, 25

RF random forest. xiii, xx, 1, 8, 12, 13, 36, 45, 57, 58

RL reinforcement learning. xiii, 8

RNN recurrent neural network. xiii, 51

ROC receiver operating characteristic. xiii, 1, 8, 17, 19, 49, 64, 65, 68, 69, 79, 80, 83

SNE stochastic neighbour embedded. xiii, 6, 7

SVM support vector machine. xiii, xx, 1, 8, 10, 11, 13, 36, 45, 55, 58, 66, 69, 83

TN true negative. xiii, 17

TP true positive. xiii, 17

TPR true positive rate. xiii, 19, 49

TSNE t-distributed stochastic neighbour embedded. xiii, 3, 6, 7, 44

TTS text to speech. xiii, 20, 23, 26, 34, 44, 50–52

VC voice conversion. xiii, 20, 23, 26, 34, 44, 50–52

Introduction

Nowadays it is increasingly common to get in contact with artificial intelligence generated multimedia content. On one hand the spread of social media like Twitter, Facebook, Instagram, WhatsApp, Telegram, has made incredibly easy to instantly share both information and multimedia content like images, audio and video with anybody around the world, with no possibility to guarantee the truthfulness or authenticity of the material. This could lead to the diffusion of fake news, which is the most common side effect of the propagation of social media.

On the other hand, artificial intelligence applications are more and more common and easy to use. As concerns audio signals, and in particular synthesised speech, it is easy to find it in everyday life. We can bump into artificial speech on public transportation, in many call centers, in the medical field (speech synthesis is used to support people with phonatory pathologies or with dyslexia / dysgraphia issues or visually impaired people), in audio books, navigation systems, smartphones and tablets, and domestic virtual assistants such as Amazon Alexa and Google Home.

Moreover, some tools that allow to algorithmically modify or generate speech have become more and more popular and straightforward, and while they have interesting applications, in the wrong hands they could become dangerous.

Allowing people to share multimedia content together with the diffusion of artificial intelligence (AI) generated one, leads us to issues when it comes to understand if this material is real or fake, and there are situations in which understanding it becomes absolutely fundamental.

In the case of speech, which is utterly delicate, let us imagine a scenario where someone was able to realistically modify the speech of some wiretapping to be used as proof in a court of law. The modified audio evidence could lead to a wrong sentence [2]. There is also the possibility that the voice of a public figure (maybe a known political figure) is synthesised in order to make him/her say unwanted things. In this case the forged signal could be used to manipulate the results of an election or the relationships between states [3]. The examples above make clear the importance of understanding the nature of speech recordings (and multimedia content in general), which is one of the applications of the field of multimedia forensics (and more specifically audio forensics).

The work we have done aims at developing general automatic procedures that are able to recognise recordings of real speakers (which we

are going to refer to as bonafide or real speech recordings) from speech synthesised using artificial intelligence methods (which we are going to refer to as spoofed or fake speech recordings).

In order to do that, we employed bispectral analysis, which consists in the computation of a complex feature called bicoherence, that can be derived starting from a time-frequency representation of an audio excerpt. The hypothesis we do is that spoofed speech recordings are supposed to show some kind of higher order correlations which are not present in bonafide speech recordings. These correlations are due to nonlinear processing applied to the signals, which alters the spectral content of the signals themselves, and cannot be detected by standard features like the power spectrum. Since the bicoherence is a third order feature (w.r.t to the power spectrum which is a second order feature) there should be a difference between the bicoherences computed from the spoofed and bonafide speech recordings, as pointed out in [1] and [4].

Once computed the bicoherences, we have classified them using different techniques taken from the machine learning field.

In particular, we extracted three different sets of features from the bicoherences, which we are going to denote as mean-kurtosis (MK), UNET and mean-kurtosis-UNET (MKU) features.

MK features are obtained computing mean, variance, skewness and kurtosis of both the modules and the phases of the Bicoherences. This features extraction operation has already been proposed in [1].

For the extraction of UNET features, five different U-Nets have been trained with the modules of the bicoherences extracted from five different classes of spoofed speech. After the training, the modules of all the available classes of bicoherences have been given as input to the five different U-Nets, obtaining, as output for each speech recording, five different images reconstructed by the networks. UNET features then correspond to the mean square errors (MSEs) between the "compressed" versions (obtained through the encoder part of the U-Net) of the original and reconstructed image. The result consists in five different MSEs for each speech recording.

MKU features, finally, are simply obtained concatenating MK and UNET features.

We have performed binary (bonafide versus fake) and multiclass (to distinguish every speech synthesis algorithm from all the others) classifications of MK , UNET and MKU features using a support vector machine (SVM), a logistic regression (LR) and a random forest (RF).

Moreover, we have also implemented an open set environment using a series of support vector machines, where each SVM operates a one-versus-the-rest classification, with and without the use of known-unknown features.

We hope that the results achieved can further clarify the role of bispectral analysis in the distinction between real and fake speech and could lead to more research in the field of audio (and speech) forensics.

This work is organised as follows: in Chapter 1 we present some notions on features extraction and normalisation, dimensionality reduction techniques and classification algorithms that can be useful in order to understand the work we have done. Chapter 2 gives an overview on the state of the art as concerns multimedia forensics and speech synthesis techniques, other than describing few methods in literature that can be used to recognise real speakers from speech generated automatically by computing systems and artificial intelligence techniques. In Chapter 3 we will describe the methodology we have used in detail, while in Chapter 4 we will show the results of the open set and closed set classifications performed using the MK, UNET and MKU features, testing our models with a various and heterogeneous dataset of voices (both bonafide and spoofed speech signals). In Appendix A can be found the results of classifications performed using a convolutional neural network (CNN) applied to the modules of the bicoherences. In Appendix B are shown the results of experiments made using a U-Net trained on bonafide speech. Finally, in Appendix C we will show all the confusion matrices related to the closed set classifications that have been performed.

1 Theoretical Background

In this Chapter we will briefly introduce feature extraction and normalisation and we will describe few techniques of dimensionality reduction.

Then, after an introduction on machine learning, will describe the problem of classification in detail. Moreover, we are going to analyse few different classifications techniques such as SVM, LR and RF, which we are going to use to classify our features, and we will show how to use CNNs and convolutional autoencoders to perform classifications.

Finally, we will explain what it means to perform an open set classification and we and we will introduce some metrics like confusion matrices, accuracy and receiver operating characteristic (ROC) curves, which will be useful in order to evaluate the performance of our classifiers.

1.1 Features extraction, features normalisation and dimensionality reduction

In this Section we are going to introduce the concept of features extraction from multimedia content such audio files or images and we will present some post-processing techniques, like feature normalisation.

Moreover, we are going to describe few dimensionality reductions techniques, which will be used in order to visualise two different set of features extracted by the bicoherences , i.e. mean-kurtosis (MK) and mean-kurtosis-UNET (MKU) features, both in a 2D and a 3D space.

1.1.1 Features extraction and normalisation

Features extraction commonly refers to the set of techniques used to reduce the dimensions of signals (and therefore the amount of data that needs to be processed by a digital system) selecting and computing variables in order to maintain relevant information about the original data.

For example, if we have a signal consisting of temperature measurements in a room every minute, a possible feature that one may want to extract is the daily mean and variance of the temperature. This way we reduce the amount of data, keeping the relevant information about the temperature itself.

The most common features used in audio processing (and speech processing in particular) are basically three: the Mel-frequency cepstrum coefficients (MFCCs), the linear prediction cepstrum coefficients (LPCCs) and the spectrogram.

In order to extract these features, usually the time-domain audio signal $\boldsymbol{y}(k)$ is subjected to some pre-processing operations like segmentation and windowing. Segmentation consists in dividing the signal into small fragments, each one of which is processed on its own, while windowing is simply the multiplication of each segmented fragment for a window function; the most common window functions used in audio processing are rectangular window, Hamming window and Hann window.

Another common operation applied to audio signals, which is frequently involved as a fundamental step in feature extraction techniques, is the computation of the frequency representation of the signal itself, that is performed applying to the signal some implementation of the discrete Fourier transform (DFT), commonly referred to as fast Fourier transform (FFT).

Feature extraction is a crucial step in multimedia processing. How to extract ideal features that can reflect the intrinsic content of an image or audio signal as complete as possible is still a challenging problem in computer vision [5].

Once the desired features have been extracted, one may want to normalise them in order to change the range of values they assume. We are going to show two methods for features normalisation, which are the most commonly used in practice.

In the following we are going to use $\boldsymbol{x}$ and $\boldsymbol{X}$ to indicate respectively a vector or matrix that contains the features.

The first method is called min-max normalisation; it transform the features by scaling each feature to a given range as follows:

$$\boldsymbol{x}_{\text{scaled}} = a + \frac{(\boldsymbol{x} - \min(\boldsymbol{x}))(b - a)}{\max(\boldsymbol{x}) - \min(\boldsymbol{x})}, \tag{1.1}$$

where $\boldsymbol{x}$ is the original feature vector, $\boldsymbol{x}_{\text{scaled}}$ is the normalised vector, a and b are the minimum and maximum values of the interval of values that the normalised features are going to assume, and $\min(\boldsymbol{x})$ and $\max(\boldsymbol{x})$ are respectively the minimum and maximum values of the original signal.

Usually, for features to be used in machine learning applications, one wants $a = 0$ and $b = 1$, such that the features are going to assume values in the interval $[0, 1]$. In that case the transformation becomes

$$\boldsymbol{x}'_{\text{scaled}} = \frac{\boldsymbol{x} - \min(\boldsymbol{x})}{\max(\boldsymbol{x}) - \min(\boldsymbol{x})}. \tag{1.2}$$

Another possible normalisation technique is called Z normalisation; it yields values that belong to the interval $[0, 1]$, as follows:

$$\boldsymbol{x}''_{\text{scaled}} = \frac{\boldsymbol{x} - \mu}{\sigma}, \tag{1.3}$$

with

$$\mu = \frac{\sum_{i=1}^{N} x_i}{N} \tag{1.4}$$

and

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (x_i - \mu)^2}, \tag{1.5}$$

where N is the number of samples, μ, σ are respectively the mean and standard deviation, and x_i is the i-th sample of the feature vector.

In the proposed system, the main feature extraction step corresponds to the transformation of speech audio signal into the bicoherence. Indeed bicoherences are supposed to retain all the relevant information from the original audio signals that can be used to classify the various classes of speech with respect to the synthesis method used to generate them. We will show how bicoherences are defined and how to extract them in Chapter 2. Moreover, we will extract additional sets of features from the bicoherences and these features will be used to perform our closed set and open set classifications; this latter feature extraction is useful to further reduce the amount of data we are going to work with.

1.1.2 Dimensionality reduction techniques

Dimensionality reduction is the transformation of data from a high-dimensional space into a low-dimensional space such that the low-dimensional representation retains some meaningful properties of the original data.

Working in high-dimensional spaces can be undesirable, for example for computational complexity reasons. Moreover, working in a two or three-dimensional space can be useful for visualizing the data by means of scatter plots.

In the following, we are going to describe three different methods for dimensionality reduction: principal component analysis (PCA), independent component analysis (ICA) and t-distributed stochastic neighbour embedded (TSNE). These techniques will be used to reduce the dimensionality of mean-kurtosis (MK) and mean-kurtosis-UNET (MKU) features in order to be able to visualise them in a 2D and 3D space, and see

how the different classes of speech that we are going to classify cluster in these spaces.

1.1.2.1 PCA

Principal component analysis (PCA) is one of the oldest and most widely used techniques for dimensionality reduction. PCA consists in finding new variables that are linear functions of those in the original dataset, that successively maximize variance and that are uncorrelated with each other. Finding such new variables, also known as the principal components (PCs), reduces to solving an eigenvalue/eigenvector problem. PCA as a descriptive tool needs no distributional assumptions and, as such, is very much an adaptive exploratory method which can be used on numerical data of various types [6].

Suppose we have n individuals, and for each of them we acquire observations of p different variables. We can represent our initial dataset with an $n \times p$ data matrix $\boldsymbol{X}$ whose j-th column is the vector $\boldsymbol{x_j}$ of observations on the j-th variable.

We are looking for a linear combination of the columns of matrix $\boldsymbol{X}$ with maximum variance. Such linear combination is given by

$$\sum_{j=1}^{p} a_j \boldsymbol{x_j} = \boldsymbol{X}\boldsymbol{a}, \tag{1.6}$$

where $\boldsymbol{a}$ is a vector of constants $a_1, a_2 ... a_p$. The variance of any such linear combination is given by

$$\text{var}(\boldsymbol{X}\boldsymbol{a}) = \boldsymbol{a}^{\boldsymbol{T}}\boldsymbol{S}\boldsymbol{a}, \tag{1.7}$$

where $\boldsymbol{S}$ is the sample covariance matrix associated with the dataset and the symbol ^{T} denotes the transpose operation.

Hence, identifying the linear combination with maximum variance is equivalent to obtaining a p-dimensional vector $\boldsymbol{a}$ which maximizes the quadratic form $\boldsymbol{a}^{\boldsymbol{T}}\boldsymbol{S}\boldsymbol{a}$.

For this problem to have a well-defined solution, an additional restriction must be imposed, which involves working with unit-norm vectors, i.e. requiring

$$\boldsymbol{a}^{\boldsymbol{T}}\boldsymbol{a} = 1. \tag{1.8}$$

The problem is equivalent to solve the well known eigenvalue/eigenvector problem given by

$$\boldsymbol{S}\boldsymbol{a} - \lambda\boldsymbol{a} = \boldsymbol{0} \iff \boldsymbol{S}\boldsymbol{a} = \lambda\boldsymbol{a} \tag{1.9}$$

Thus, $\boldsymbol{a}$ must be a (unit-norm) eigenvector, and λ the corresponding eigenvalue, of the covariance matrix $\boldsymbol{S}$.

In particular, we are interested in the largest eigenvalue, λ_1 (and corresponding eigenvector $\boldsymbol{a_1}$), since the eigenvalues are the variances of the linear combinations defined by the corresponding eigenvector $\boldsymbol{a}$:

$$\text{var}(\boldsymbol{Xa}) = \boldsymbol{a^T Sa} = \lambda \boldsymbol{a^T a} = \lambda. \tag{1.10}$$

Any $p \times p$ real symmetric matrix, such as a covariance matrix $\boldsymbol{S}$, has exactly p real eigenvalues, λ_k $(k = 1, ..., p)$, and their corresponding eigenvectors can be defined to form an orthonormal set of vectors.

The full set of eigenvectors of $\boldsymbol{S}$ are the solutions to the problem of obtaining up to p new linear combinations

$$\boldsymbol{Xa_k} = \sum_{j=1}^{p} a_{jk} \boldsymbol{x_j}, \tag{1.11}$$

which successively maximize variance, subject to uncorrelatedness with previous linear combinations. These linear combinations $\boldsymbol{Xa_k}$ are called the principal components (PCs) of the dataset [6].

1.1.2.2 ICA

Another dimensionality reduction technique that we are going to introduce is independent component analysis (ICA). ICA is a linear transformation method in which the desired representation is the one that minimises the statistical dependence of the components.

ICA of the random vector $\boldsymbol{x}$ consists of finding a linear transform

$$\boldsymbol{s} = \boldsymbol{Wx} \tag{1.12}$$

such that the components s_i are as independent as possible, in the sense of maximising some function $F(s_1, ..., s_m)$ that measures independence.

We can formulate the problem also in a generative way as follows: ICA of a random vector $\boldsymbol{x}$ consists of estimating the following generative model for the data:

$$\boldsymbol{x} = \boldsymbol{As}, \tag{1.13}$$

where the components s_i in the vector $\boldsymbol{s} = (s_1, ..., s_n)^T$ are assumed independent, and the matrix $\boldsymbol{A}$ is a constant $m \times n$ "mixing" matrix. It can be shown that if the data follows the generative model, the Equations (1.12) and (1.13) become asymptotically equivalent, and the natural relation

$$\boldsymbol{W} = \boldsymbol{A^{-1}} \tag{1.14}$$

can be used with $n = m$.

The identifiability of the model can be assured by imposing the following fundamental restrictions:

- All the independent components s_i with the possible exception of one component, must be non Gaussian.
- The number of observed linear mixtures m must be at least as large as the number of independent components n, i.e. $n \geq m$.
- The matrix $\boldsymbol{A}$ must be of full column rank.

The estimation of the data model of independent component analysis (ICA) is usually performed formulating an objective function and minimising or maximising it.

An example of objective function is the likelihood: denoting by $\boldsymbol{W} = (\boldsymbol{w_1}, ...\boldsymbol{w_m})^T$ the matrix $\boldsymbol{A^{-1}}$, the log-likelihood takes the form

$$L = \sum_{t=1}^{T} \sum_{i=1}^{m} \log f_i(\boldsymbol{w_i^T x(t)}) + T \ln |det(\boldsymbol{W})|, \tag{1.15}$$

where the f_i are the density functions of the s_i (here assumed to be known), and the $\boldsymbol{x}(t), \boldsymbol{t} = 1, ..T$ are the realisations of $\boldsymbol{x}$ [7].

The log-likelihood can be maximised (for example using a classic gradient descent approach) in order to estimate the matrix $\boldsymbol{W}$, which can be used to compute the independent components s thanks to Equation (1.12).

1.1.2.3 TSNE

The last dimensionality reduction technique that we are going to describe is the t-distributed stochastic neighbour embedded (TSNE). TSNE has been proposed by L. van der Maaten and G. Hinton [8] in order to offer a technique to convert an high dimensional dataset $X = \{x_1, x_2, ..., x_n\}$ to a two or three dimensional dataset $Y = \{y_1, y_2, ..., y_n\}$ that can be displayed in a scatter plot.

First of all we are going to introduce the stochastic neighbour embedded (SNE), and then we are going to explain the modifications to obtain the t-distributed SNE.

SNE starts by converting the high-dimensional Euclidean distances between data points into conditional probabilities that represent similarities. The similarity of data point x_j to data point x_i is the conditional probability $p_{j|i}$ that x_i would pick x_j as its neighbour if neighbors were picked in proportion to their probability density under a Gaussian centered at x_i. For nearby data points $p_{j|i}$ is relatively high, whereas for widely separated data points, $p_{j|i}$ will be almost infinitesimal [8].

The conditional probability $p_{j|i}$ is given by

$$p_{j|i} = \frac{e^{\frac{-||x_i - x_j||^2}{2\sigma_i^2}}}{\sum_{k \neq i} e^{\frac{-||x_i - x_k||^2}{2\sigma_i^2}}}, \tag{1.16}$$

where σ_i is the variance of the Gaussian centered on data point x_i.

It is not likely that there is a single value of σ_i that is optimal for all data points in the dataset, because the density of the data usually varies. In dense regions, a smaller value of σ_i is usually more appropriate than in sparser regions. Since we are only interested in modeling pairwise similarities, we set the value of $p_{i|i}$ to zero.

For the low dimensional counterparts y_i and y_j of the high dimensional points x_i and x_j it is possible to compute a similar conditional probability, which we denote by $q_{j|i}$; we set the variance of the Gaussian involved in the computation of $q_{j|i}$ to $\frac{1}{\sqrt{2}}$, and we obtain the following:

$$q_{j|i} = \frac{e^{-||y_i - y_j||^2}}{\sum_{k \neq i} e^{-||y_i - y_k||^2}}. \tag{1.17}$$

Again, since we are only interested in modeling pairwise similarities, we set $q_{i|i} = 0$.

SNE aims to find a low dimensional data representation that minimizes the mismatch between $p_{j|i}$ and $q_{j|i}$.

A natural measure of the faithfulness with which $q_{j|i}$ models $p_{j|i}$ is the Kullback-Leibler divergence. SNE minimizes the sum of Kullback-Leibler divergences over all data points using a gradient descent method.

The t-distributed SNE, has been introduced to solve problems like the difficulty to optimize the objective function of SNE and the so called "crowding problem" (for more information see [8]).

The cost function used by TSNE differs from the one used by SNE in two ways:

- It uses a symmetrised version of the SNE cost function with simpler gradients
- It uses a student-t distribution rather than a Gaussian to compute the similarity between two points in the low-dimensional space

1.2 Machine learning techniques and classification algorithms

The field of machine learning is concerned with the question of how to construct computer programs that automatically improve with experience.

Most of machine learning processes can be divided into two phases: the learning phase and the performance measurement phase. The learning phase is commonly known as "training" (we say that a model is trained), and consists in learning new experience from data (which is called "training data" or "training set"); learning is the process of finding statistical regularities or other patterns of data [9].

The second phase consists in measuring the performance of the model after it has been trained; it is usually called "test", and is performed on

a dataset (called "test set") which has to be disjoint from the training set.

There are several types of learning, which differ w.r.t the way the model is trained. We are going to introduce four of them:

- The first one is the so called supervised learning, and consists of generating a function that maps the inputs to desired outputs, which are available during the training phase. The outputs are usually represented by labels on the training data [9].

- In opposition to supervised learning, we also have unsupervised learning, where the outputs are not available during the training phase. An example of unsupervised algorithm is clustering, which is the assignment of a set of observations into subsets (called clusters) such that observations within the same cluster are similar according to one or more predesignated criteria [10].

- Another form of learning is reinforcement learning (RL); in RL the algorithm learns a policy of how to act given an observation of the world. Every action has some impact in the environment, and the environment provides feedback that guides the learning algorithm. It has applications in many fields such as operations research and robotics [11].

- Finally, we have anomaly detection, which consists in analysing real world datasets in order to find which instances stand out as being dissimilar to all others. These instances are known as anomalies, and the goal of anomaly detection (also known as outlier detection) is to determine all such instances in a data-driven fashion. Anomalies can be caused by errors in the data, but sometimes they are indicative of a new, previously unknown, underlying process. In fact an outlier has been defined by Hawkins [12] as an observation that deviates so significantly from other observations as to arouse suspicion that it was generated by a different mechanism [13].

In the remaining of this Section, we are going to introduce the classification problem (which is one of the most famous tasks in the field of machine learning); after that, we will describe some models such as support vector machine (SVM), logistic regression (LR), random forest (RF), convolutional neural networks and convolutional autoencoder (and U-Net), which will be used in the following chapters to classify the features extracted from the bicoherences; moreover, we will discuss the problem of the open set classification. In the following Section we will introduce some metrics like confusion matrix, accuracy and ROC curve that are useful to measure the performance of a classification algorithm.

1.2.1 Classification problems

The goal of classification is to take an input vector $\boldsymbol{x}$ and to assign it to one of K discrete classes C_k for $k = 1, ..., K$. In the common scenario, the classes are taken to be disjoint, such that each input is assigned to one and only one class. The input space is thereby divided into decision regions, where boundaries are called "decision boundaries" or "decision surfaces" [14].

Here we consider linear models for classifications, which means that the decision surfaces are linear functions of the input vector $\boldsymbol{x}$.

1.2.1.1 Linear models for classification

Datasets whose classes can be separated exactly by linear decision surfaces are said to be linearly separable. For two-class (or binary) problems, we can define a single target variable $t \in [0, 1]$ such that $t = 1$ represents class C_1 and $t = 0$ represents class C_2. We can represent t as the probability that the class is C_1, where t assumes only the extreme values of 0 and 1.

For $K > 2$ classes, we can define $\boldsymbol{t}$ as a vector of length K such that if the class is C_j, then all the elements t_k of $\boldsymbol{t}$ are zero except element t_j, which takes value 1. For instance, if we have $K = 5$ classes, then a pattern from class 2 would be given the vector

$$\boldsymbol{t} = (0, 1, 0, 0, 0)^T. \tag{1.18}$$

There are two different approaches to the classification problem. The simplest one involves constructing a "discriminant function" that directly assigns each vector $\boldsymbol{x}$ to a specific class.

A more powerful approach, however, models the conditional probability distributions $p(C_k|\boldsymbol{x})$ and uses these distributions to make optimal decisions. In order to determine the probability distributions $p(C_k|\boldsymbol{x})$ we can proceed in two ways. One way is to model them directly, for example by representing them as parametric models and then optimising the parameters using a training set (also known as probabilistic discriminative approach).

The other way is to adopt a probabilistic generative approach, in which we model the class conditional densities given by $p(\boldsymbol{x}|C_k)$ together with the prior probabilities $p(C_k)$ for the classes, and then we compute the required conditional probabilities using the Bayes theorem

$$p(C_k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|C_k)p(C_k)}{p(\boldsymbol{x})}. \tag{1.19}$$

We can model a prediction associated with input $\boldsymbol{x}$ as

$$y(\boldsymbol{x}) = f(\boldsymbol{w^T x} + w_0), \tag{1.20}$$

where $f()$ is a nonlinear function known as "activation function", $\boldsymbol{w}$ is the vector of coefficients to be determined called "weight vector" and w_0 is a "bias" [14].

1.2.2 Support vector machine

In this section and the following ones we are going to describe some machine learning techniques that are useful to solve the classification problem and that we are going to use to classify the different classes of speech. We start from support vector machine (SVM).

Let us consider a linear model for the two-class classification problem:

$$y(\boldsymbol{x}) = \boldsymbol{w^T}\boldsymbol{\phi(x)} + b, \tag{1.21}$$

where $\boldsymbol{\phi(x)}$ denotes a fixed feature-space transformation, and b is the bias.

The training data set is composed by N input vectors $\boldsymbol{x_1}, ..., \boldsymbol{x_N}$ with corresponding target values $t_1, ..., t_N$ where $t_n \in [-1, 1]$, and new data points $\boldsymbol{x}$ are classified according to the sign of $y(\boldsymbol{x})$.

We shall assume for the moment that the training dataset is linearly separable in the feature space.

The support vector machine (SVM) approaches the problem through the concept of the "margin", which is defined to be the smallest distance between the decision boundary and any of the samples. The decision boundary is chosen to be the one for which the margin is maximised.

The distance of a point $\boldsymbol{x_n}$ to the decision surface is given by

$$\frac{t_n y(\boldsymbol{x_n})}{||\boldsymbol{w}||} = \frac{t_n(\boldsymbol{w^T}\boldsymbol{\phi(x)} + b)}{||\boldsymbol{w}||}. \tag{1.22}$$

The margin is given by the perpendicular distance of the decision boundary to the closest point $\boldsymbol{x_n}$ of the dataset, and we wish to optimise the parameters $\boldsymbol{w}$ and b in order to maximise this distance.

Thus the maximum margin solution is found by solving

$$\arg\max_{\boldsymbol{w},b} \left\{ \frac{1}{||\boldsymbol{w}||} \min_n [t_n(\boldsymbol{w^T}\boldsymbol{\phi(x)} + b)] \right\}. \tag{1.23}$$

Direct solution of this optimisation problem would be very complex, and so we shall convert into an equivalent problem that is much easier to solve. To do this, we set

$$t_n(\boldsymbol{w^T}\boldsymbol{\phi(x)} + b) = 1 \tag{1.24}$$

for the point that is closest to the surface. In this case all data points will satisfy the constraints

$$t_n(\boldsymbol{w^T}\boldsymbol{\phi(x)} + b) \geq 1, \quad n = 1, ..., N. \tag{1.25}$$

This is known as the canonical representation of the decision hyperplane.

The optimisation problem then simply requires that we maximise $||\boldsymbol{w}||^{-1}$, which is equivalent to maximising $||\boldsymbol{w}||^2$, and so we have to solve the optimisation problem

$$\arg\min_{\boldsymbol{w},b} \frac{1}{2}||\boldsymbol{w}||^2 \tag{1.26}$$

subject to the constraints given by Equation (1.25).

This is an example of a quadratic programming problem, in which we are trying to minimise a quadratic function subject to a set of linear inequality constraints [14].

The points that lie on the maximum margin hyperplanes in feature space are called support vectors, and are the only one that will be retained in order to classify new data points. All other points, once the model is trained, can be discarded. This property is fundamental for the practical application of SVMs.

1.2.3 Logistic regression

The next classification model we discuss is logistic regression (LR), which is a probabilistic discriminative model.

In order to do that, instead of working with the original input vector $\boldsymbol{x}$, we will work in the feature space using $\boldsymbol{\phi}(\boldsymbol{x})$, which is a nonlinear transformation of the inputs.

All algorithms are equally applicable in the feature space. The resulting decision boundaries will be linear in the feature space $\boldsymbol{\phi}$, but will correspond to nonlinear decision boundaries in the original $\boldsymbol{x}$ space. Classes that are linearly separable in the feature space $\phi(\boldsymbol{x})$ need not be linearly separable in the original observation space $\boldsymbol{x}$.

Consider a two-class problem. We model the posterior probability of class C_1 as

$$p(C_1|\boldsymbol{\phi}) = y(\boldsymbol{\phi}) = \sigma(\boldsymbol{w^T\phi}) \tag{1.27}$$

with $p(C_2|\boldsymbol{\phi}) = 1-p(C_1|\boldsymbol{\phi})$. Here the σ is the logistic sigmoid function defined by

$$\sigma(a) = \frac{1}{1+e^{-a}}. \tag{1.28}$$

This model is known as logistic regression (LR). For an M-dimensional feature space $\boldsymbol{\phi}$, this model has M adjustable parameters.

We can use maximum likelihood to determine the parameters of the logistic regression model. We can define an error function by taking

the negative logarithm of the likelihood, which gives the "cross-entropy" error function in the form

$$E(\boldsymbol{w}) = -\ln p(\boldsymbol{t}|\boldsymbol{w}) = -\sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\}, \qquad (1.29)$$

where $y_n = \sigma(a_n)$, and $a_n = \boldsymbol{w^T \phi_n}$.

For logistic regression there is no closed-form solution, due to the nonlinearity of the logistic sigmoid function. However, the error function is concave, and so it has a unique minimum.

This minimum can be found using an efficient iterative technique based on the Newton-Raphson iterative optimisation scheme. The algorithm that uses the Newton-Raphson iterative optimisation scheme to solve the LR problem is also known as iterative reweighted least squares. Additional information can be found in [14].

1.2.4 Decision tree and Random Forest

The next classification technique that we will introduce is called RF. We will first describe decision trees, and then we will explain how to combine them to obtain the RF.

1.2.4.1 Decision tree

Decision trees classify instances by sorting them down a tree from the root to some leaf node, which provides the classification of the instance. Each node of the tree specifies a test of some attribute of the instance, and each branch descending from that node corresponds to one of the possible values of this attribute. An instance is classified by starting from the root of the tree, testing the attribute specified by this node and then moving down the tree branch corresponding to the value of the attribute. This process is then repeated for the subtree rooted at the new node.

Decision tree learning is best suited to problems with the following characteristics:

- Instances are described by a fixed set of attributes and their values. The easiest situation for a decision tree classifier is when each attribute takes on a small number of disjoint possible values. However, extensions to the basic algorithm allow handling real value attributes as well
- The target function has discrete output values

Most algorithms that have been developed for learning decision trees are variations on a core algorithm that employs a top-down greedy search through the space of possible decision trees [15]. Some examples are the ID3 and C4.5 algorithms.

1.2.4.2 Random forest

Significant improvements in classification accuracy have resulted in growing an ensemble of trees and letting them vote for the most popular class. In order to grow these ensembles, often random vectors are generated, that govern the growth of each tree in the ensemble.

In particular, for the k-th tree, a random vector θ_k is generated, independently on the past random vectors $\theta_1 ... \theta_{k-1}$, but with the same distribution; and a tree is grown using the training set and θ_k, resulting in a classifier $h(\boldsymbol{x}, \theta_k)$, where $\boldsymbol{x}$ is an input vector.

After a large number of trees is generated, they vote for the most popular class. We call these procedures random forests. More specifically, a RF is a classifier consisting of a collection of tree-structures classifiers $\{h(\boldsymbol{x}, \theta_k), k = 1...\}$ where the $\{\theta_k\}$ are independent identically distributed random vectors and each tree casts a unit vote for the most popular class associated with input $\boldsymbol{x}$ [16].

1.2.5 Convolutional neural networks

The next classification model we discuss is called artificial neural network or simply neural network (NN). NNs are also known as "multilayer perceptrons", which anyway is a quite misleading name: indeed, even if they are actually composed by many layers, each one of these layers can be seen as a logistic regression (LR) (instead of a perceptron).

The resulting model is significantly more compact and faster to evaluate than a SVM having the same generalisation performance. However, the likelihood function, which forms the basis for the network training, is no longer a convex function of the model parameters.

With neural networks, we fix the number of basis functions in advance, but we allow them to be adaptive, which means that we use parametric forms for the basis functions in which the parameters values are adapted during training [14].

We start from the following linear classification model

$$y(\boldsymbol{x}, \boldsymbol{w}) = f\left(\sum_{j=1}^{M} w_j \phi_j(\boldsymbol{x})\right), \tag{1.30}$$

where $f()$ is a nonlinear activation function. Our goal is to extend this model by making the basis functions $\phi_j(\boldsymbol{x})$ depend on parameters, and then to allow these parameters to be adjusted, along with the coefficients $\{w_j\}$ during training.

Neural networks use basis functions that follow the same form as Equation (1.30), such that each basis function is itself a nonlinear function of linear combinations of the inputs, where the coefficients in the linear combinations are adaptive parameters.

In order to derive the basic neural network (NN) model, which can be described as series of functional transformations, first we construct M

linear combinations of the input variables $x_1, ..., x_D$ of the form

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}, \tag{1.31}$$

where $j = 1, ..., M$ and the superscript $^{(1)}$ indicates that the corresponding parameters are in the first layer of the network. The parameters $w_{ji}^{(1)}$ are called "weights", while the parameters $w_{j0}^{(1)}$ are the "biases".

The quantities a_j are known as "activations". Each of them is then transformed using a differentiable nonlinear activation function $h()$ to give

$$z_j = h(a_j). \tag{1.32}$$

These quantities correspond to the output of the basis functions, and are called "hidden units". The nonlinear functions $h()$ are generally chosen to be sigmoidal functions such as the logistic sigmoid.

The values z_j are then again linearly combined to give "output unit activations"

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)} \tag{1.33}$$

where $k = 1, ..., K$ and K is the total number of outputs. This transformation corresponds to the second layer of the network.

Finally, the output unit activations are transformed using a logistic sigmoid function for multiple binary classification problems or a softmax activation function for multiclass problems.

We can combine these various stages to give the overall network function that, for sigmoidal output unit activation functions, takes the form

$$y_k(\boldsymbol{x}, \boldsymbol{w}) = \sigma \left(\sum_{j=1}^{M} w_{kj}^{(2)} h \left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)} \right) \tag{1.34}$$

where the sets of all weight and bias parameters have been grouped together into a single vector $\boldsymbol{w}$. Thus the neural network model is simply a nonlinear function from a set of input variables $\{x_i\}$ to a set of output variables $\{y_k\}$ controlled by a vector$\boldsymbol{w}$ of adjustable parameters. The process of evaluating Equation (1.34) is called "forward propagation" of information through the network. This model can be easily generalised adding more layers of processing, where each layer consists in a weighted linear combination followed by an element-wise transformation using a nonlinear activation function [14].

A convolutional neural network (CNN) is a neural network which comprises one or more convolutional layers. This kind of networks are particularly suitable to work with image data. In the convolutional layer

the units are organised into planes, each of which is called a "feature map". Units in a feature map each take inputs only from a small subregion of the image, and all the units in a feature map are constrained to share the same weight values. Inputs values are linearly combined using the weights and the bias, and the result transformed by a sigmoid nonlinearity using Equation (1.30).

If we think of the units as feature detectors, then all of the units in a feature map detect the same pattern, but at different locations in the input image. Due to weight sharing, the evaluation of the activations of these units is equivalent to a convolution of the image pixel intensities with a "kernel" comprising the weight parameters.

The outputs of the convolutional units form the inputs to the subsampling layer (also known as pooling layer) of the network. For each feature map in the convolutional layer there is a plane of units in the subsampling layer and each unit takes inputs from a small receptive field in the corresponding feature map of the convolutional layer. These units perform subsampling. In a practical architecture there may be several pairs of convolutional and subsampling layers [14]. In Figure 1.1 is shown an example structure of a convolutional neural network used to perform a multiclass classification task.

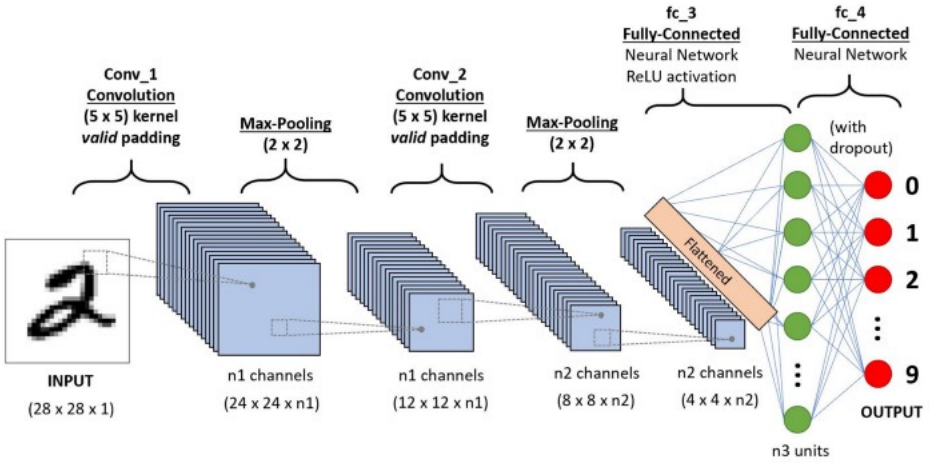


Figure 1.1: Example structure of a convolutional neural network used for classification. Taken from https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

In Appendix A you can find the results of both binary and multiclass classifications of the modules of the bicoherences (treated as images) using a simple convolutional neural network.

1.2.6 Convolutional autoencoder and U-Net

A convolutional autoencoder is a convolutional neural network, generally used to work with image data, which can be divided into two parts: an

"encoder" and a "decoder". The encoder part of the network takes the image as input and gives as output a compressed version of the image itself, while the decoder takes as input the compressed version (i.e. the output of the encoder) and gives as output a reconstructed image which is usually similar to the original one (See Figure 1.2).

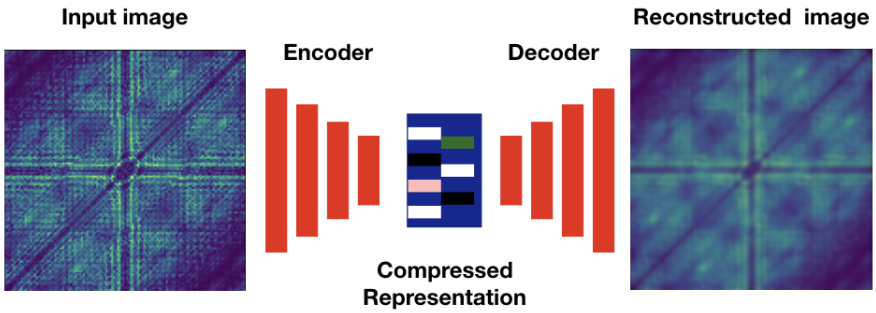


Figure 1.2: Autoencoder structure.

The key aspect of autoencoders is that they can reconstruct better the class of images used in the training phase. For example, if an autoencoder is trained with images of dogs, than it will perform a better reconstruction when tested with images of dogs instead of images of other animals.

Usually the autoencoder is used for features extraction, anomaly detection or image segmentation. In [17] robust autoencoders used for anomaly detection are described and compared to PCA. In Chapter 3 we are going to describe a method for features extraction , while in Appendix B we will describe a method for anomaly detection. Both of these methods involve the use of a complex convolutional autoencoder called U-Net, which has been initially used in [18] for image segmentation. The main difference between the U-Net and a traditional autoencoder is that in the U-Net some skip-connection (or concatenation) layers are present, which link deeper layers of the network to more shallow ones.

1.2.7 Closed set vs open set classification

We now introduce the problem of open set classification. In an open set environment, the aim is to both discriminate among some "known" classes, which are used to train the model, and to classify as "unknown" instances of the input space belonging to classes which have never been seen by the model during the training phase.

Following the definition in [19], three different kinds of data are involved in an open set classification:

- Known data: data belonging to known classes that the model has to correctly detect and classify; this is used for both training and test

- Known-unknown data: data available at training time but assumed as unknown in order to model unknown data at training time; this is used for both training and test
- Unknown-unknown Data: data only used to test the model with instances belonging to classes which have never been seen during the training phase.

In [20] you can find a method for open set classification that involves the use of neural networks.

In Chapter 3 we will describe a method to implement an open set environment using a series of support vector machines.

1.3 Confusion matrix, accuracy and ROC curve

In this Section we will introduce confusion matrices, accuracy and ROC curves, which are useful metrics when it comes to evaluate the performance of a classification algorithm.

1.3.1 Confusion matrix

Usually, when a classification has been performed, for each input $\boldsymbol{x}_t$ belonging to the test set, two values are available: the true class it belongs t to and the predicted class that it has been assigned to $\hat{t}$.

From this information, we can build a table that allows visualization of the performance of the classification algorithm. In this table, each row represents the instances of the predicted class, while each column represents the instances of the actual class. For the case of two-class problems, we have a 2×2 matrix as in Figure 1.3, where the two classes are indicated as positive class (P) and negative class (N).

The elements that are contained in the matrix are the following:

- True positive (TP): is the number of instances of the input vector $\boldsymbol{x_t}$ that belong to the positive class and were correctly classified as positive.
- False positive (FP): is the number of instances of the input vector $\boldsymbol{x_t}$ that belong to the negative class and were misclassified as belonging to the positive class.
- False negative (FN): is the number of instances of the input vector $\boldsymbol{x_t}$ that belong to the positive class and were misclassified as belonging to the negative class.
- True negative (TN): is the number of instances of the input vector $\boldsymbol{x_t}$ that belong to the negative class and were correctly classified as negative.

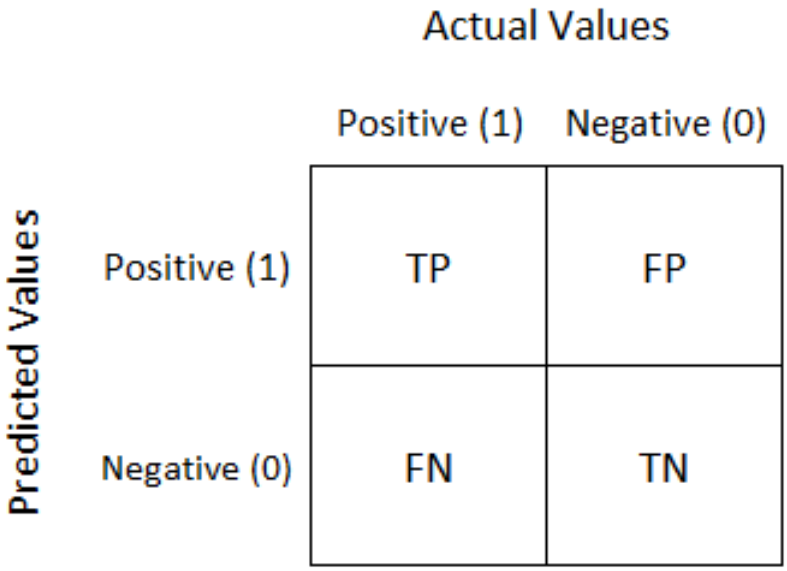


Figure 1.3: Confusion matrix for a two-class classification problem

In order to have a balanced matrix, it is recommended to test the model with an equal number of positive and negative instances. In that case we have the the sum of the elements of each column is constant.

A confusion matrix can be normalised dividing the elements of each column by the total number of elements belonging to each class. If the matrix is normalised, the elements in the matrix will assume values in the interval $[0, 1]$, and moreover, we have that the sum of the elements of each column is exactly 1. Confusion matrices can also be used for classifications involving more than two classes. In that case we have a row (and a column) for each one of the classes.

1.3.2 Accuracy and other metrics

A number of other metrics are available that give an estimation of how well we are classifying the instances in the input vector $\boldsymbol{x_t}$. In the following we are going to show four of them (i.e. accuracy, precision, recall and F1 score), and we consider the quantities defined in the previous section:

- **Accuracy**: The accuracy is defined as

$$ACC = \frac{TP + TN}{N}, \tag{1.35}$$

  where N is the number of total instances in the input vector $\boldsymbol{x_t}$ in case the confusion matrix is not normalised, while it is the number of classes in case of a normalised confusion matrix.

- **Precision**:

$$PRE = \frac{TP}{TP + FP}. \tag{1.36}$$

- **Recall**:

$$REC = \frac{TP}{TP + FN}. \tag{1.37}$$

- **F1 score**:

$$F1 = \frac{2 \cdot PRE \cdot REC}{PRE + REC}. \tag{1.38}$$

The highest these metrics are, the better the model performs in the classification task. All of them take values belonging to the interval $[0, 1]$. In the following chapters we are going to use accuracy to evaluate the performance of our classifications.

1.3.3 ROC curve

Another method, together with confusion matrices, which is useful to visualise the performance of a classification algorithm, and that we are going to use in the next Chapters, is called receiver operating characteristic (ROC) curve. This is used specifically for binary classifications. The curve is a plot of the true positive rate (TPR), defined as

$$TPR = \frac{TP}{p}, \tag{1.39}$$

where p is the number of instances belonging to the positive class P, versus the false positive rate (FPR), defined as

$$FPR = \frac{FP}{n}, \tag{1.40}$$

where n is the number of instances belonging to the negative class N.

Usually, given a ROC curve, it is desirable to compute the area under curve (AUC), which is a value that belongs to the interval $[0, 1]$ that represents the area that lies under the curve, and is a metric of the probability that an instance is correctly classified.

2
State of the Art

In this Chapter, after an introduction on multimedia forensics techniques, we will describe text to speech (TTS) and voice conversion (VC) algorithms used to synthesise speech. Moreover, we will present few methods in current literature that can be used to distinguish between real and synthesised speech, making the distinction between methods that use traditional features and methods that employ data driven features. Among the described methods, a couple of them are based on traditional feature extraction (in particular they make use of the bicoherence and the spectrogram, respectively, as features given as input to the classifiers), while in the last method also data driven features are involved.

2.1 Overview on Multimedia Forensics

Multimedia forensics is a scientific field that comprehends many techniques useful to acquire information from multimedia content such as audio, images and video. This information may concern the identification of the acquisition device that generated the content, the evaluation of the integrity of the signal or characterising the environment where the signal was recorded, but could also be about detecting digital tampering on the signal or recognising computer generated content from natural one.

The key idea behind multimedia forensics is that acquisition, coding (for example compression) and editing on a signal leave intrinsic traces (also called digital footprints) on the signal itself, which can be detected using specific methods.

Methods to gather information on the life-cycle and truthfulness of some multimedia content can be classified into two classes: active and

passive methods.

Active methods are usually based on the use of a "watermark" [21] [22] [23] or a "digital signature" [24] [25]; these are left by the device (digital camera or microphone) at the moment of the acquisition, before the content is compressed and stored on a hard disk, and can be analysed.

Passive methods, on the contrary, in order to be applied, only need the content itself, and are based on the assumption that, as we said above, all digital operations applied to a signal leave traces on it. Passive methods are the object of study of multimedia forensics [26].

For the case of images (and video), during the acquisition phase, footprints are left by the lenses, the camera sensor (usually a CCD or CMOS) and the color filter array (which is a slight film on the sensor that allows a certain component of the light to get through and reach the sensor) [27].

The lenses produce many types of aberrations (such as the lateral chromatic aberration) which leave unique traces that can be used to link an image to the camera that has taken it [28] [29] [30] or to reveal modifications to the image itself [31] [32] ; the camera sensor produces sensor pattern noise, whose main component is the photo response non uniformity (PRNU), which is unique for each camera and can be used to link images to their source [33]; finally, the color filter array selects a single color (red, green or blue) for each pixel, such that, in order to obtain the missing pixel values, some kind of interpolation (also known as demosaicing) has to be performed, and this operation leaves traces on the image as well [27].

Image coding also leaves digital footprints on the image, which can be used to reveal digital forgery [34] [35]: indeed compression (usually using the JPEG format) is a "lossy" operation, which means that information is lost during the compression, and it is not a reversible operation.

Finally, also editing (i.e. post-processing) on an image can be detected: in order to do so, one can work at signal level (to see if there are artifacts on the signal due to the modifications) or at scene level (analysing shadows, lights, reflections, perspective, and geometry of objects) [27].

As concerns audio forensics, there are techniques to associate an audio recording to the microphone that generated it, to acquire information about the time a recording was made and even to retrieve information about the environment where the signal has been recorded, all of this exploiting footprints left at the time the audio signal has been acquired. Compression of an audio signal leaves footprints that can be detected in order to obtain information such as the original bit-rate of the signal. It is also possible to obtain information about editing operations that have been performed after the signal has been created, such as removal or insertion of audio data into the signal (cut-type edits) or signal processing operations like applications of nonlinear effects or mixing of audio signals.

In [36] we can find results of classifications of audio recordings w.r.t.

the microphone they where acquired with: these results yield that the set of features that can better discriminate the recordings is the second derivative of the Mel-frequency cepstrum coefficients (MFCCs).

In order to compute the time when an audio recording has been made we can make use of the electric network frequency (ENF) [37] [38], which is the frequency of the power adapter (that corresponds to 50 Hz in Europe and 60 Hz in the United States); the variations of the ENF measured over a significantly long time form a unique signature that can be used to determine the acquisition time of an audio signal; ENF information is introduced into the audio signals both if the acquisition device is directly connected to the power grid (thanks to non ideal voltage controllers and magnetic interferences within the device) and also in cases of portable devices (since the electromagnetic field of nearby supply lines superimpose on the audio recording).

The information that can be retrieved from an audio signal about the environment where the recording has been taken mainly consists in two specific quantities i.e. the reverberation time [39] and the room impulse response [40]: both of them can give an idea of the dimensions of the room.

Audio, just like images and video, is often subjected to compression; the most common audio coders are MPEG AAC and MPEG-1/2 layer 3 (also known as MP3). Both of them belong to the class of perceptual audio coders. Footprints are left by the coding because of the presence of the quantization; the estimation of the quantization step, together with the number of steps used in the quantization can be useful to understand whether an audio recording has been previously encoded at a lower bit rate before it was re-encoded at a higher bit rate (so called "fake quality" detection) [41].

In order to detect cut-type editing on an audio signal, there are various approaches: detection of discontinuities of the signal in the time domain [42], detecting changes of audio power in successive frames of the signal (fuzzy clustering) [43], detecting large variations of the short-time energy in the background noise region of a signal [44] or detecting discontinuities in the features introduced by lossy audio compression [45] or on the electric network frequency (ENF) [46].

Moreover, there are means to detect mixing parameters using a least-square technique [47].

Finally, with the use of bispectral analysis, it is possible to detect nonlinear processing on audio signals, and in particular to distinguish bonafide from spoofed speech. Indeed, spoofing of a speech signal leaves in the signal itself higher order correlations, which can be detected by bispecrtral analysis, since the bicoherence is a third order feature. The result is that there is a difference between the bicoherences extracted from bonafide signals w.r.t the ones extracted by spoofed speech recordings, and these differences can be detected in an automatic way by means of artificial intelligence (AI) and machine learning techniques [1] [4].

In the next Chapters we will show and discuss in detail methods to recognise digital forgery on speech signals and we will propose our own method and show the results of its application on a wide and heterogeneous dataset of voices.

2.2 Speech Synthesis: text to speech and voice conversion algorithms

Voice is one of the most natural and straight-forward means to perform biometric person recognition. A system which is able to recognise a person using its voice is called automatic speaker verification (ASV) system. Like all other biometric systems, an ASV system is vulnerable to spoofing, which is also known as "presentation attacks" [48]. The most common presentation attacks are impersonation, replay and speech synthesis.

Impersonation consists in a person modifying its voice in order to make it resemble someone else voice.

Replay happens when a recording of someone's voice is played and presented to the ASV system interface instead of a voice produced by that person talking in that moment.

Our main interest here will be speech synthesis algorithms, which consist of generating a voice by means of computer systems and artificial intelligence techniques.

Speech synthesis algorithms can be mainly divided into two classes: text to speech (TTS) and voice conversion (VC) algorithms. The conceptual difference between these two classes of algorithms is what is given as input to the algorithm itself; while text to speech techniques take as input some text and give as output a voice which "reads the text aloud", voice conversion techniques take as input a voice and transform it to make it seem as it was produced by another speaker.

2.2.1 Text to speech synthesis

Following the description in [49], in general, a text to speech (TTS) synthesizer is composed by two different modules. The first module is a natural language processing (NLP) module, which must be able to produce a phonetic transcription of the text to be read. The second module is a digital signal processing module, which transforms the symbolic information received by the first module , together with information about the desired intonation and rhythm, and produces natural sounding speech. We will now analyse each one of these two modules.

2.2.1.1 Natural language processing module

The natural language processing (NLP) module is in turn composed by the following sub-modules (as shown in Figure 2.1): a morpho-syntactic

analyser (MSA), a letter-to-sound (LTS) module and a "natural" prosody generator (PG).

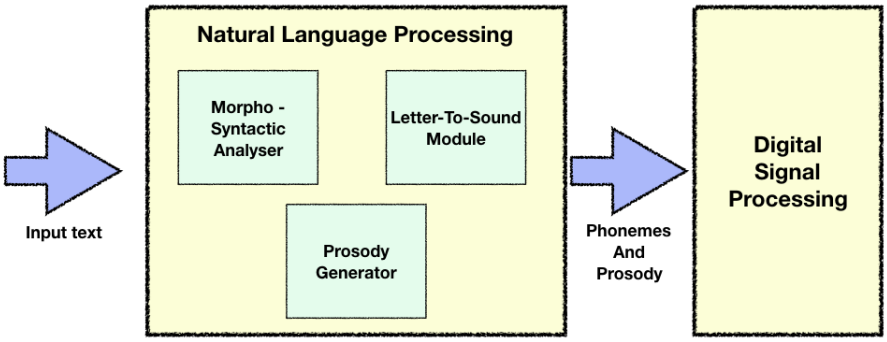


Figure 2.1: General architecture of a text to speech synthesiser.

In order to generate high quality speech, the latter two modules need the MSA module; indeed accurate phonetic transcription may depend on knowing the dependency relationship between successive words, and natural prosody heavily relies on syntax [49].

The morpho-syntactic analyser (MSA) module takes care of reducing a given sentence to the sequence of its parts of speech (i.e. nouns, pronoun, verb, adjective, ecc.) and describing it in the form of a syntax tree.

The letter-to-sound (LTS) module is used to produce a phonetic transcription of the text; we remark here that this operation is not as simple as it seems, since a single character can correspond to two phonemes or none, several characters may produce a single phoneme, a character can be pronounced in different ways as a function of the preceding and following characters and finally a single phoneme can result from several spellings. Phonetic transcription techniques are generally classified into two classes: dictionary-based and rules based techniques.

The prosody generator (PG) module, finally, is responsible for the production of melodic contours and rhythmic patterns in speech, which make it seem as natural as possible [49].

2.2.1.2 Digital signal Processing module

The digital signal processing module involved in a text to speech synthesizer has the role of producing the actual audio waveform associated to the speech signal starting from the information about the phonemes and prosody of the speech given by the NLP module.

Classical techniques based on the vocal tract model are mainly three: formant synthesis, linear prediction synthesis and articulatory synthesis.

Formant synthesis makes use of individually controllable formant filters which can produce an accurate estimation of the vocal tract transfer

function. Voiced sounds are produced using a train of impulses, while obstruent sounds are produced with a noise signal as input. It is possible to derive the parameters of the formant synthesiser thanks to a set of rules derived by the analysis of phones characteristics and context, even if there is an intrinsic difficulty in tuning the parameters of the formant synthesiser.

The linear prediction technique is similar to formant synthesis, except from the fact that all the sounds are generated by an all pole filter, while formant synthesis usually makes use of parallel filters. The peculiarity of linear prediction is that it acquires vocal tract parameters estimating them from data.

Articulatory synthesis models the human articulatory behaviour. It is subjected to difficulties in acquiring data to determine rules and models.

These techniques all produce intelligible but not natural sounding speech.

A second generation of techniques is inspired by the data driven approach of linear prediction synthesis. The main difference lays in the fact that in this case also the source waveform is generated in a data driven fashion. The concept behind these techniques is to work with a set of items, where each item is composed by a phone description, a fundamental frequency and a duration associated to that phone.

Typically, for each phonetic specification, a unique recorded speech unit is available, and the synthesis works simply retrieving these units from a database and concatenating them. Sometimes, instead of working with phonemes, these techniques make use of "diphones", which are composed by the second half of a phone together with the first half of the following one.

What changes from a second generation technique to the other is the way the pitch and timing of each phoneme is modified over time. The techniques used to do that are the following: pitch synchronous overlap and add (PSOLA) , residual excited linear prediction, sinusoidal models, harmonic noise models and Mel-frequency cepstrum coefficients (MFCCs) synthesis.

These second generation techniques are also known as "synthesis by concatenation" approaches, since they simply put together recorded pieces of audio, one for each phoneme.

More recently, with the use of larger databases, it was possible to select more appropriate speech units that match both phonemes and other linguistic contexts such as lexical stress, pitch accent, and part-of-speech information in order to generate high quality natural sounding synthetic speech with appropriate prosody [50]. These methods are called "unit selection" techniques, and currently are among the most widely used.

Another data driven approach, other than concatenative synthesis, which has gained much popularity, is to use statistical, machine learning techniques to infer the mapping between phoneme specification and the

associated parameters. There are many possibile approaches to statistical synthesis; anyway, most of the work has been focused on the use of hidden Markov models (HMMs) as generative models [51].

In HMM-based speech synthesis, the HMM models both the phoneme sequences and also various contexts of the linguistic specification in a similar way to the unit selection approach. Acoustic parameters generated from HMMs, selected according to the linguistic specification, are used to drive a vocoder, which is a simplified speech production model, in which speech is represented by vocal tract parameters and excitation parameters, in order to generate a speech waveform [50].

Hidden Markov models can deliver high quality speech, which also takes into account the expressivity and emotions of the speaker, while unit selection methods restricts the output speech to the same style as that in the original recordings as no (or few) modifications to the selected pieces of recorded speech are normally done [50].

Some recent vocoders used to produce the speech audio waveform, are based on the use of convolutional neural networks. An example is Google Wavenet (see [52]), which achieves state of the art performances implementing layers of the network which perform causal convolutions.

2.2.2 Voice conversion synthesis

Voice conversion (VC) commonly refers to the set of techniques used to convert the speaker identity. The input of a VC system is speech from a source speaker, while the output is speech which must be perceived as being generated by a target speaker different from the source one.

Generally, a database of speech produced by both the source and target speaker must be available to use in the training of the model.

Following the description in [53], a VC algorithm consists of three phases: the analysis phase, the mapping phase and the synthesis phase.

The analysis phase consists in extracting features from both source and target speech. These features usually have to represent the factors of speech which are speaker-dependent, and should allow easy modification of the perceptually important characteristics of speech as well as to provide high-quality waveform resynthesis [54]. There are many possible choices for the selection of these parameters. One possibility is to use the formants involved in the source-filter model of the vocal tract, as we said for TTS synthesis; other common choices are the line spectral frequency derived by linear prediction methods or the Mel-frequency cepstrum coefficients (MFCCs); another alternative is to use a sinusoidal model, which consists in estimating frequency and phases of a series of time varying sinusoids.

The mapping phase consists in mapping the features extracted from the source speaker to those extracted from the target speaker. There are few possible approaches to do that. The currently most popular choice is to use a Gaussian mixture model (GMM), which is trained with

the extracted parameters, and usually models the joint density between source and target features; another approach is to train a codebook of combined features vectors; finally, another possibility is to use a frequency warping, which involves generating a function between the source and target spectra.

Once the model has been trained, the third phase consists in generating new target speech associated to source speech which has not been used to train the model.

2.3 Recognising real speakers from AI synthesised speech

In this Section we are going to describe few state of the art methods that can be used to recognise bonafide speech from AI generated speech. We make here the distinction between two different classes of such methods, that differentiate from each other as concerns the features that are involved in the classification of bonafide and fake speech.

The first class comprehends methods based on classical features that are computed from the input audio files. Examples of these features are the Mel-frequency cepstrum coefficients (MFCCs), Q cepstral coefficients, spectrogram, linear prediction cepstrum coefficients (LPCCs), and also the bicoherence itself. We will describe here two of these methods: one based on bispectral analysis, which is the one that inspired the work of this thesis, and one that makes use of the spectrogram of the signal in order to perform a classification.

The second class of methods are the one based on features that are learned from data. These methods make use of some kind of machine learning techniques, which exploit the data available in the training phase to dinamically extract features, which vary w.r.t. the chosen model.

Ususally the best classification performances are achieved by combining both the classical features and the data driven generated ones, as happens for example for the experiments performed in [55], where convolutional autoencoders are used to extract data based features, and whose method is going to be detailed shortly.

We state here that the methodology that we have used to perform our classifications, which will be described in Chapter 3, also makes use of both classical features, i.e. mean-kurtosis (MK) features, and data driven features extracted using a U-Net (UNET features). In our case, the best performances have been achieved by combining these two sets of features by means of a simple concatenation, obtaining this way the mean-kurtosis-UNET (MKU) features. Indeed, MKU features are the ones that perform better on both closed set and open set classifications of the speech signals.

2.3.1 Methods based on classical feature extraction

As we said above, there are many possible sets of traditional features which can be used when working on speech signals (and in particular for the purpose of distinguishing between bonafide and AI generated speech). Among the most commonly used we find the Mel-frequency cepstrum coefficients (MFCCs), which take into account a perceptual model of the human ear, and can be derived from the spectrum of the signal obtained by means of a discrete Fourier transform (DFT), applying a triangular filter bank. Another popular feature is the spectrogram, obtained from the signal applying a short-time Fourier transform. Linear prediction cepstrum coefficients (LPCCs) are still very common, and are computed from the smoothed auto-regressive power spectrum of an audio frame [55]. Another possible choice consists in constant Q cepstral coefficients, obtained from a perceptually motivated time-frequency analysis known as constant Q transform. Finally, also the bicoherence belongs to this class of traditional features, even if it has not been widely used in literature to perform speech processing and recognition.

We will concentrate now on a couple of methods which make use of the above features: in particular, the first method is the one that really inspired our work, and makes use of bispectral analysis, while the other exploits the more common spectrogram as feature used to perform classifications of real and fake speech. We start from the former method, which has been developed by the authors in [4] and [1].

The first of these papers starts showing that a nonlinear operation such as the square, applied to a simple sinusoidal signal, produces a signal that has new harmonics which are correlated to the original ones. If we call ω_1 and ω_2 the original frequencies, and ϕ_1 and ϕ_2 the original phases of the signal, then the square operation produces the harmonics with frequencies $\omega_1 + \omega_2$, $\omega_1 - \omega_2$ and phases $\phi_1 + \phi_2$, $\phi_1 - \phi_2$.

These are higher order correlations in the signal due to nonlinear processing; they cannot be introduced on the signal just through linear operations, and it is not possible to detect them using a feature like the power spectrum. On the contrary, a third order feature like the bispectrum should be able to retain information about these kind of correlations.

In particular, if we define the discrete Fourier transform (DFT) of a signal $\boldsymbol{y}(k)$ as

$$\boldsymbol{Y}(\boldsymbol{\omega}) = \sum_{k=0}^{K-1} y(k) e^{-i\frac{2\pi}{K}k\boldsymbol{\omega}}, \tag{2.1}$$

then we can express the power spectrum as

$$\boldsymbol{P}(\omega) = \boldsymbol{Y}(\omega)\boldsymbol{Y}^*(\omega) \tag{2.2}$$

(where $\boldsymbol{Y}^*(\omega)$ denotes the conjugate of $\boldsymbol{Y}(\omega)$), and the bispectrum

as

$$\boldsymbol{B}(\omega_1, \omega_2) = \boldsymbol{Y}(\omega_1)\boldsymbol{Y}(\omega_2)\boldsymbol{Y}^*(\omega_1 + \omega_2). \tag{2.3}$$

Intuitively, we can see how the bispectrum response is able to capture "un-natural" higher order correlations between the harmonics of the signal produced by the application of a non linearity to the signal itself.

Since the bispectrum is a complex feature, we can express it in terms of its module and phase as follows:

$$|\boldsymbol{B}(\omega_1, \omega_2)| = |\boldsymbol{Y}(\omega_1)||\boldsymbol{Y}(\omega_2)||\boldsymbol{Y}^*(\omega_1 + \omega_2)|, \tag{2.4}$$

$$\angle\boldsymbol{B}(\omega_1, \omega_2) = \angle\boldsymbol{Y}(\omega_1) + \angle\boldsymbol{Y}(\omega_2) + \angle\boldsymbol{Y}^*(\omega_1 + \omega_2). \tag{2.5}$$

We now introduce the definition of bicoherence, which is the normalised bispectrum:

$$\boldsymbol{B_c}(\omega_1, \omega_2) = \frac{\boldsymbol{Y}(\omega_1)\boldsymbol{Y}(\omega_2)\boldsymbol{Y}^*(\omega_1 + \omega_2)}{\sqrt{|\boldsymbol{Y}(\omega_1)\boldsymbol{Y}(\omega_2)|^2|\boldsymbol{Y}(\omega_1 + \omega_2)|^2}}. \tag{2.6}$$

This is useful in practice, since the modules of the bicoherences assume values in the interval $[0, 1]$, which are the common values that features are supposed to assume in machine learning applications.

When working with audio signals, it is common to divide them into small segments and analyse and process each segment on its own. We are going to do that also in this case, extracting the bicoherence by each segment of the original audio signal and averaging all of them in order to obtain a more stable estimation. This process can be described by the following Equation:

$$\hat{\boldsymbol{B}}_c(\omega_1, \omega_2) = \frac{\frac{1}{W}\sum_w \boldsymbol{Y_w}(\omega_1)\boldsymbol{Y_w}(\omega_2)\boldsymbol{Y_w^*}(\omega_1 + \omega_2)}{\sqrt{\frac{1}{W}\sum_w |\boldsymbol{Y_w}(\omega_1)\boldsymbol{Y_w}(\omega_2)|^2 \frac{1}{W}\sum_w |\boldsymbol{Y_w}(\omega_1 + \omega_2)|}}. \tag{2.7}$$

where W is the number of segments, which can be assumed to be overlapping.

In [4] it is shown how the higher order correlations introduced by the application of non linear processing to the signal can be noticed also by human eye displaying the modules and phases of the bicoherences (basically treating them as images) extracted by the signal before and after the application of a simple square operation.

In Figure 2.2 are shown the modules and phases of few bicoherences extracted from both bonafide and spoofed speech.

In the second paper ([1]) are shown the results of classifications based on bispectral analysis of speech signals.

The dataset used consists of human speech plus five different classes of state of the art synthesised speech. The features used in the classification, which have been extracted from the bicoherences, consist in the first

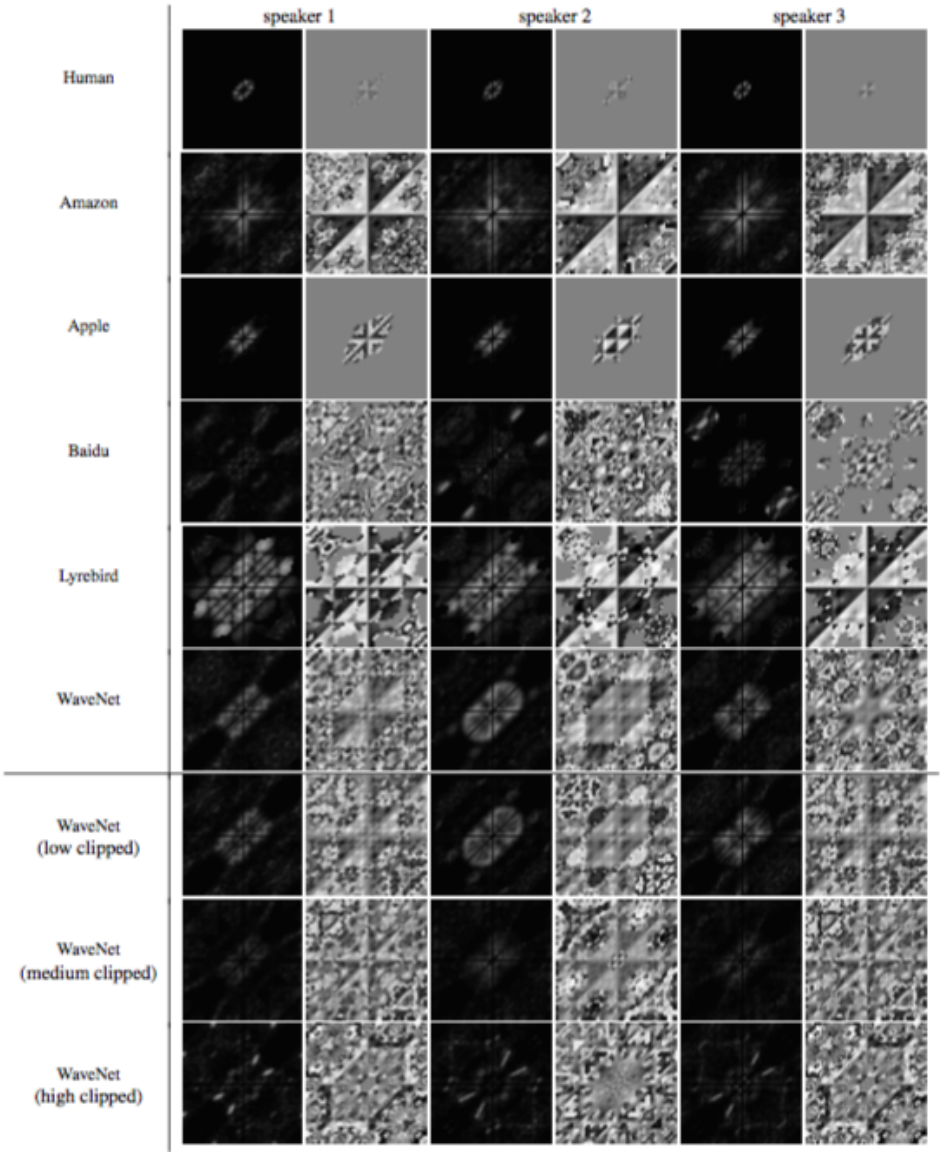


Figure 2.2: Modules and phases of the bicoherences extracted from both bonafide speech and speech synthesised with state of the art methods. The picture is taken from [1]

four statistical moments, i.e. mean, variance, skewness and kurtosis, computed from both the modules and the phases of the bicoherences.

These quantities are respectively obtained as

$$\mu_X = E_X[\boldsymbol{X}], \tag{2.8}$$

$$\sigma_X = E_X[(\boldsymbol{X} - \mu_X)^2], \tag{2.9}$$

$$\gamma_X = E_X\left[\left(\frac{\boldsymbol{X} - \mu_X}{\sigma_X}\right)^3\right], \tag{2.10}$$

$$\kappa_X = E_X \left[\left(\frac{\boldsymbol{X} - \mu_X}{\sigma_X} \right)^4 \right], \tag{2.11}$$

where $E_X[]$ is the expected value operator with respect to the random variable $\boldsymbol{X}$, and $\boldsymbol{X}$ represents the underlying distribution in turn for the magnitude and phase of the bicoherence. The estimation of these four moments is performed replacing the expected value operator with a simple average [1].

In order to classify these features, the authors in [1] use a series of one-versus-the-rest logistic regressions (See Chapter 1), each of which is able to recognise a single class of speech (either bonafide or one of the five classes of spoofed speech) from all the other classes. This way, a multiclass classification is implemented.

The accuracy achieved by this classification is $85\% - 86\%$, with the bonafide instances being among the ones that are perfectly classified, while instances of some of the classes of spoofed speech being misclassified as to belong to other classes of spoofed speech.

In general, this result is very convenient and useful, and is an indicator of a really appropriate methodology by means of which real and fake voices can be told apart.

In the next Chapters, we will further explore the results showed in this Section; indeed, we will replicate them, applying the algorithm to a wider and more heterogeneous dataset of speech recordings, and we will present new methodologies, which have been inspired by the above one, that will yield more general and strong results.

The second method that we describe here is the one based on the spectrogram of the signal, whose applications gives interesting results in the task of recognising speech generated by real speakers from spoofed speech. This method is detailed in [56].

The authors here extract from the audio files, which consist of both human and three different classes of state of the art synthesised speech, two different features: the spectrogram and the Mel-frequency spectrogram.

The spectrogram is computed by means of a short-time Fourier transform, after the signal has been segmented and windowed using overlapping Hanning windows. The Mel-frequency spectrogram is then computed from the spectrogram applying a simple transformation of the frequency axes such that the samples are not uniformly spaced in the frequency domain any more; this transformation relies on a psychoacustic model of the human hear, which is used to take into account the fact that sounds with different frequency content are perceived at different volumes.

Both the above quantities are represented as images, and are used as inputs to the classifier, which consists of a convolutional neural network (CNN).

In this case, the network has been trained to perform binary classifications, which means that it distinguishes between bonafide and fake speech, without classifying the different classes of fake speech accordingly to the synthesis algorithm.

Also in this case the results are very optimistic, consisting in generally high accuracies, specially for the classifications performed using the Mel-frequency spectrogram.

2.3.2 Methods based on data driven features

A second class of features which can be used in the classification of bonafide and fake speech consists in the class of data driven features. These features are generally extracted using machine learning techniques, which require a large set of input speech signals in order to perform the training of the chosen model.

Usually these features, by themselves, are not sufficient to achieve better performances than the traditional ones; instead, some experiments found in literature have shown that the combination of data driven and traditional features is what can really make a difference in the performance of the classifiers.

There are many possible choices of machine learning based feature extraction models. In this Section we will describe a method, which can be found in [55], which makes use of autoencoders (See Chapter 1) in order to extract data driven features starting from a set of classical features. These features are then used in order to distinguish bonafide speech from speech generated by a number of replay methods, without prior knowledge of the details of the spoofing algorithms used to obtain replayed speech.

The authors have extracted a number of different traditional features from speech audio. These features are the Mel-frequency cepstrum coefficients (MFCCs), the spectrogram, the constant Q cepstrum coefficients, the linear prediction cepstrum coefficients (LPCCs), the spectral subband centroid coefficients, the complex centroid coefficients and other features related to the MFCCs.

After this first feature extraction, each set of features is elaborated by means of an autoencoder, and the "encoder" part of the network is used to generate a new set of data driven features for each set of traditional features. The classifiers then consists in a Gaussian mixture model - universal background model (GMM-UBM), and the performance is evaluated by means of the equal error rate (EER).

Three different experiments have been performed, which differ from each other w.r.t the features used in the classifications: the first experiment, which only involves the traditional features, achieved good results, which could not be replicated by the second experiment, which used only the data driven features; as we have anticipated, the best performances were achieved by the third experiment, which involved both traditional

and data driven features.

3

Problem formulation and proposed methodologies

In this Chapter we present the problem we are focusing on, which generally consists in recognising real speakers from AI synthesised speech. We also introduce the proposed methodology in order to address the problem at hand. This methodology can be divided in two steps: a feature extraction and a classification step.

3.1 Problem formulation

In order to introduce the problem, we consider here a dataset consisting of a number of speech recordings in the form of digital audio signals. These signals have been generated by either recording real speakers (bonafide or real speech), or through a speech synthesis method (spoofed or fake speech). We suppose that spoofed speech has been synthesised using a finite number of different speech synthesis algorithms, which consist of text to speech (TTS), voice conversion (VC) and mixed methods.

The major objective that drives this work is to find an automatic procedure which is able to recognise bonafide speech from spoofed speech, having as input speech signals that belong to one of the two classes. This can be seen as a two-class classification problem.

To solve this problem, we are going to make use of bispectral analysis, which has attracted our attention for its nice properties of being able to retain information about higher order correlations introduced in the speech signals by nonlinear processing, as we saw in Chapter 2.

Bispectral analysis, anyway, is such a powerful tool, that it can be used to extract more information from the speech signals, other than just

which class, i.e. bonafide or fake, it belongs to.

With that in mind, we can expand the range of our aim, identifying three main objectives for the purpose of this thesis:

- Discriminating between real and fake speech recordings.
- Classifying the spoofed speech with respect to the synthesis method used to generate it.
- Detecting spoofed speech generated with speech synthesis algorithms which are not known in advance.

The first objective, as we said above, can be achieved by means of simple two-class classifications, while the second one requires multiclass classifications, where each class corresponds to a single speech synthesis algorithm. The third objective, instead, will be achieved simulating an open set environment (see Figure 3.1).

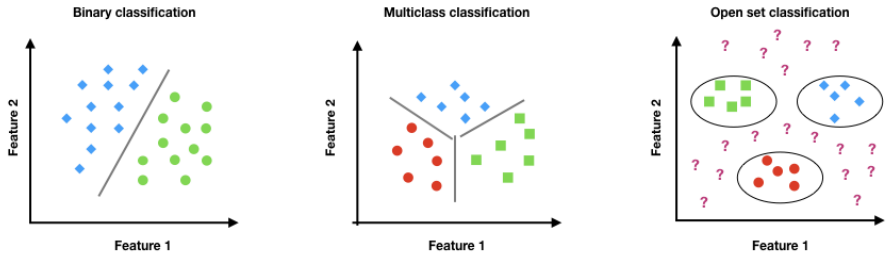


Figure 3.1: Scheme that shows the three different types of classification corresponding to the three objectives of this thesis.

In the following, we will give the details of the methodology we used to solve these problems. We will first focus on the feature extraction techniques, and then we will describe the classification algorithms.

3.2 Proposed Methodologies

As we said above, the methodology that we present here mainly consists in a feature extraction and a classification steps.

The feature extraction we are going to propose consists in two phases. The first one is the computation of the bicoherences from both bonafide and spoofed speech recordings, while the second phase is the extraction of other three different sets of features from the bicoherences themselves (namely MK, UNET and MKU features).

The classifications we are going to perform can be generally divided into the classes of closed set and open set classifications (see Figure 3.2).

The closed set ones are both two-class and multiclass classification tasks, performed using well known machine learning algorithms such as

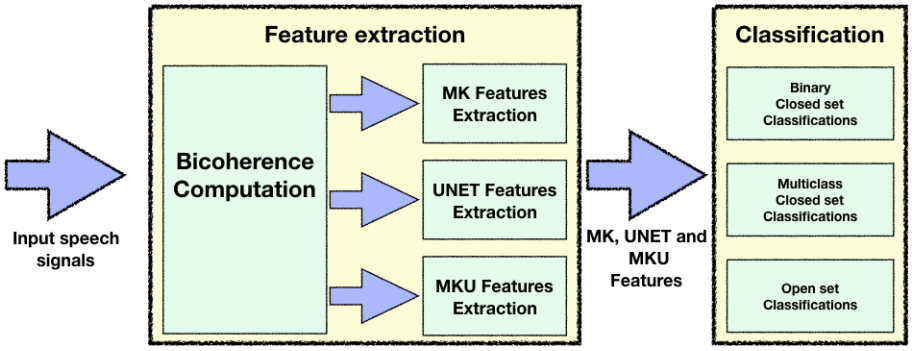


Figure 3.2: Block diagram of the proposed methodology.

support vector machine (SVM), logistic regression (LR) and random forest (RF). As concerns the open set classifications, we will describe our algorithm, which can be seen as the construction of a multiclass discriminant function which is able to both discriminate among each class of known speech and to detect instances belonging to unknown classes (i.e. not used to train the model). This algorithm makes use of a series of support vector machines.

3.2.1 Feature extraction

The feature extraction algorithm consists in two phases (as shown in Figure 3.2). The first one is the computation of the bicoherence from all input speech signals, while the second phase consists in extracting from the bicoherences other three sets of features (as shown in Figure 3.2), which we are going to call mean-kurtosis (MK), UNET and mean-kurtosis-UNET (MKU) features. These features are the one we are going to use to perform the classifications.

3.2.1.1 Bicoherence computation

We already detailed the main properties of the bicoherence and showed how to extract it from an audio file in Section 2.3.1. Here we will propose again some of the main aspects of the argumentation.

The bicoherence of a digital audio signal $\boldsymbol{y}(k)$ is computed as

$$\boldsymbol{B}_c(\omega_1, \omega_2) = \frac{\boldsymbol{Y}(\omega_1)\boldsymbol{Y}(\omega_2)\boldsymbol{Y}^*(\omega_1 + \omega_2)}{\sqrt{|\boldsymbol{Y}(\omega_1)\boldsymbol{Y}(\omega_2)|^2 |\boldsymbol{Y}(\omega_1 + \omega_2)|^2}}. \tag{3.1}$$

where the * operator denotes the complex conjugate and $\boldsymbol{Y}(\omega)$ is the discrete Fourier transform (DFT) of $\boldsymbol{y}(k)$, defined as

$$\boldsymbol{Y}(\omega) = \sum_{k=0}^{K-1} y(k) e^{-i\frac{2\pi}{K}k\omega}, \tag{3.2}$$

which gives a representation of the spectral information of the input signal. We have assumed here that $\boldsymbol{y}(k) = (y_0, y_1, ..., y_{K-1})$ is the sampled version of a continuous-time signal $\boldsymbol{y}(t)$, and $\omega = (\omega_0, \omega_1, ...\omega_{K-1})$ is a discrete frequency variable.

There are efficient algorithms for the computation of the DFT, known as fast Fourier transforms, which are the ones we are going to use here.

Moreover, it is common to work with small frames of the input signal, instead of the original signal as a whole. Therefore, usually the signal $\boldsymbol{y}(k)$ is segmented and windowed, obtaining W different audio frames $\boldsymbol{y}_w(k')$, with $k' = 0, ..., N-1$, where each frame consists of N samples. After that, a single bicoherence is extracted from each segment of the signal, and the final estimate for the whole signal is obtained averaging all these quantities as follows:

$$\hat{\boldsymbol{B}}_c(\omega_1, \omega_2) = \frac{\frac{1}{W}\sum_w \boldsymbol{Y}_w(\omega_1)\boldsymbol{Y}_w(\omega_2)\boldsymbol{Y}_w^*(\omega_1 + \omega_2)}{\sqrt{\frac{1}{W}\sum_w |\boldsymbol{Y}_w(\omega_1)\boldsymbol{Y}_w(\omega_2)|^2 \frac{1}{W}\sum_w |\boldsymbol{Y}_w(\omega_1 + \omega_2)|}}. \tag{3.3}$$

The bicoherence is a complex quantity, consisting in a square matrix of complex numbers, which can be described in terms of its module and phase as follows (for the sake of conciseness here we changed the notation, defining $p = \omega_1$ and $q = \omega_2$, with $p = 0, ..., P-1$ and $q = 0, ...Q-1$):

$$\boldsymbol{M}(p, q) = |\boldsymbol{B}_c(\omega_1, \omega_2)|, \tag{3.4}$$

$$\boldsymbol{H}(p, q) = \angle \boldsymbol{B}_c(\omega_1, \omega_2). \tag{3.5}$$

These quantities consist in turn of square matrices of real numbers, and can be treated as images. The dimension of these matrices is equal to half the number of samples of the window i.e. $P = Q = \frac{N}{2}$. This is due to the fact that after the computation of the discrete Fourier transform, only half of the symmetric spectrum (i.e. the one associated to the positive frequencies) is retained. We remark here that since the bicoherence is a normalised feature, its module assumes values in the range $[0, 1]$, while the phase assumes values in the range $[-\pi, \pi]$.

We are going to investigate on the discriminating power of the bicoherence as concerns the ability of retaining information about higher

order correlations introduced in the speech signals by nonlinear processing, which is surely involved in the synthesis phase in case of spoofed recordings, while should be absent in case of bonafide recordings of real speakers.

3.2.1.2 MK Features

Extraction of the mean-kurtosis (MK) features from the bicoherence, consists in computing the first four statistical moments (i.e. mean, variance, skewness and kurtosis) from both the module and the phase of the bicoherence as follows:

$$\mu_X = E_X[\boldsymbol{X}], \tag{3.6}$$

$$\sigma_X = E_X[(\boldsymbol{X} - \mu_X)^2], \tag{3.7}$$

$$\gamma_X = E_X\left[\left(\frac{\boldsymbol{X} - \mu_X}{\sigma_X}\right)^3\right], \tag{3.8}$$

$$\kappa_X = E_X\left[\left(\frac{\boldsymbol{X} - \mu_X}{\sigma_X}\right)^4\right], \tag{3.9}$$

where $E_X[]$ is the expected value operator with respect to the random variable $\boldsymbol{X}$, and $\boldsymbol{X}$ represents the underlying distribution in turn for the magnitude $\boldsymbol{M}(p,q)$ and phase $\boldsymbol{H}(p,q)$ of the bicoherence. The estimation of these four moments is performed replacing the expected value operator with a simple average as follows:

$$\mu_M = \sum_{p=0}^{P-1}\sum_{q=0}^{Q-1}\frac{1}{PQ}\boldsymbol{M}(p,q), \tag{3.10}$$

$$\sigma_M = \sum_{p=0}^{P-1}\sum_{q=0}^{Q-1}\frac{1}{PQ}(\boldsymbol{M}(p,q) - \mu_M)^2, \tag{3.11}$$

$$\gamma_M = \sum_{p=0}^{P-1}\sum_{q=0}^{Q-1}\frac{1}{PQ}\left(\frac{\boldsymbol{M}(p,q) - \mu_M}{\sigma_M}\right)^3, \tag{3.12}$$

$$\kappa_M = \sum_{p=0}^{P-1}\sum_{q=0}^{Q-1}\frac{1}{PQ}\left(\frac{\boldsymbol{M}(p,q) - \mu_M}{\sigma_M}\right)^4, \tag{3.13}$$

$$\mu_H = \sum_{p=0}^{P-1}\sum_{q=0}^{Q-1}\frac{1}{PQ}\boldsymbol{H}(p,q), \tag{3.14}$$

$$\sigma_H = \sum_{p=0}^{P-1}\sum_{q=0}^{Q-1}\frac{1}{PQ}(\boldsymbol{H}(p,q) - \mu_H)^2, \tag{3.15}$$

$$\gamma_H = \sum_{p=0}^{P-1} \sum_{q=0}^{Q-1} \frac{1}{PQ} \left(\frac{\boldsymbol{H}(p,q) - \mu_H}{\sigma_H} \right)^3, \tag{3.16}$$

$$\kappa_H = \sum_{p=0}^{P-1} \sum_{q=0}^{Q-1} \frac{1}{PQ} \left(\frac{\boldsymbol{H}(p,q) - \mu_H}{\sigma_H} \right)^4, \tag{3.17}$$

where $\boldsymbol{M}(p,q)$ and $\boldsymbol{H}(p,q)$, with $p = 0, ..., P-1$ and $q = 0, ...Q-1$ indicate respectively the generic module and phase of a bicoherence, and we have that $P = Q = \frac{N}{2}$.

The MK features, then consist in the vector

$$\boldsymbol{\alpha} = (\mu_M, \sigma_M, \gamma_M, \kappa_M, \mu_H, \sigma_H, \gamma_H, \kappa_H). \tag{3.18}$$

These features have already been used in [1] in order to perform a multiclass classification of some speech signals (both bonafide and fake) with respect to the synthesis algorithm used to generate them, and have proved to be meaningful for the purpose of discriminating between recordings of real speakers and spoofed recordings. We are going to use them in order to replicate the results achieved by the authors, applying their algorithm to a new and more various dataset of voices.

3.2.1.3 UNET and MKU Features

We show now a method that can be used to extract from the bicoherences a second set of features, which we are going to call UNET features, since a convolutional neural network (CNN) called U-Net has been used to perform this feature extraction.

As we said in Chapter 1, a U-Net is a complex CNN that works like a convolutional autoencoder, for the fact that it consists in two modules: an encoder and a decoder. While it has been introduced in [18] to perform biomedical image segmentation, it can be used in many applications, such as anomaly detection and feature extraction. In Figure 3.3 is schematised the architecture of the U-Net we used to extract the UNET features.

The layers that compose the encoder of the U-Net are convolutional and subsampling layers, while the decoder is composed by convolutional and upsampling layers. Moreover, in the decoder module, we can find few skip-connection layers (also known as concatenation layers) (see Table 3.1), which are used to concatenate the output of some shallow layers with the one of deeper layers.

In general, we can say that the U-Net is used to work with image data; the images, in our case, are the modules of the bicoherences.

Recall that the encoder module of the U-Net takes as input an image and gives as output a compressed version of the image itself, while the decoder takes as input the compressed image (i.e. the output of the encoder) and gives as output a reconstructed image, similar to the original one.

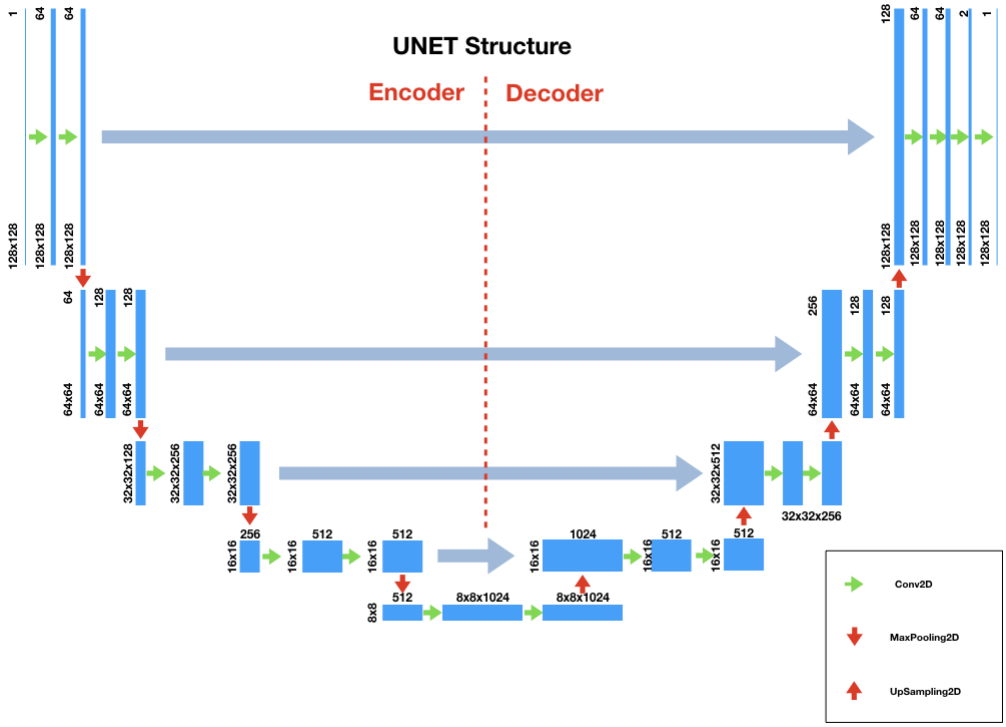


Figure 3.3: Scheme that shows the architecture of the U-Net used for feature extraction.

The fundamental concept of the U-Net, is that it is able to reconstruct better the class of images used to train it. Thus, if we train it with the modules of the bicoherences extracted from a specific class of speech (for example bonafide speech), then it will be able to reconstruct the modules of the bicoherences extracted from bonafide speech signals better than the modules of the bicoherences extracted by spoofed speech recordings. This could lead to the development of methods which are able to detect differences between the two classes of speech.

In Appendix B we will show a method, that makes use of the above property of the U-Net (and of autoencoders in general) to perform anomaly detection using the modules of the bicoherences extracted from bonafide speech to train a U-Net.

Our main interest here is to extract meaningful features from the modules of the bicoherences computed from all the classes of speech using a series of U-Nets. In order to do that, we trained five U-Nets, using five different classes of spoofed speech. Each class of spoofed speech has been generated using a specific and unique speech synthesis algorithm. These five speech synthesis algorithms have been chosen among the available ones in order to obtain a set of speech classes where each class is as much different as possible from the other four ones. For more details, see Chapter 4, where we give the details of the dataset we used.

After the training has been performed, the features have been ex-

Layer	Output shape	# Parameters	Connected to
input1 (InputLayer)	(None, 128, 128, 1)	0	
conv2d1 (Conv2D)	(None, 128, 128, 64)	640	input1
conv2d2 (Conv2D)	(None, 128, 128, 64)	36928	conv2d1
maxpooling2d1 (MaxPooling2D)	(None, 64, 64, 64)	0	conv2d2
conv2d3 (Conv2D)	(None, 64, 64, 128)	73856	maxpooling2d1
conv2d4 (Conv2D)	(None, 64, 64, 128)	147584	conv2d3
maxpooling2d2 (MaxPooling2D)	(None, 32, 32, 128)	0	conv2d4
conv2d5 (Conv2D)	(None, 32, 32, 256)	295168	maxpooling2d2
conv2d6 (Conv2D)	(None, 32, 32, 256)	590080	conv2d5
maxpooling2d3 (MaxPooling2D)	(None, 16, 16, 256)	0	conv2d6
conv2d7 (Conv2D)	(None, 16, 16, 512)	1180160	maxpooling2d3
conv2d8 (Conv2D)	(None, 16, 16, 512)	2359808	conv2d7
dropout1 (Dropout)	(None, 16, 16, 512)	0	conv2d8
maxpooling2d4 (MaxPooling2D)	(None, 8, 8, 512)	0	dropout1
conv2d9 (Conv2D)	(None, 8, 8, 1024)	4719616	maxpooling2d4
conv2d10 (Conv2D)	(None, 8, 8, 1024)	9438208	conv2d9
dropout2 (Dropout)	(None, 8, 8, 1024)	0	conv2d10
upsampling2d1 (UpSampling2D)	(None, 16, 16, 1024)	0	dropout2
conv2d11 (Conv2D)	(None, 16, 16, 512)	2097664	upsampling2d1
concatenate1 (Concatenate)	(None, 16, 16, 1024)	0	dropout1, conv2d11
conv2d12 (Conv2D)	(None, 16, 16, 512)	4719104	concatenate1
conv2d13 (Conv2D)	(None, 16, 16, 512)	2359808	conv2d12
upsampling2d2 (UpSampling2D)	(None, 32, 32, 512)	0	conv2d13
conv2d14 (Conv2D)	(None, 32, 32, 256)	524544	upsampling2d2
concatenate2 (Concatenate)	(None, 32, 32, 512)	0	conv2d6, conv2d14
conv2d15 (Conv2D)	(None, 32, 32, 256)	1179904	concatenate2
conv2d16 (Conv2D)	(None, 32, 32, 256)	590080	conv2d15
upsampling2d3 (UpSampling2D)	(None, 64, 64, 256)	0	conv2d16
conv2d17 (Conv2D)	(None, 64, 64, 128)	131200	upsampling2d3
concatenate3 (Concatenate)	(None, 64, 64, 256)	0	conv2d4, conv2d17
conv2d18 (Conv2D)	(None, 64, 64, 128)	295040	concatenate3
conv2d19 (Conv2D)	(None, 64, 64, 128)	147584	conv2d18
upsampling2d4 (UpSampling2D)	(None, 128, 128, 128)	0	conv2d19
conv2d20 (Conv2D)	(None, 128, 128, 64)	32832	upsampling2d4
concatenate4 (Concatenate)	(None, 128, 128, 128)	0	conv2d2, conv2d20
conv2d21 (Conv2D)	(None, 128, 128, 64)	73792	concatenate4
conv2d22 (Conv2D)	(None, 128, 128, 64)	36928	conv2d21
conv2d23 (Conv2D)	(None, 128, 128, 2)	1154	conv2d22
conv2d24 (Conv2D)	(None, 128, 128, 1)	3	conv2d23

Table 3.1: Summary of all the layers of the U-Net.

tracted in the following way (see also Figure 3.4), where we denote with $\boldsymbol{M}$ the module $\boldsymbol{M}(p, q)$ of a generic bicoherence:

- We tested the five U-Nets with the image $\boldsymbol{M}$, obtaining five different reconstructed versions of the original image, which we are going to denote as $\boldsymbol{M}_b^1, \boldsymbol{M}_b^2, \boldsymbol{M}_b^3, \boldsymbol{M}_b^4, \boldsymbol{M}_b^5$, where the superscript $^1, ...,^5$ indicates the U-Net used to perform the operation
- We computed the compressed version (using only the encoder part of the networks) of the original image $\boldsymbol{M}$, obtaining five different compressed versions of M, denoted as $\boldsymbol{M}_c^1, \boldsymbol{M}_c^2, \boldsymbol{M}_c^3, \boldsymbol{M}_c^4, \boldsymbol{M}_c^5$
- We computed the compressed versions of the reconstructed images (i.e. the output of the five U-Nets given M as input), where each reconstructed version is elaborated only through the encoder part

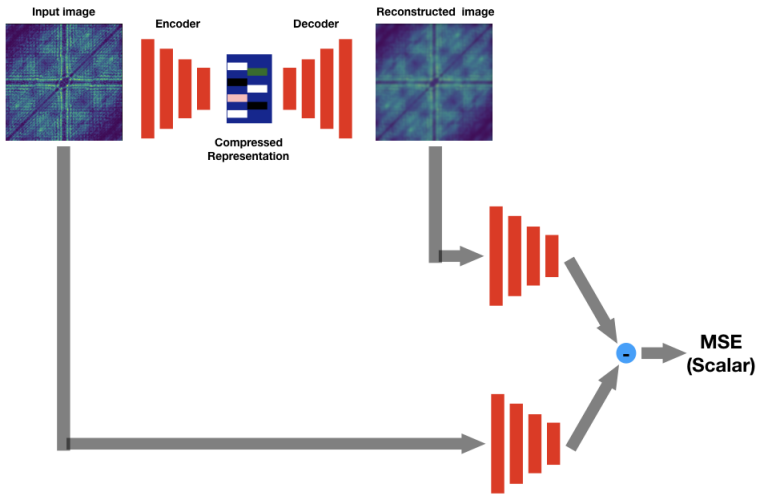


Figure 3.4: Feature extraction performed using an autoencoder.

of the network used to generate it (which means for example that the image $\boldsymbol{M}_b^1$ is tested using only the encoder of the U-Net denoted by the superscript ^{1}); this way we obtained other five compressed versions of the image denoted by $\boldsymbol{M}_{bc}^1, \boldsymbol{M}_{bc}^2, \boldsymbol{M}_{bc}^3, \boldsymbol{M}_{bc}^4, \boldsymbol{M}_{bc}^5$.

- Finally, we computed the mean square error (MSE) between the two compressed versions of the original and the reconstructed image, which in this case have three dimensions; for example, for the two compressed versions associated with the U-Net ^{1}, we computed

$$MSE^1 = \frac{1}{ABC} \sum_a \sum_b \sum_c (\boldsymbol{M}_c^1(a,b,c) - \boldsymbol{M}_{bc}^1(a,b,c))^2, \quad (3.19)$$

where the indexes a, b and c scan each item in the data structure that represents each one of the two compressed versions of the image, which, by construction, have the same dimensions.

This procedure has to be applied to the modules of the bicoherences extracted by every available speech signal (i.e. speech signals belonging to all the classes of speech, both bonafide and spoofed recordings).

This way, for each signal, we obtain basically five numbers, corresponding to the five MSEs. These numbers are our UNET features, defined by the vector

$$\boldsymbol{\beta} = (MSE^1, MSE^2, MSE^3, MSE^4, MSE^5). \quad (3.20)$$

We are going to show that these features are meaningful for the problem of classifying the speech recordings w.r.t the synthesis algorithm they

have been produced with, and are able to add more discriminating power to the MK features, which have already been proved to yield interesting performances.

To end this Section, we introduce the MKU features: these are obtained performing a simple concatenation of the MK and UNET features as

$$\boldsymbol{\theta} = (\boldsymbol{\alpha}, \boldsymbol{\beta}). \tag{3.21}$$

These features are the ones that perform better both in the closed set and in the open set environment.

In Chapter 4 we are going to show the results of closed set classifications of all three sets of features, while for the open set classifications we will only use the MK and MKU features.

3.2.2 Feature normalisation and dimensionality reduction

In this Section we will describe how the MK, UNET and MKU features have been normalised, before being used for the classification tasks. Moreover, we will explain how the scatter plots in Chapter 4 have been produced following the application of some dimensionality reduction techniques.

3.2.2.1 Feature normalisation

Once extracted the features, they have been normalised, in order to make them assume values in the range $[0, 1]$, which are the desired values for instances that have to be used in machine learning applications.

The algorithm used to normalise them, which is known as min-max normalisation, has already been introduced in Chapter 1. We will report it here:

$$\boldsymbol{x}'_{\text{scaled}} = \frac{\boldsymbol{x} - \min(\boldsymbol{x})}{\max(\boldsymbol{x}) - \min(\boldsymbol{x})}. \tag{3.22}$$

where we denoted as $\boldsymbol{x}$ the signal that consists in the sequence of the features (which will in turn be one of the vectors $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\theta}$ defined in the previous sections), $\boldsymbol{x}'_{\text{scaled}}$ is the signal consisting in the sequence of normalised features, and $\min(\boldsymbol{x})$ and $\max(\boldsymbol{x})$ are respectively the minimum and maximum values of the original signal.

The above operation has been implemented making use of the "MinMaxScaler" of the Python library "scikit-learn" [57], using as range the interval $[0, 1]$.

3.2.2.2 Dimensionality reduction

In order to visualise the features, and see if they actually cluster into some groups with respect to the speech synthesis algorithm used to generate them, we used three different methods of dimensionality reduction.

These are useful to reduce the actual number of features associated to each speech signal to two or three features (while the original ones are eight for the MK features and thirteen for the MKU features) retaining all the meaningful properties of the features themselves (i.e. their discriminating power).

The reduced sets of features can be plotted in a two or three dimensional space, where the axis of these spaces correspond to the values of each item in the feature set. The resulting plots are commonly known as scatter plots.

The dimensionality reduction algorithms that we applied are principal component analysis (PCA), independent component analysis (ICA) and t-distributed stochastic neighbour embedded (TSNE), and have been detailed in Chapter 1. We will show the scatter plots of the MK and MKU features obtained after applying these algorithms in Chapter 4.

3.2.3 Closed set classification

After all the features have been extracted and normalised, we now think about how to classify them. As we said before, in order to describe the algorithms, we are making a major distinction between the two classes of closed set and open set classifications. Our main interest in this Section are closed set classifications, while in the next Section we will take care of describing open set ones.

As concerns the closed set classifications, we make another distinction between two-class (or binary) and multiclass classifications.

In particular, in a binary closed set classification we have that the training and test sets will contain data that belong to one of two classes, which for clarity we will label here here as *Bonafide* and *Fake*. The *Bonafide* class will contain recordings of real speakers, while the *Fake* class may be generally composed by speech signals synthesised using one or more different TTS or VC algorithms.

On the contrary, in a multiclass closed set classification, we have more than two classes to discriminate among. One of them will always be the *Bonafide* class, while the others will be generally indicated as *Alg1*, *Alg2*, ...*AlgN*. Each one of these latter classes contains speech synthesised using a single speech synthesis algorithm. This way, we not only try and distinguish between bonafide and spoofed speech recordings, but we also try classify each fake signal with respect to the synthesis algorithm it has been generated with.

We remark here that in any case, the training and test sets are always going to be disjoint, which is a key assumption in every machine learning application.

In particular, for each set of features (MK, UNET and MKU), we performed the following classifications:

- A multiclass classification, which involve the classes *Bonafide*, *Alg1*, *Alg2*, ...*AlgN*. We will indicate this classification with the acronym closed set multiclass (CSM).
- A binary classification, which involve the classes *Bonafide* and *Fake*, where we tried to distinguish bonafide speech from the set of all spoofed speech classes, taken together. We will refer to this classification as closed set bonafide versus all spoof (CSBVAS).
- A series of binary classifications, where we tried to distinguish bonafide speech from each one of the classes of spoofed speech, taken one at the time, such that the classes involved in each classification will be in turn: *Bonafide* and *Alg1*, *Bonafide* and *Alg2*, ..., *Bonafide* and *AlgN*. This classification will be referred as closed set bonafide versus single spoof (CSBVSS).

The machine learning algorithms that we selected among the many possible ones to perform closed set classifications are the following: support vector machine (SVM), logistic regression (LR) and random forest (RF). In particular, the multiclass classification that involves the logistic regression and the support vector machine, has been implemented using respectively a series of one-versus-the-rest logistic regressions, and a series of one-versus-the-rest support vector machines. The binary classifications between bonafide and every single spoof class have been performed using only support vector machines. Chapter 1 contains a description of all these techniques.

All the classification algorithms have been implemented using the Python library scikit-learn [57].

In Chapter 4 we are going to show some of the confusion matrices associated to these classifications.

For completeness, in Appendix C are shown the confusion matrices associated to all the performed closed set classifications, together with a Table that reports all the classification accuracies, which can be used to compare the different closed set classification algorithms .

Moreover, in Appendix A are reported the results of binary and multiclass closed set classifications performed using a simple convolutional neural network (CNN). This network takes as input the modules of the bicoherences and gives as output, for each module, a label that indicates the class of speech they belong to.

3.2.4 Open set classification

Finally, we focus now on the open set classification algorithm, which has been applied using as input the MK and MKU features.

As we said before, the aim of an open set classification, is to both correctly classify the instances of the test set belonging to the classes that the model "knows", (i.e. the ones used in the training phase), and to recognise as "unknown" instances of the test set that belong to classes that have never been "seen" by the model, (i.e. not used in the training phase).

An open set environment can be used to give more general results, independently on the dataset used to train the model. Suppose for example that a new speech synthesis technique has been recently proposed, and that few samples of spoofed speech produced by that technique may be available. In this scenario, all previous closed set classifiers would not be able to label these new instances of speech as fake, since, even if there were enough samples to gather a training set, these classifiers would need to be re-adapted in order to correctly classify the new speech instances.

An open set classifier is absolutely appropriate to solve this problem, since it can be tested with speech signals belonging to any class, i.e. synthesised with any speech synthesis method.

We implemented three versions of the open set classification algorithm. The first one does not make use of known-unknown features. We shall label the classes involved the training phase of this classification as *Bonafide*, *Alg1*, ..., *AlgN*. The other two versions use some of the features as known-unknown in the training and test phase, such that the classes involved in the training can be labeled as *Bonafide*, *Alg1*,..., *AlgN*, *Known-Unknown*. During the test phase of all the three versions of the open set classification algorithm, a new class of speech is involved, which will be indicated as *Unknown* class; this class will contain speech synthesised by means of algorithms different from the ones used in the training phase.

Basically, known-unknown features are used to simulate the class labeled as *Unknown* during the training phase, and consist of features which in theory are "known", since they are used to train the model, but that are considered as "unknown". These features are disjoint from the set of real unknown features, which are only used in the test phase and have never been seen by the model before that phase.

For all the three versions of the open set classifier, we performed two experiments, which differ w.r.t the role of bonafide speech. In the first experiment bonafide speech has been placed among the known classes, while in the second experiment, instances of bonafide speech have been used as real unknown speech signals.

Now we give the details of the implementations of all the three versions of our open set classification algorithm.

The first version, which is the one that does not make use of the known-unknown features, has been implemented training five different binary one-versus-the-rest classifiers. Each classifier is used to distinguish a single class of speech by the other four classes. We tested the classifiers with both speech belonging to these five classes used in the training, and

speech that belongs to classes not used in the training (*Unknown*).

We labeled the test instances as known (and in that case we assigned them to a specific class among the ones used in the training) in case they were recognised by only one of the classifiers and "rejected" (i.e. classified as "the rest") by all the others. If more than one classifier assigned a specific instance to the class they represent, we chose the right class basing on the values of the scores, outputs of the five classifiers, which give an idea of the probability that an instance belongs to the training classes (i.e. assigning the instances to the class corresponding to the maximum score of the corresponding classifier). If all the classifiers classified the input as "the rest", we labeled it as unknown.

The results of this classification reflect the "pessimistic" nature of the model, by the fact that most of the unknown speech was actually associated by the model to one of the known classes, while the known instances were mostly correctly classified. To solve this problem, we implemented the other two versions of the open set classifier, which make use of the known-unknown features.

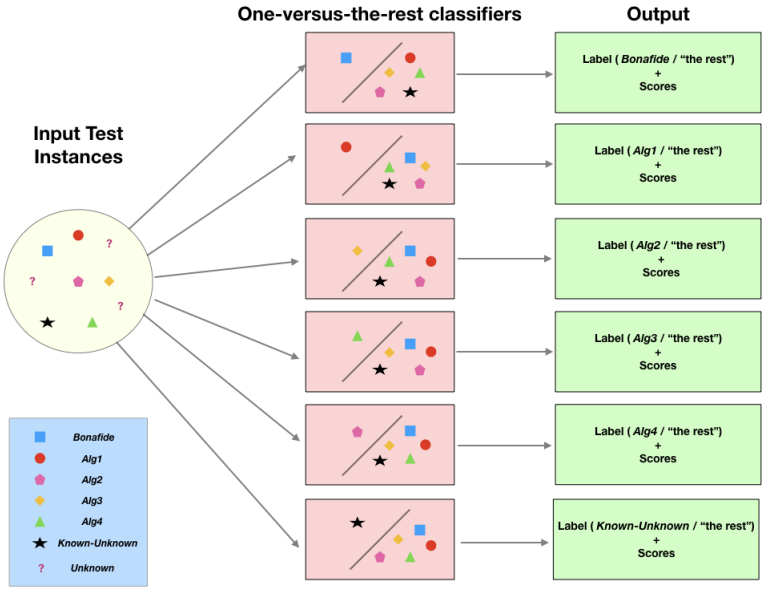


Figure 3.5: Scheme of the open set classification algorithm which involves known-unknown features.

The second version of our classification algorithm, makes use in the training phase of six different one-versus-the-rest classifiers (see Figure 3.5), where the first five are analogous to the ones of the first version, and the sixth one adds to the training the known-unknown features. This latter classifier is used to recognise *Known-Unknown* speech from all the other classes used in the training. We precise here that instances of speech belonging to the *Known-Unknown* class have been inserted among the speech labeled as "the rest" in the other five classifiers.

We tested this model with known, known-unknown and real unknown

features. As before, we labeled as known all the features that were recognised by one of the classifiers and rejected by all the others, and we associated them to the corresponding class, always making use of the values of the scores. If the classifier that recognised the features was the one trained with *Known-Unknown* features, we labeled these instances as *Known-Unknown*, while if all the classifiers rejected the features, we labeled them as *Unknown*.

This way we noticed that a relevant amount of real unknown features were labeled as known-unknown. Given this desirable property of our classification algorithm, we built the third version, which is pretty similar to the second one, except from the fact that we labeled as *Unknown* both instances that have been classified by the model as real unknown, and the ones that have been classified as known-unknown. This way we obtained a remarkable improvement in the classification accuracy.

The classifiers involved in the training of all the three versions of the open set classification algorithm have been implemented using, among the many possible machine learning classification techniques, support vector machines.

As we said before, all the experiments have been performed using the bonafide instances both as known features and as real unknown features.

We will refer to these open set classification algorithms with the following acronyms:

- Open set bonafide training version 1 (OSBT1) will indicate the open set classification performed by the first version of the classification algorithm using bonafide among the instances of the training set.
- Open set bonafide training version 2 (OSBT2) will indicate the open set classification performed by the second version of the classification algorithm using bonafide among the instances of the training set.
- Open set bonafide training version 3 (OSBT3) will indicate the open set classification performed by the third version of the classification algorithm using bonafide among the instances of the training set.
- Open set spoof training version 1 (OSST1) will indicate the open set classification performed by the first version of the classification algorithm using only fake classes in the training.
- Open set spoof training version 2 (OSST2) will indicate the open set classification performed by the second version of the classification algorithm using only fake classes in the training.
- Open set spoof training version 3 (OSST3) will indicate the open set classification performed by the third version of the classification algorithm using only fake classes in the training.

In Chapter 4 we are going to show the results of all the open set classifications.

In the same chapter, we will show few ROC curves (see Chapter 1) used to distinguish between known and unknown features. These curves have been computed having as input, for each instance, either the maximum score or the ratio between the maximum score and the score right below the maximum of the first five classifiers associated to the training of the third version of the open set classification algorithm (neglecting the scores of the classifier associated to the known-unknown features), together with labels indicating the nature of the instance (either known or unknown), computing and plotting the true positive rate (TPR) and the false positive rate (FPR) associated to these items.

The two ROC curves (i.e. the one associated to the maximum scores and the one associated to the ratios of the scores) will also be compared to the results obtained from the third version of the open set classification algorithm as concerns the ability of distinguishing between known and unknown instances.

4
Experimental results

In this Chapter we will introduce the ASVspoof 2019 dataset, which is a collection of both real and fake speech signals that we used in our experiments. We will give all the details about the parameters that compose our setup and we will also show some examples of the modules and phases of the bicoherences extracted by speech belonging to all the classes avilable in the dataset.

After that, we will show the results of the application of our methodology, which has been detailed in Chapter 3. In particular, for clarity and conciseness, in this Chapter we will only show the most relevant results. This means that, for example, when the same classification has been performed with more than one algorithm, we will show the result associated to the algorithm that achieved the best performance. However, the complete results can be found in Appendix C.

4.1 Dataset and setup

First of all, let us give an introduction of the dataset used in our experiments: the ASVspoof 2019 [48]. This dataset has been produced in order to encourage research on the development of countermeasures to presentation attacks against an automatic speaker verification (ASV) system. These presentation attacks, as we outlined in Chapter 2, include replay and speech synthesis.

The dataset is divided into two parts: logical access (LA) and physical access (PA). The physical access include speech signals generated through replay spoofing attacks, while the logical access contains voices synthesised using state of the art text to speech (TTS) and voice conversion (VC) algorithms.

Since we are only interested in the TTS and VC spoofing attacks, we discarded the PA scenario, and concentrate on the LA case.

The LA is divided into three sets: training set, development set and evaluation set. The training and development set consist in both bonafide speech recordings and spoofed recordings generated by six different algorithms; we are going to indicate these classes of spoofed speech signals as A01, A02, ..., A06. On the contrary, the evaluation set, is wider, and consist in both bonafide speech and speech synthesised using thirteen different TTS and VC methods. We indicate these latter spoofing algorithms as A07, A08, ..., A19. The classes A04 and A016 use the same synthesis algorithm, as well as the classes A06 and A19.

For simplicity, in our experiments we only used the evaluation set, using bonafide speech and instances belonging to its thirteen classes of spoofed speech. Now we give a brief description of all these spoofing algorithms, which have been initially detailed in [48].

Each of these algorithms can be characterised by the following basic blocks: an input processor that is used to convert text into a series of linguistic features (i.e. the phonemes), a duration model that predicts the duration of context-dependent phones, an acoustic model that estimates the acoustic features and finally a waveform generator, which converts the acoustic features into an audio waveform.

Here are the details of each algorithm:

- In A07 (TTS) speech, both the duration model and the acoustic model are based on recurrent neural networks (RNNs), and the acoustic features are the Mel-frequency cepstrum coefficients (MFCCs). The waveform is generated using the NN based "WORLD" vocoder [58].

- A08 (TTS) uses a hidden Markov model (HMM) based duration model, together with a RNN based acoustic model that estimates the MFCCs, and a NN based source filter model that generates the waveform.

- A09 (TTS) consist in RNN based duration and acoustic model, followed by a vocoder called "Vocaine" [59], which takes as input MFCCs.

- A10 is a TTS neural network based model, which uses as acoustic feature the Mel-frequency spectrogram.

- A11 is a TTS system which similar to A10, except from the fact that it uses the Griffin lim algorithm [60] to generate the acoustic waveform.

- A12 is a RNN based TTS system which uses the Wavenet vocoder [52] for the waveform generation.

- A13 is a mixed TTS-VC method, where a TTS system is first used to generate a voice from text, and that voice is the input to a VC system. The TTS part uses a unit selection approach, while the VC is based on neural networks.
- A14 is a TTS-VC method which makes use of MFCCs features and the STRAIGHT vocoder [61].
- A15 is another TTS-VC method similar to A14, except from the fact that it uses Wavenet [52] for the waveform generation
- A16 is a TTS system that uses a concatenation based unit selection approach with the MFCCs as acoustic features
- A17 is a NN based VC system that uses a generalised direct waveform modification method for waveform generation [48].
- A18 is a VC system which uses a MFCCs based vocoder for the generation of the waveform.
- A19 is a transfer function based VC system, which uses a source-filter model and the overlap and add method in order to generate the waveform of the target speaker.

This dataset is incredibly various and heterogeneous, since it comprehends instances of spoofed speech synthesised using a number of different state of the art methods.

As concerns the setup, in the following Tables can be found the details on all the parameters of our experiments, that can be used to replicate them.

In particular, Table 4.1 shows the details of the computation of the bicoherences, Table 4.2 shows the parameters used to extract the UNET features, in Table 4.3 can be found the parameters of the support vector machines used in all the experiments, Table 4.4 shows the setup of the closed set classifications (CSM, CSBVAS and CSBVSS), Table 4.5 contains the details of the open set classification without known-unknown features (OSBT1 and OSST1) and in Table 4.6 you can find the setup for the open set classification that uses known-unknown features (OSBT2, OSBT3, OSST2 and OSST3). The above acronyms have been defined in Chapter 3 and will be used also in the following Sections to indicate the different classification tasks.

One should notice that, given the symmetry of the spectrum of a signal, which is the output of the fast Fourier transform (FFT), we only selected the part corresponding to the positive frequencies. The result is that using a window of length N in the computation of the bicoherence, yields a complex matrix whose dimensions are $\frac{N}{2} \times \frac{N}{2}$. So for instance a window with length 256 gives as output a 128×128 matrix.

Now that we have given the details of all the synthesis methods, together with the values of the parameters in our setup, we are going

Bicoherence computation	
Window	Tukey window (shape=0.25)
Window length	256
Overlap	50%

Table 4.1: Details on the setup of the experiments: computation of the bicoherences.

UNET feature extraction	
Classes used for training	<A07, A10, A15, A16, A17>
# instances for training	3000
# instances for validation	1000

Table 4.2: Details on the setup of the experiments: UNET features extraction.

to show what the bicoherences look like. In Figure 4.7, can be found the plot of the modules and phases of the bicoherences extracted by an instance of speech signal belonging to each one of the classes involved in the classifications, including bonafide speech taken from the evaluation set of ASVspoof 2019.

Clearly, the bicoherences take various forms depending on the algorithm used to generate the underlying speech signal. Anyway, observing more examples, we can say that it is not visually possible to notice a clear difference, neither among the bicoherences extracted by speech signals generated with different synthesis algorithms, nor between fake and real bicoherences. Thus, the problem of classifying the speech signals using the bicoherences is not trivial, and must be dealt with carefully.

4.2 Closed set classification results

From this Section, we will show the results of our classification procedures, starting from the closed set classification of the MK, UNET and MKU features.

All the algorithms have been implemented using the Python library scikit-learn [57].

For conciseness, in this Chapter we will only show part of the results, which means the following: we will show only a couple of scatter plots for each set of features; when a classification has been implemented by more than one algorithm, we will show the confusion matrix associated to the algorithm that performs better; finally, we omit here the binary classifications between bonafide and each one of the spoof classes (CSBVSS), retaining only the two class classifications between bonafide and all the spoofed recordings taken as a single class (CSBVAS).

Parameters of the support vector machines	
C (regularization parameter)	1.0
Kernel type	'rbf' (radial basis function)
Gamma	'scale'
Shrinking heuristic	'True'
Probability estimates	'True'
Tolerance	1×10^{-3}
Kernel cache size	200
Maximum # iterations	-1 (no limit)

Table 4.3: Details on the setup of the experiments: parameters of the support vector machines involved in the closed set and open set classification (for more information see https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html).

Closed set classifications	
# instances for training (multiclass)	3000 for each class
# instances for test (multiclass)	1900 for each class
# instances for training (bonafide vs all fake)	4940 bonafide, 380 for each spoof class
# instances for test (bonafide vs all fake)	1950 bonafide, 150 for each spoof class
# instances for training (bonafide vs each fake class)	3000
# instances for test (bonafide vs each fake class)	1000

Table 4.4: Details on the setup of the experiments: closed set classifications.

For MK and MKU features, you can find here a 2D and a 3D scatter plot. Additionally, for all the sets of features, a confusion matrix associated to a multiclass classification and a confusion matrix associated to a two-class (or binary) classification will be shown.

All the remaining results are left for the reader in Appendix C.

4.2.1 Closed set classification of MK features

We start showing the results considering the mean-kurtosis (MK) features, which, as we already said, correspond to the four statistical moments extracted by both the module and the phase of the bicoherences, as shown in Chapter 3.

Open set classifications without known-unknown features	
known classes	<Bonafide, A09, A11, A14, A17> and <A09, A11, A13, A14, A17>
# instances for training	2000 for the target class, 500 for each of the other classes
unknown classes	<A13, A15, A18, A19> and <Bonafide, A15, A18, A19>
# instances for test	2000 for each known class, 500 for each unknown class

Table 4.5: Details on the setup of the experiments: open set classification without known-unknown features.

Open set classifications	
known classes	<Bonafide, A09, A11, A14, A17> and <A09, A11, A13, A14, A17>
known-unknown classes	<A07, A08, A10>
# instances for training	2999 for the target known class, 500 for each of the other known classes, 333 for each known-unknown class
unknown classes	<A13, A15, A18, A19> and <Bonafide, A15, A18, A19>
# instances for test	500 for each known class, 160 for each unknown-unknown class, 125 for each unknown class

Table 4.6: Details on the setup of the experiments: open set classification with known-unknown features.

In Figure 4.1 and Figure 4.2 can be found a 2D and a 3D scatter plot of some of these features , after they have been dimensionally reduced using the t-SNE algorithm (see Chapter 1).

Clearly, the various classes of speech do not cluster neatly neither in the 2D nor in the 3D space, with the only possible exception of class A09. This makes the classification task quite demanding.

Figure 4.3 shows the confusion matrix of a multiclass classification (CSM) performed using a series of one-versus-the-rest SVMs (see 1. The accuracy of this classification is 62%, and it is the highest among the

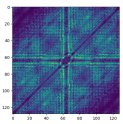
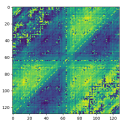
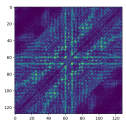
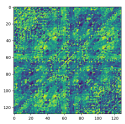
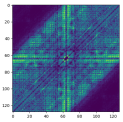
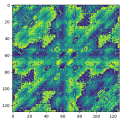
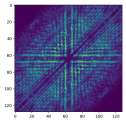
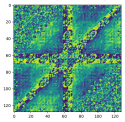
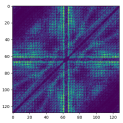
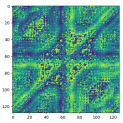
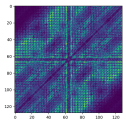
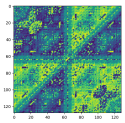
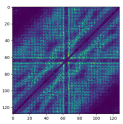
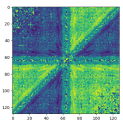
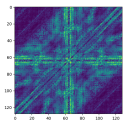
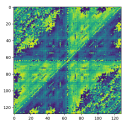
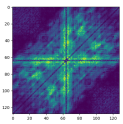
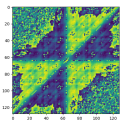
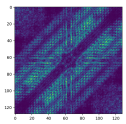
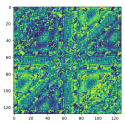
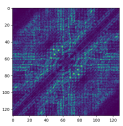
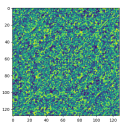
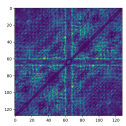
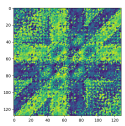
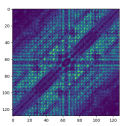
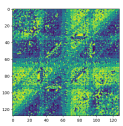
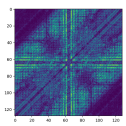
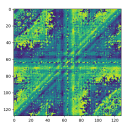
 Bonafide module	 Bonafide phase	 A13 module	 A13 phase
 A07 module	 A07 phase	 A14 module	 A14 phase
 A08 module	 A08 phase	 A15 module	 A15 phase
 A09 module	 A09 phase	 A16 module	 A16 phase
 A10 module	 A10 phase	 A17 module	 A17 phase
 A11 module	 A11 phase	 A18 module	 A18 phase
 A12 module	 A12 phase	 A19 module	 A19 phase

Table 4.7: Examples of bicoherence module and phase for each class involved in the classifications.

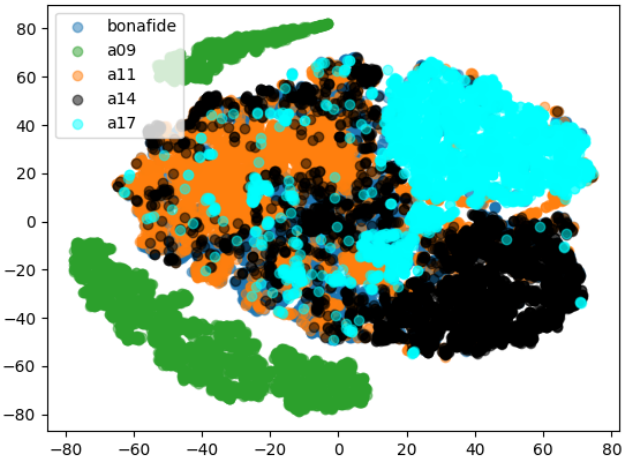


Figure 4.1: 2D catter plot of the MK features belonging to the classes Bonafide, A09, A11, A14 and A17, reduced using t-SNE.

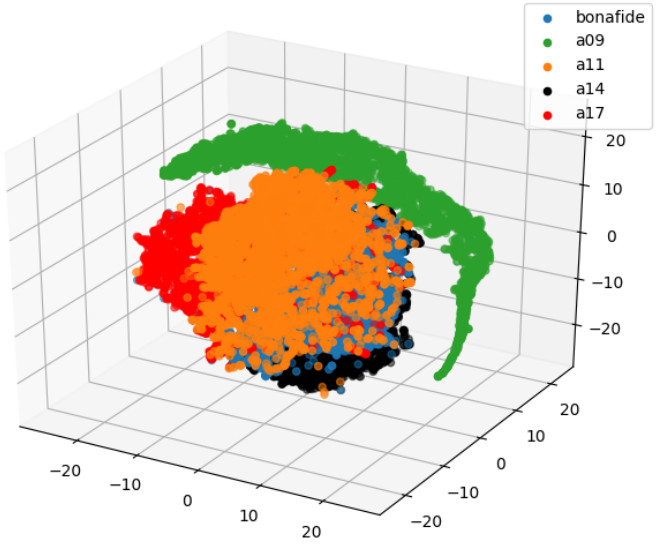


Figure 4.2: 3D scatter plot of the MK features belonging to the classes Bonafide, A09, A11, A14 and A17 reduced using t-SNE.

multiclass classifications of this kind performed using other algorithms such as LR and RF.

Finally, in Figure 4.4, you can find the confusion matrix associated to a binary classification (CSBVAS) where we tried to distinguish the Bonafide class from all the other classes taken together. The accuracy of this classification is 79%, which is naturally higher than the accuracy of the multiclass classification.

4.2.2 Closed set classification of UNET features

Now we concentrate on the UNET features, which have been extracted from the modules of the bicoherences as depicted in Chapter 3.

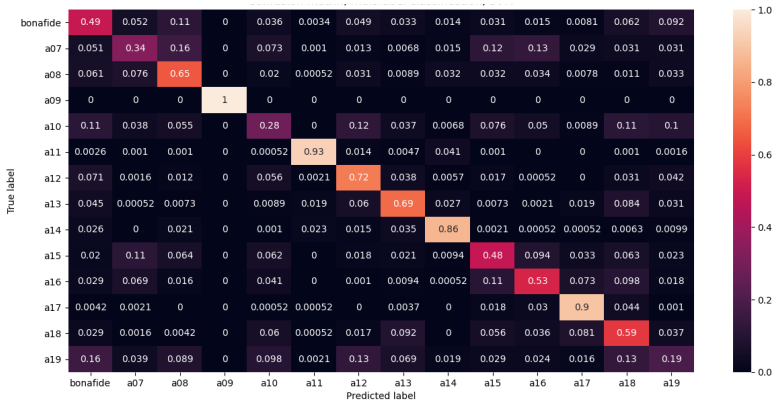


Figure 4.3: Confusion matrix of a multiclass classification (CSM) that involves MK features, performed using a series of one-versus-the-rest support vector machines.

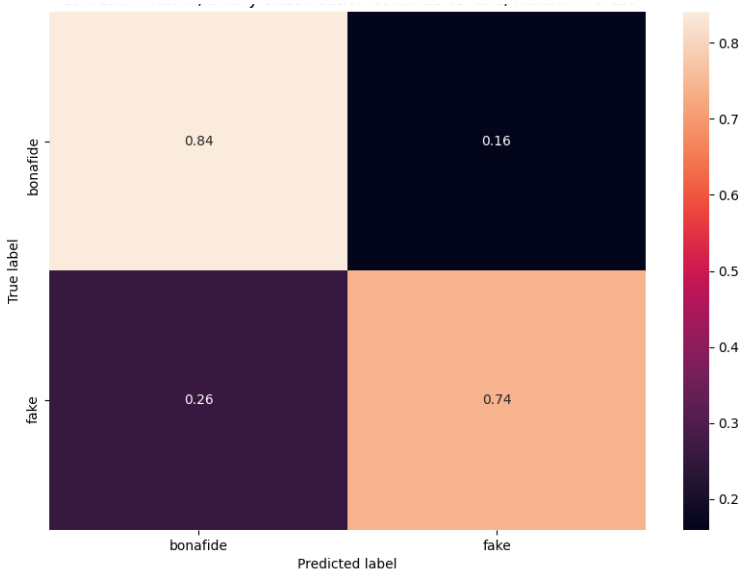


Figure 4.4: Confusion matrix of a binary (bonafide vs all spoof) (CSBVAS) classification that involves MK features, performed using a random forest.

In Figure 4.5 is shown the confusion matrix associated to the multiclass classification (CSM) of these features using a series of support vector machine. The accuracy of this classification is 35%, with the bonafide features being among the ones that have been classified better.

In Figure 4.6 is shown the confusion matrix of the binary (bonafide vs all fake) classification (CSBVAS) of the UNET features using a logistic regression (LR). Notice that the accuracy of this classification is 100%, and is equal to the accuracy of the same classification performed using a support vector machine (SVM) or a random forest (RF).

We notice from these results how the UNET features perform poorly when used to distinguishing the classes of spoofed speech from each other,

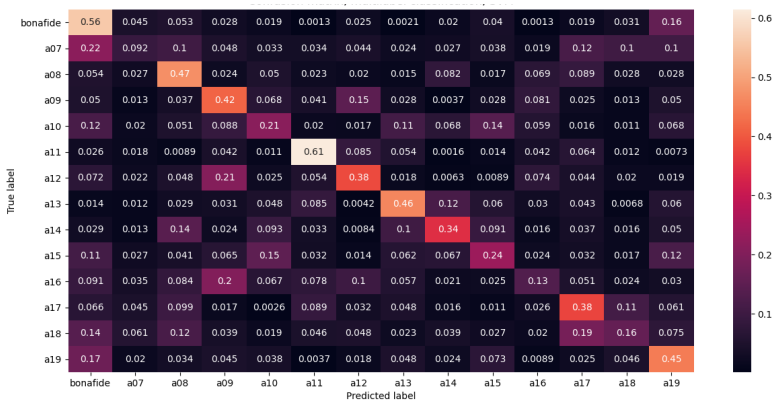


Figure 4.5: Confusion matrix of a multiclass classification (CSM) that involves UNET features, performed using a series of one-versus-the-rest support vector machines.

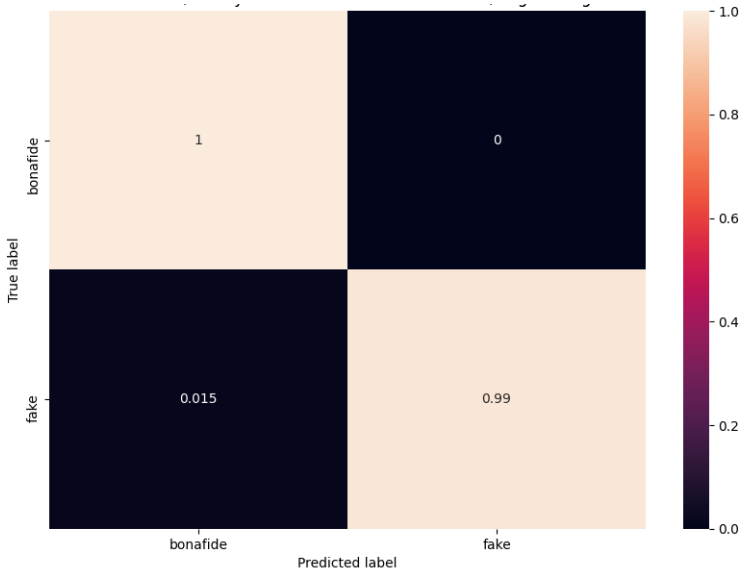


Figure 4.6: Confusion matrix of a binary (bonafide vs all spoof) (CSBVAS) classification that involves UNET features, performed using a logistic regression.

while they outperform MK features in the distinction between bonafide and fake speech.

4.2.3 Closed set classification of MKU features

Now we consider the MKU features, which are the result of a simple concatenation of the MK and the UNET features.

In Figures 4.7 and 4.8 can be found the 2D and 3D scatter plots obtained reducing the dimensionality of the MKU features using PCA (see Chapter 1), while in Figures 4.9 and 4.10 are shown the corresponding 2D and 3D plots obtained after the application of ICA to the MKU features

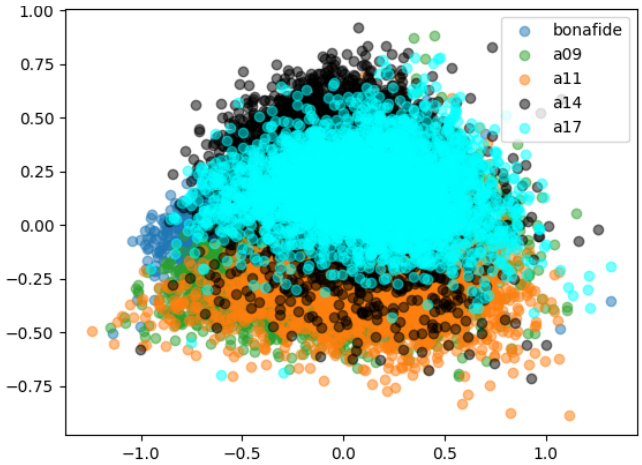


Figure 4.7: 2D scatter plot of the MKU features belonging to the classes bonafide, A09, A11, A14 and A17, reduced using PCA.

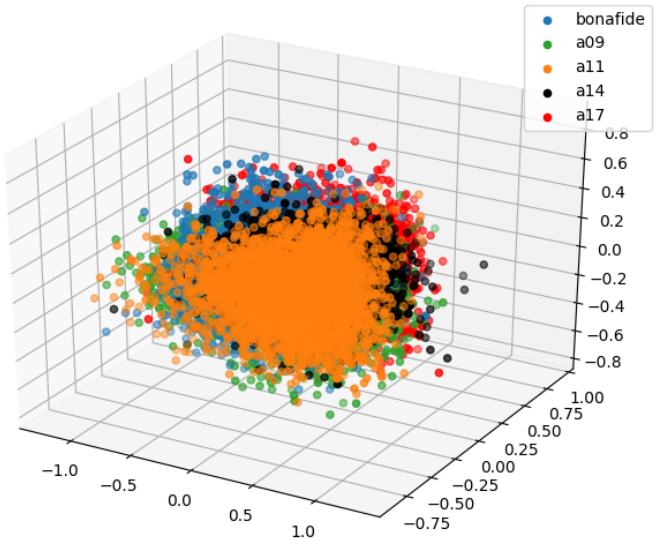


Figure 4.8: 3D scatter plot of the MKU features belonging to the classes bonafide, A09, A11, A14 and A17, reduced using PCA.

(see Chapter 1).

We notice that visually there has not been much improvement in the clustering, since the different classes still appear to be overlapping with each other. Anyway, we can still see an increase in the multiclass classification accuracy CSM, as shown in Figure 4.11

The accuracy of this classification is 80%.

As with the UNET features, we observe a 100% classification accuracy in the task of distinguishing the bonafide instances from all other classes of spoofed speech (see Figure 4.12).

From the results in this Section and the previous ones, we derive the conclusion that UNET features are actually able to add some discriminative power to the MK ones, yielding a sensible improvement in both

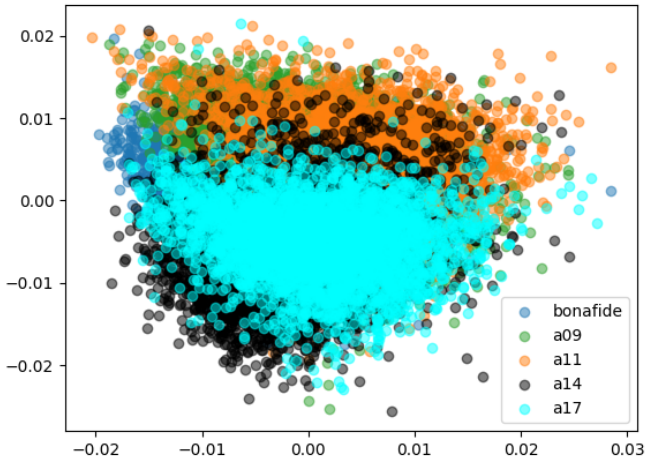


Figure 4.9: 2D scatter plot of the MKU features belonging to the classes bonafide, A09, A11, A14 and A17, reduced using ICA.

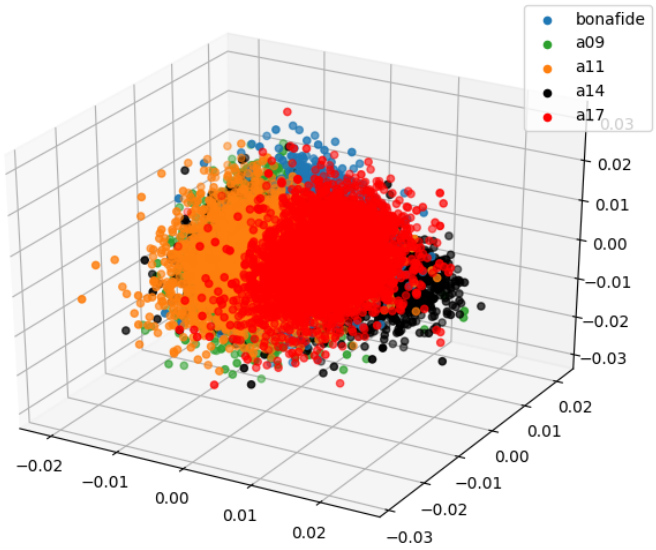


Figure 4.10: 3D scatter plot of the MKU features belonging to the classes bonafide, A09, A11, A14 and A17, reduced using ICA.

multiclass and binary classification accuracies.

4.3 Open set classification results

We switch now to the open set classification, which have been performed for both the MK and the MKU features.

As we already said in Chapter 3, we implemented three different versions of the open set classification algorithm: the first one does not make use of known-unknown features; on the contrary, the other two use the known-unknown features and differentiate from each other by how the test is performed, in particular as concerns the way the unknown features

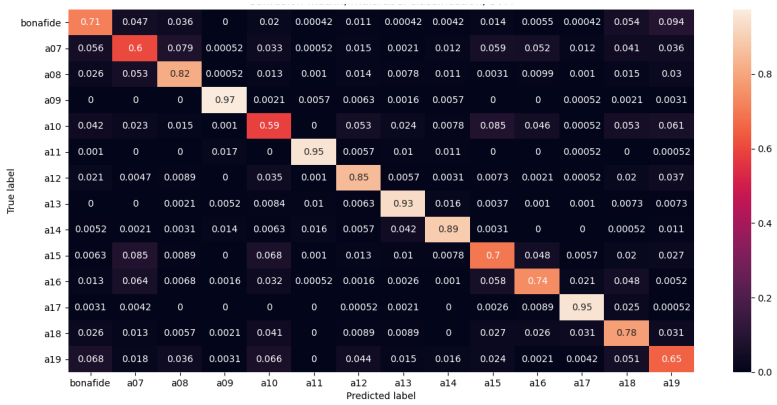


Figure 4.11: Confusion matrix of a multiclass classification (CSM) that involves MKU features, performed using a series of one-versus-the-rest support vector machines.

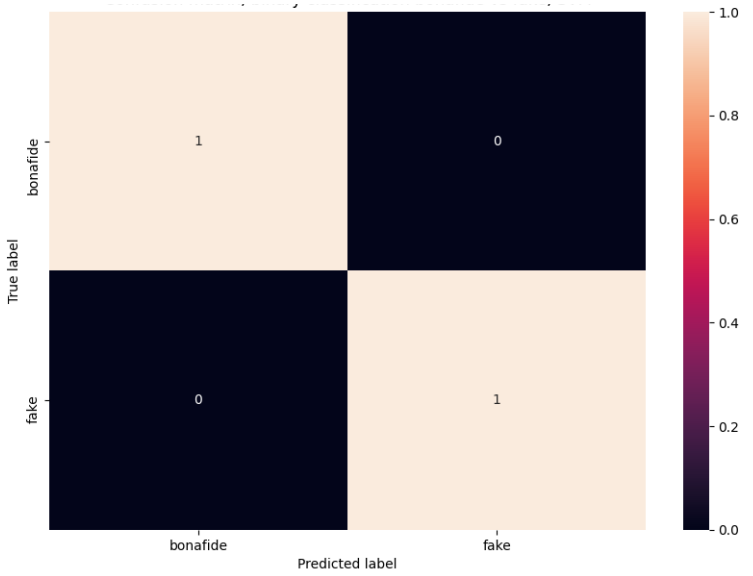


Figure 4.12: Confusion matrix of a binary (bonafide vs all spoof) (CSBVAS) classification that involves MKU features, performed using a support vector machine.

are labeled.

In the following you can find the results of these classifications.

4.3.1 Open set classification of MK features

We start from the MK features.

In Table 4.8 are summarised the classes of speech that have been used as known, known-unknown and unknown in the open set classification.

The next Figures show the confusion matrices associated to the classification performed with the three different versions of the open set classification algorithm using bonafide as known features (OSBT1, OSBT2

and OSBT3).

Classes of speech used in the open set classification	
Known	Bonafide, A09, A11, A14, A17
Known-unknown	A07, A08, A10
Unknown	A13, A15, A18, A19

Table 4.8: Summary of the classes of speech used in the open set classification.

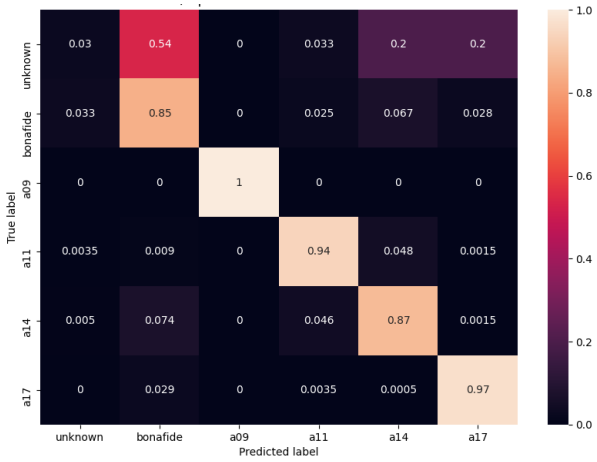


Figure 4.13: Confusion matrix of an open set classification that involves the MK features, without known-unknown features, performed using bonafide features as known (OSBT1).

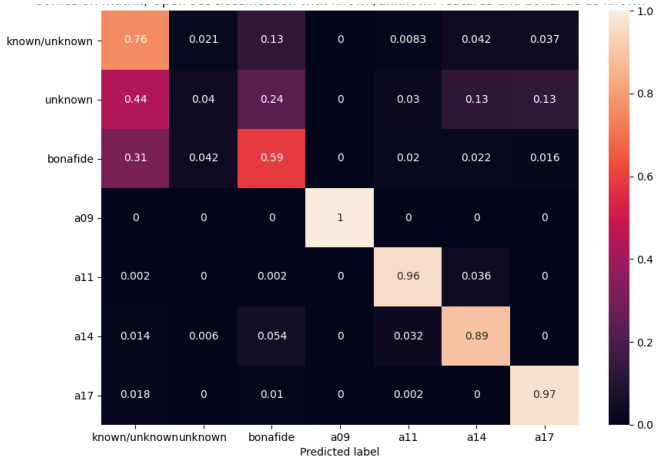


Figure 4.14: Confusion matrix of an open set classification that involves the MK features, with known-unknown features, performed using bonafide features as known (OSBT2).

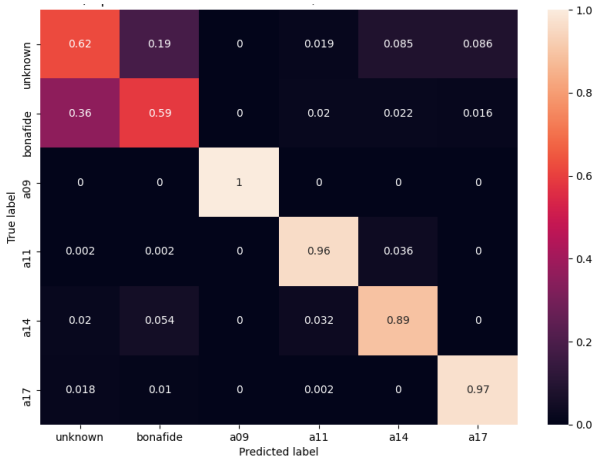


Figure 4.15: Confusion matrix of an open set classification that involves the MK features, with known-unknown features, performed using bonafide features as known (OSBT3).

We can see from Figure 4.13, which shows the result of the open set classification without known-unknown features (OSBT1), most of the unknown features are misclassified, and confused with the known ones (especially with the bonafide ones), while the known features are generally well classified.

The second version of the open set classification algorithm (OSBT2), whose results are shown in Figure 4.14 adds the known-unknown features, which have been assigned their own label. Here the real unknown features are still misclassified, but most of them are labeled as to belong to the known-unknown class.

In the third version of the classification algorithm (OSBT3), whose confusion matrix is shown in Figure 4.15, the known-unknown and real unknown classes have been unified, yielding a generally more positive performance (with an accuracy of 84%), even if some confusion with the bonafide class still persists.

In Figure 4.16 you can find the ROC curves we derived from the classification performed by the third version of our open set classification algorithm. These curves are the result of a classification procedure used to distinguish between known and unknown features. The blue curve is derived by the set of maximum scores of the five support vector machines involved in the training of the open set classification algorithm, while the orange one is based on the ratios between the maximum score and the score below the maximum. We used these values in order to discriminate between known and unknown features. The red dot in the Figure represents a working point associated to the confusion matrix derived by the third version of the classification algorithm. In this case, the ROC curve associated to the maximum scores represents the best method to discriminate between known and unknown features.

Now we concentrate on the second experiment, i.e. the one where

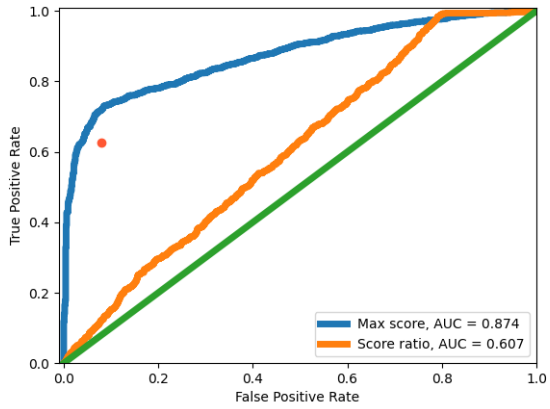


Figure 4.16: Comparison between the three methods for distinguishing known and unknown features that consist in the ROC curves derived by the third version of the open set classifier using the scores of the SVMs and the confusion matrix of the third classifier, for MK features with bonafide as known. The blue curve corresponds to the maximum scores, while the orange one is associated to the ratio of the scores of the SVMs. The red dot represents the confusion matrix related to the third version of the open set classification algorithm (OSBT3).

instances of the Bonafide class are used as real unknown features.

Classes of speech used in the open set classification	
Known	A09, A11, A13, A14, A17
Known-unknown	A07, A08, A10
Unknown	Bonafide, A15, A18, A19

Table 4.9: Summary of the classes of speech used in the open set classification.

In Table 4.9 are summarised the classes used in the open set classification.

Figures 4.17, 4.18 and 4.19 show the confusion matrices associated to the three versions of the open set classification algorithm (OSST1, OSST2 and OSST3).

As before, we can notice that all the three classifiers correctly label most of the known features. Anyway, the first version of the open set classifier confuses most of the unknown instances for known.

Similarly to the first experiment, in the second version of the classifier the majority of the real unknown features are labeled as known-unknown. This allows us to have a general good performance for the third version, where, as before, the known-unknown and real unknown classes have been merged together. The accuracy of this latter classification is 89%.

In Figure 4.20 are shown, like in the first experiment, the two ROC curves associated to the classification between known and unknown fea-

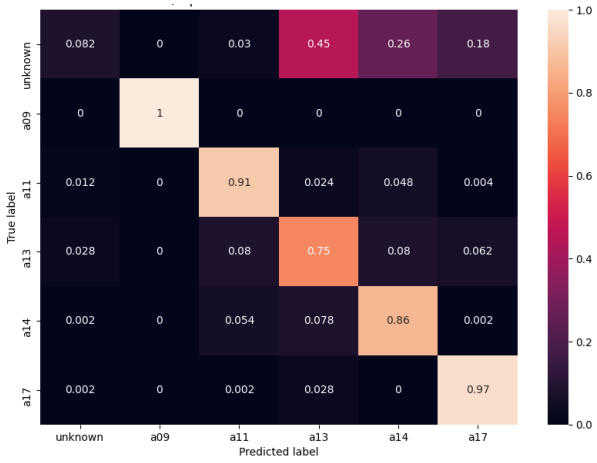


Figure 4.17: Confusion matrix of an open set classification that involves the MK features, without known-unknown features, performed using bonafide features as unknown (OSST1).

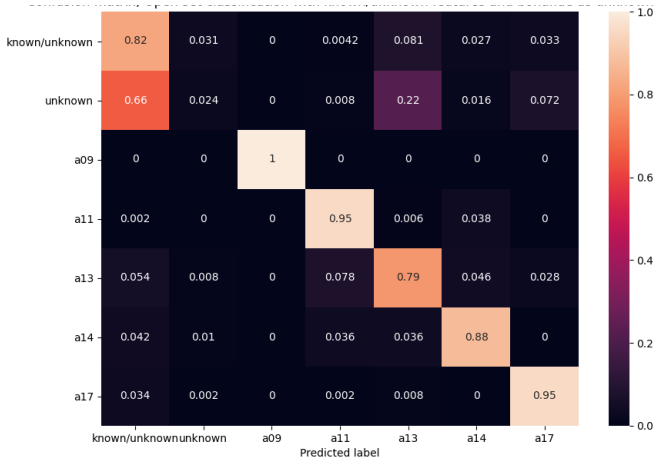


Figure 4.18: Confusion matrix of an open set classification that involves the MK features, with known-unknown features, performed using bonafide features as unknown (OSST2).

tures, taking as parameters the scores of the support vector machines associated to the third version of the classification algorithm, together with the point corresponding to the confusion matrix of that classifier (OSST3). In this case, the red dot lies above the curves, which means that the third version of the open set classifier discriminates between known and unknown features better than the other two methods based on the scores of the SVMs.

4.3.2 Open set classification of MKU features

Now we repeat all the open set classifications performed with MK features for the MKU ones, which, as we anticipated, are the ones that perform

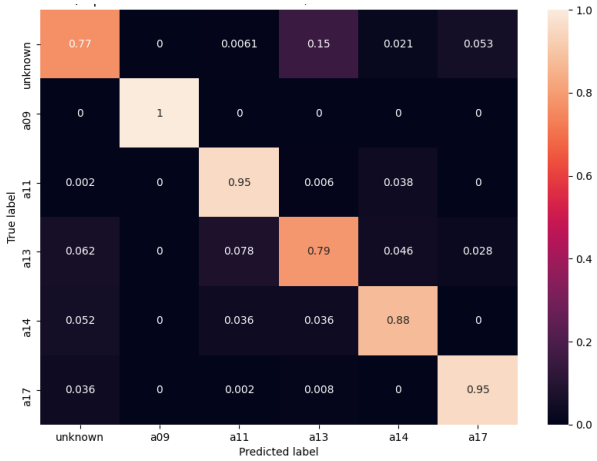


Figure 4.19: Confusion matrix of an open set classification that involves the MK features, with known-unknown features, performed using bonafide features as unknown (OSST3).

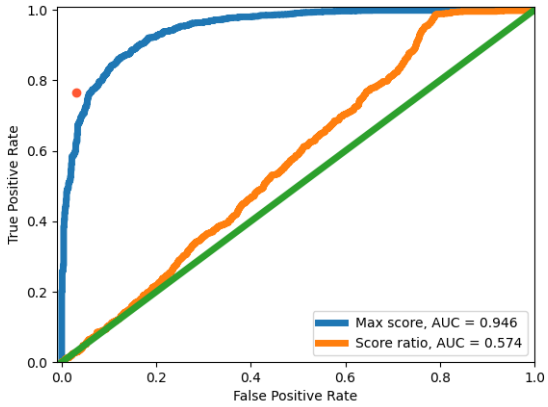


Figure 4.20: Comparison between the three methods for distinguishing known and unknown features that consist in the ROC curves derived by the third version of the open set classifier using the scores of the SVMs and the confusion matrix of the third classification algorithm, for MK features with bonafide as unknown. The blue curve corresponds to the maximum scores, while the orange one is associated to the ratio of the scores of the SVMs. The red dot represents the confusion matrix related to the third version of the open set classification algorithm (OSST3).

better also on the open set environment. The sets of known, known-unknown and unknown features are analogous to the ones used to classify the MK features.

As concerns the first experiment (i.e. the one in which bonafide instances are among the known calsses), Figures 4.21, 4.22 and 4.23 show the confusion matrices associated to the classifications performed by the three versions of the open set classification algorithm (OSBT1, OSBT2 and OSBT3).

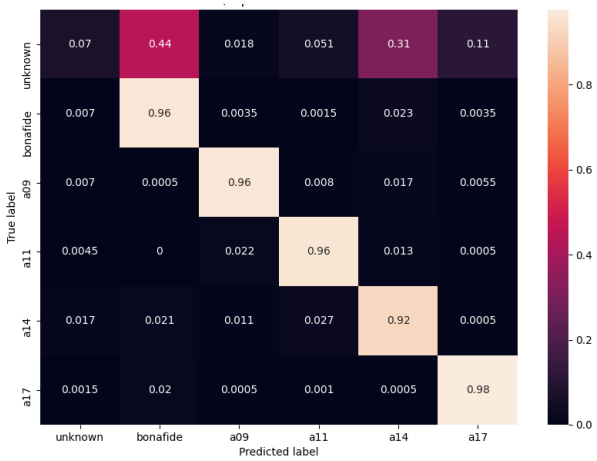


Figure 4.21: Confusion matrix of an open set classification using the MKU features, without known-unknown features, performed using bonafide features as known (OSBT1).

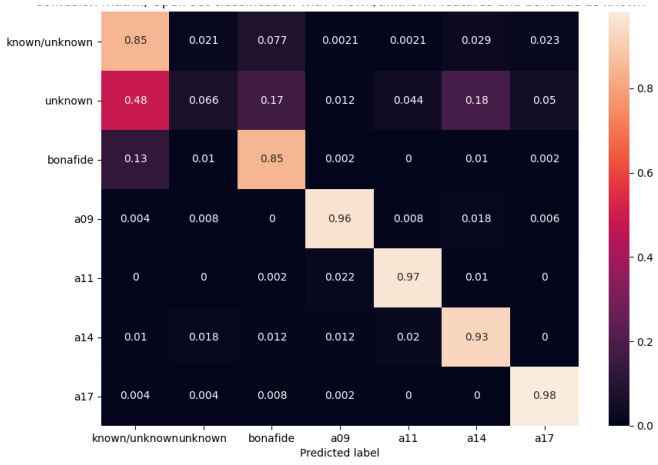


Figure 4.22: Confusion matrix of an open set classification using the MKU features, with known-unknown features, performed using bonafide features as known (OSBT2).

As usual, in the first confusion matrix the unknown features are generally labeled as known (and in particular most of them as Bonafide), in the second matrix most of the unknown features are exchanged for real unknown features, and finally the third matrix shows a cleaner result. The accuracy of this latter classification is 90%.

Figure 4.24 shows the ROC curves of the binary classifications between known and unknown features derived by the third version of the open set classifier.

Now we concentrate on the second experiment performed using the bonafide features as unknown.

In Figures 4.25, 4.26 and 4.27 are shown the confusion matrices of the classifications performed by the three open set classification algorithms

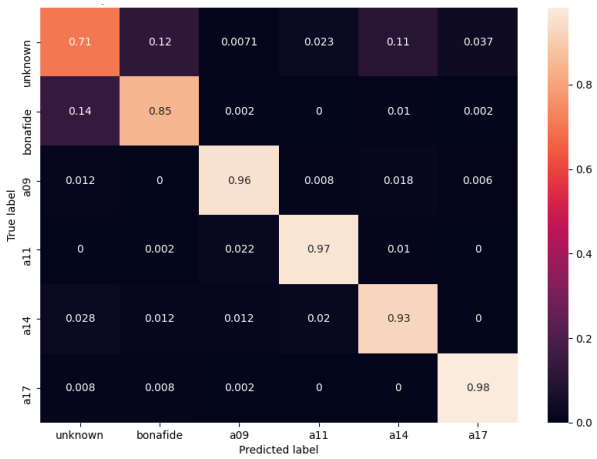


Figure 4.23: Confusion matrix of an open set classification using the MKU features, with known-unknown features, performed using bonafide features as known (OSBT3).

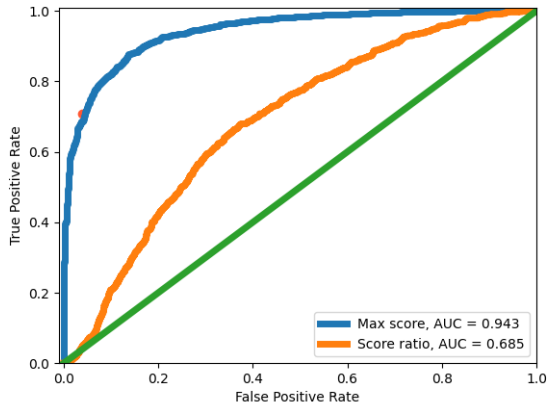


Figure 4.24: Comparison between the three methods for distinguishing known and unknown features that consist in the ROC curves derived by the third version of the open set classifier using the scores of the SVMs and the confusion matrix of the third classification algorithm, for MKU features with Bonafide as known. The blue curve corresponds to the maximum scores, while the orange one is associated to the ratio of the scores. The red dot represents the confusion matrix related to the third version of the open set classification algorithm (OSBT3).

(OSST1, OSST2 and OSST3).

Also in this case, the first classifier proves to be very pessimistic in the classification of the unknown instances; the second one still confuses real unknown features for known-unknown, while the third achieves a good performance, with an accuracy of almost 93%.

Finally, we show the ROC curves associated to the classification of known and unknown features (Figure 4.28), derived, as usual, from the scores of the SVMs used in the training of the third version of the open

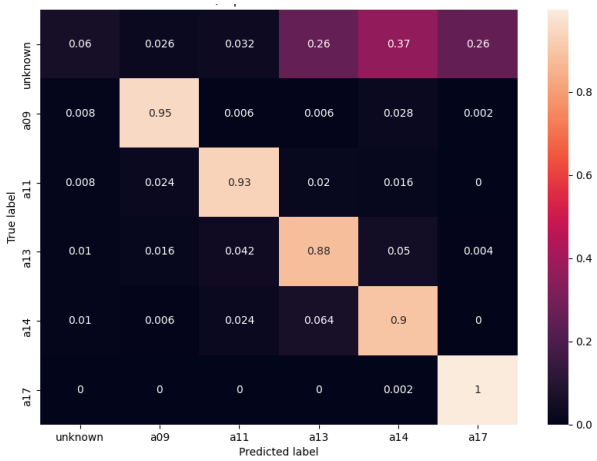


Figure 4.25: Confusion matrix of an open set classification performed using the MKU features, without known-unknown features, and bonafide features as unknown (OSST1).

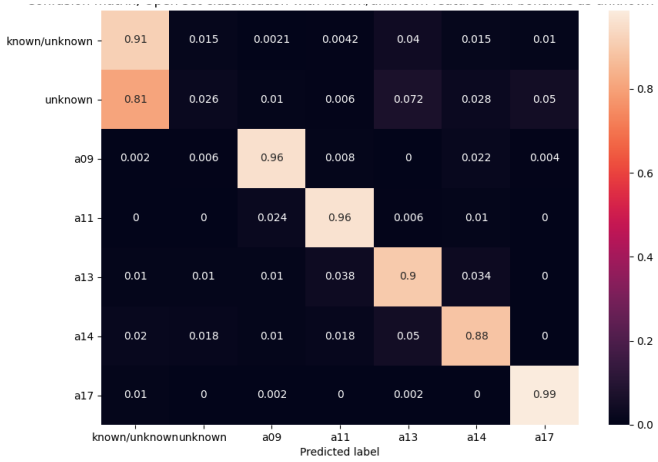


Figure 4.26: Confusion matrix of an open set classification performed using the MKU features, with known-unknown features, and bonafide features as unknown (OSST2).

set classification algorithm.

From the results shown in this Section and the above ones, we can say that MKU features, which are the combination of both traditional (MK), and data driven (UNET) features, are the ones that perform better in both the closed set and open set classifications. This is coherent with the results found in [55]. In the binary closed set classification (CSBVAS) we achieve an accuracy of 100%, while in the multiclass case (CSM) the best performance is characterised by an accuracy of 80%, which corresponds to the classifier implemented using a series of one-versus-the-rest support vector machines. For the open set case, the best results are the ones associated to the third version of our classification algorithm (OSBT3 and OSST3), where we labeled as "unknown" features that were assigned to

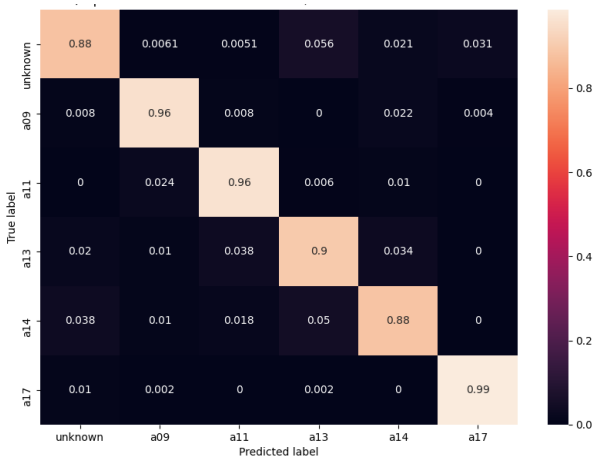


Figure 4.27: Confusion matrix of an open set classification performed using the MKU features, with known-unknown features, and bonafide features as unknown (OSST3).

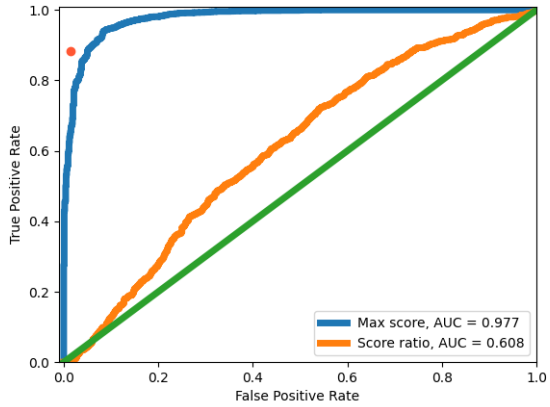


Figure 4.28: Comparison between the three methods for distinguishing known and unknown features that consist in the ROC curves derived by the third version of the open set classifier using the scores of the SVMs and the confusion matrix of the third classification algorithm, for MKU features with bonafide as unknown. The blue curve corresponds to the maximum scores, while the orange one is associated to the ratio of the scores. The red dot represents the confusion matrix related to the third version of the open set classification algorithm (OSST3).

both the "real unknown" and "known-unknown" classes. In particular, the best case scenario is the one where the bonafide instances are used as real unknown (OSST3), since in the other case (i.e. when the bonafide is used as known) there is some confusion between the unknown features and the bonafide ones. Finally, the best method to distinguish between known and unknown features is represented by the confusion matrix of the third version of the open set classification algorithm, even if good (and in one case also better) performances are achieved by the classifi-

cation performed using, for each input feature set, the maximum of the scores associated to the five one-versus-the-rest support vector machines involved in the training of the third version of the open set classification algorithm.

5
Conclusions and Future Works

The diffusion of digital multimedia content, together with the spread of artificial intelligence (AI) methods which are able to generate instances of fake audio, image and video signals, is taking us to a point where it is always more urgent to discriminate about the authenticity of this content. Speech recordings are particularly delicate as it comes to spoofing, since our voice is the most natural way to perform biometric person recognition.

The work we have done for this thesis has been motivated by the need of developing automatic procedures able to detect and recognise fake speech (i.e. speech synthesised using computing systems and artificial intelligence techniques), and to distinguish it from recordings of speech generated by real speakers.

Previous works in the field of audio forensics like [1] and [4] have showed us the path to follow, which is characterised by the use of bispectral analysis. The exploitation of the bicoherence, which is a complex feature that can be extracted from speech signals (and audio signals in general) can deliver state of the art performances in the distinction of real and fake speech. This is due to the fact that nonlinear processing that is involved in spoofing operations leaves on the synthesised signal higher order correlations which cannot be found on bonafide speech. These correlations can be detected by means of a third order feature like the bicoherence.

In the previous Chapter, we have shown the results of a number of classifications apt to distinguish real and fake speech recordings (by means of binary classification algorithms), discriminate among the classes of fake speech (using multiclass classifications) and detect the presence of instances belonging to classes of speech never seen by our models during

the training phase (simulating an open set environment).

The features involved in these classifications, which have been described in detail in Chapter 3, have been proved to deliver interesting results. In particular, the MK features, which consist in the first four statistical moments computed from both the modules and the phases of the bicoherences, had already been used in [1] to perform a multiclass classification which yielded good performances. A new set of features (namely UNET features), which does not have precedents in literature, has been extracted from the bicoherences in a data driven fashion, by making use of a series of convolutional autoencoders called U-Nets. These features proved to be very meaningful in the distinction of bonafide and spoofed speech recordings. The best results, anyway, have been achieved by the MKU features, which have been generated by means of a simple concatenation of the MK and UNET ones. This latter set yields the most interesting performances in both the closed set (binary and multiclass) and the open set classification.

The open set classification is the one that really characterises this work, since it has never been performed before with the aim of detecting digital forgeries in speech signals. The results obtained up to now show that the best performances are achieved by classifiers that make use of known-unknown features in the training. These are features which in theory are "known", since they are used to train the model, but that in practice are considered as "unknown", and are used to model the "unknown" class.

Another peculiarity of our work is the dataset that we have used to perform our experiments; as we have said in Chapter 4, ASVspoof 2019 is an incredibly various and heterogeneous database of speech signals, which comprehends both bonafide speech and speech synthesised using many different state of the art spoofing algorithms. For this reason it is particularly suitable for the task of recognising real speakers from AI generated speech.

We hope that the results we have achieved could better clarify about the interesting properties of the bicoherence, used as descriptor of speech signals, which has not been subject of many studies until now, and could lead to more research in the field of audio forensics.

In particular, future work may consists in investigating more on the nature of the higher order correlations left by spoofing operations on speech signals, in order to find even more efficient descriptors, which can be derived using bispectral analysis. These descriptors would be useful to better characterise spoofed speech signals as concerns the synthesis algorithm used to generate them.

Moreover, other experiments involving convolutional neural networks could be performed, for example as concerns the simulation of an open set environment or the extraction of data driven features using the phases of the bicoherences.

We hope that the development of more advanced and easy to use

forensics techniques could lead to avoiding the misleading impact of fake multimedia content in our every day life.

Appendices

A Results of the classification of the Bicoherences using a convolutional neural network

In this Appendix we are going to show the results obtained from the classification that involves the modules of the bicoherences performed using a simple convolutional neural network (CNN).

Layer	Output shape	# Parameters
conv2d (Conv2D)	(None, 128, 128, 32)	320
maxpooling2d (MaxPooling2D)	(None, 64, 64, 32)	0
conv2d1 (Conv2D)	(None, 64, 64, 64)	18496
maxpooling2d1 (MaxPooling2D)	(None, 32, 32, 64)	0
conv2d2 (Conv2D)	(None, 32, 32, 64)	36928
flatten (Flatten)	(None, 65536)	0
dense (Dense)	(None, 2)	131072

Table 1: Summary of all the layers of the convolutional neural network used for the classification.

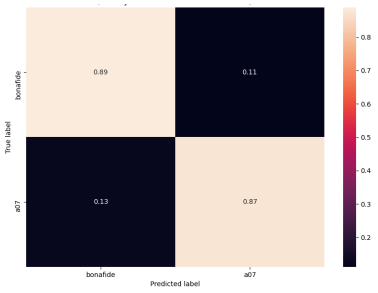
In Table 1 is shown the structure of this network, which has been implemented using the Python libraries Tensor Flow and Keras [62] [63].

The experiments we have done consist of a multiclass classification (CSM), a binary classification (CSBVAS) where we tried to distinguish bonafide speech from all the fake instances, and a series of two-class classification (CSBVSS), where each one is apt to recognise bonafide speech from every single class of spoofed recordings (namely A07, A08,...,A19).

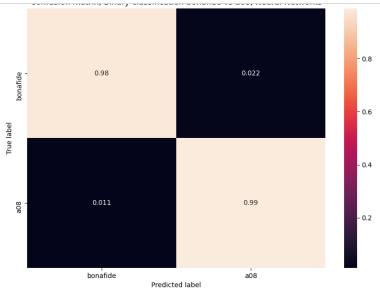
In Figure 1 is shown the confusion matrix associated to the multiclass classification. The accuracy of this classification is 74%.

In Figure 2 is shown the binary classification that involves all the classes of fake speech taken together. The accuracy of this classification is 78%.

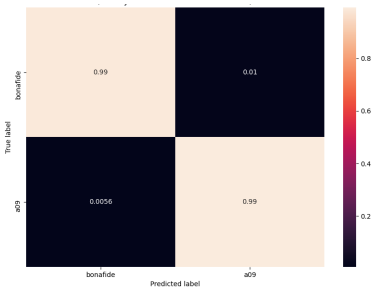
Finally, in Tables 2 and 3 are shown the confusion matrices associated to the two-class classifications performed using the modules of the bicoherences where each classification involves the Bonafide and a single class of spoofed speech.



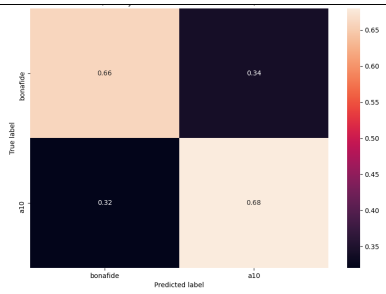
Bonafide vs A07



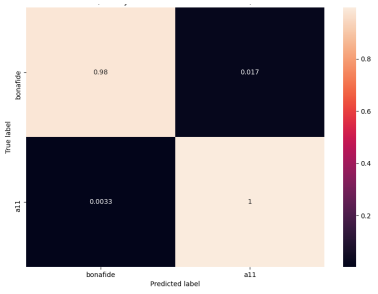
Bonafide vs A08



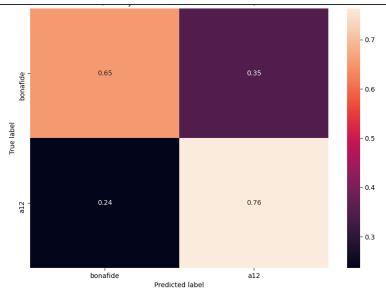
Bonafide vs A09



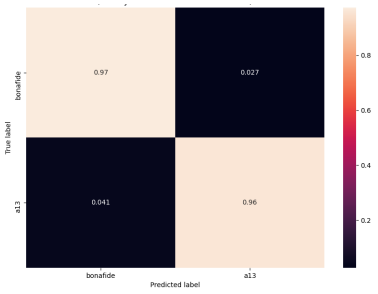
Bonafide vs A10



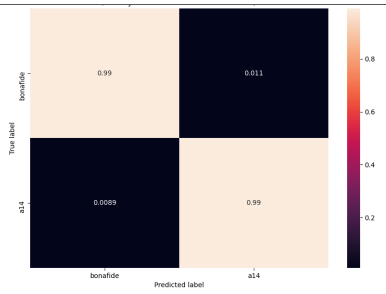
Aonafide vs A11



Bonafide vs A12



Bonafide vs A13



Bonafide vs A14

Table 2: Binary classifications (Bonafide vs each one of the fake classes) involving the modules of the bicoherences, performed using a convolutional neural network.

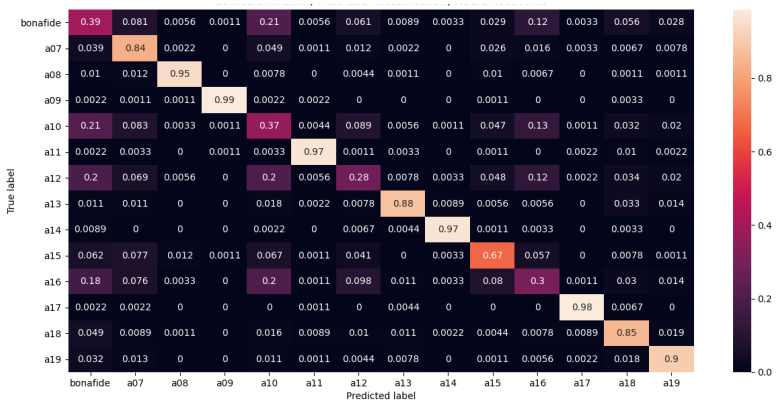


Figure 1: Confusion matrix of a multiclass classification that involves the modules of the bicoherences, performed using a convolutional neural network.

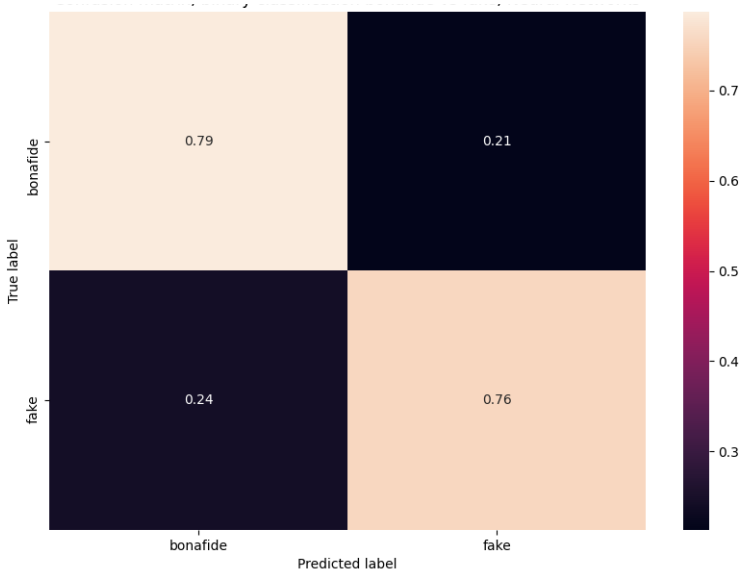


Figure 2: Confusion matrix of a binary (bonafide vs all spoof) classification that involves the modules of the bicoherences, performed using a convolutional neural network.

From these results we can say that, while our neural network performs quite well in the task of distinguishing bonafide and spoofed speech, other classification methods achieve better performances (for example the one that involves MKU features).

B Results of the classification using a U-Net trained on bonafide bicoherences

This Appendix shows the results obtained from a U-Net used to perform anomaly detection. The algorithm consists in training a U-Net with the

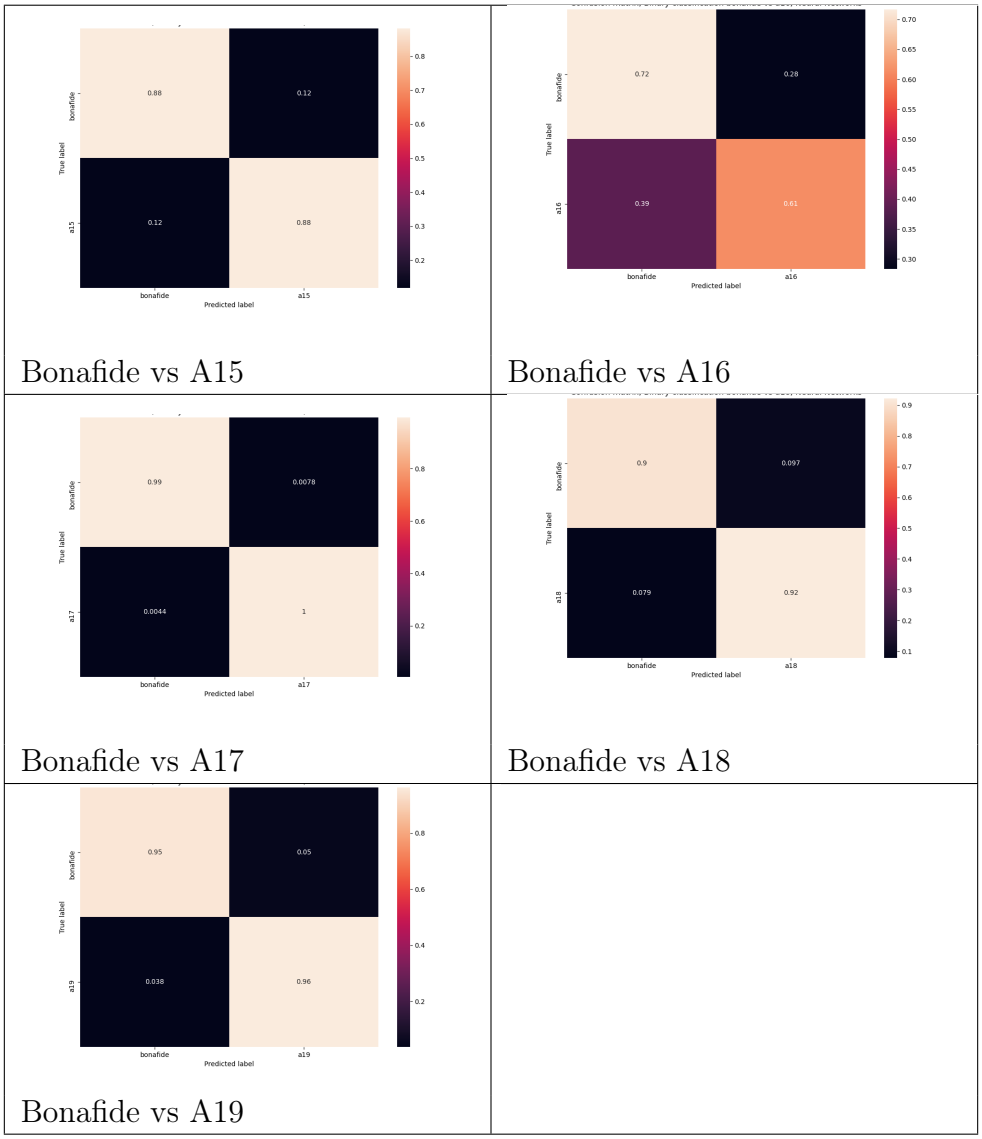


Table 3: Binary classifications (Bonafide vs each one of the fake classes) involving the modules of the bicoherences, performed using a convolutional neural network.

modules of the bicoherences extracted from bonafide speech, and testing it with instances belonging to all the other classes.

Mean square errors are computed for each test instance as described in Chapter 3, when talking about the extraction of the UNET features. After that, for each class we computed the mean MSE, and displayed these values in an histogram.

In general, what we expect is that the mean MSE associated to the Bonafide class would be lower than the ones associated to the other classes. This is due to the fact that bonafide instances should be reconstructed better than all the others by the network, since the training has been performed using bonafide instances.

We also computed a series of ROC curves, one for each class of fake

speech, having as input a vector of MSEs (one for each instance) and a label indicating the class that the instance belongs to (Bonafide or Fake). This curves are used to distinguish bonafide instances from instances belonging to each one of the fake classes of speech.

A final ROC curve, that summarises all the previous ones, is used to distinguish Bonafide from all the classes of fake speech.

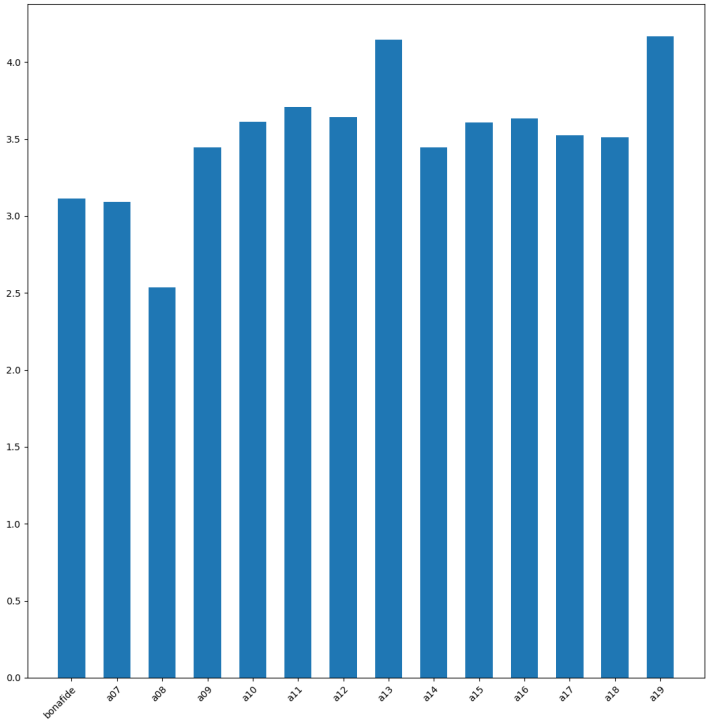


Figure 3: Histogram that shows the mean MSE for each class of speech, computed using a U-Net trained with the modules of the bicoherences extracted from bonafide speech.

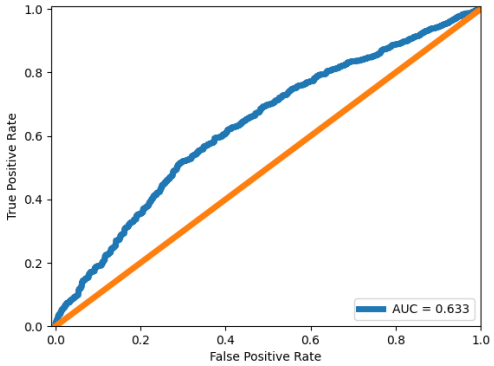


Figure 4: ROC curve used to distinguish bonafide from all the classes of spoofed speech taken together, obtained from the MSEs extracted by the U-Net trained with bonafide instances.

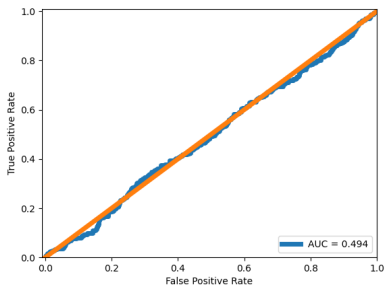
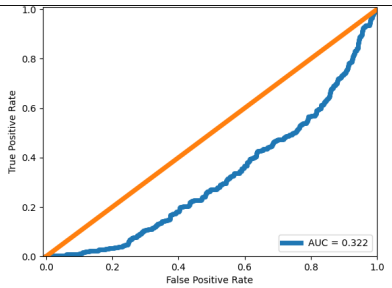
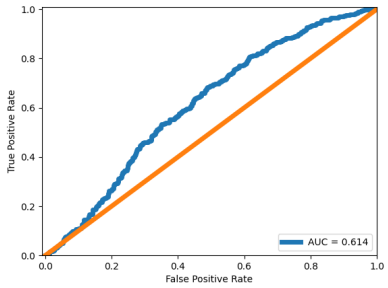
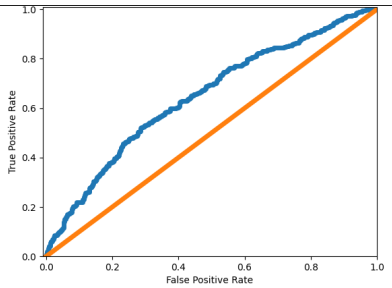
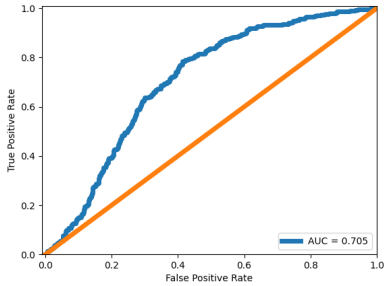
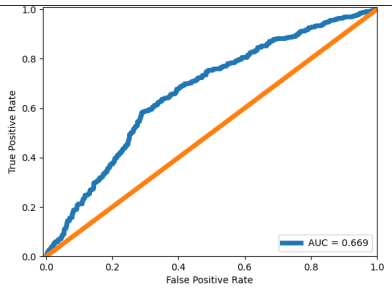
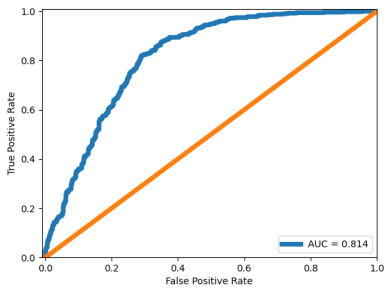
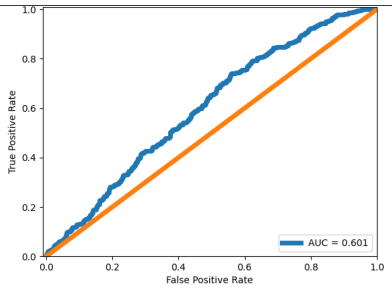
	
Bonafide vs A07	Bonafide vs A08
	
Bonafide vs A09	Bonafide vs A10
	
Bonafide vs A11	Bonafide vs A12
	
Bonafide vs A13	Bonafide vs A14

Table 4: ROC curves obtained from the MSEs computed from the U-Net trained with the modules of the bicoherences extracted from bonafide speech, one for each fake class.

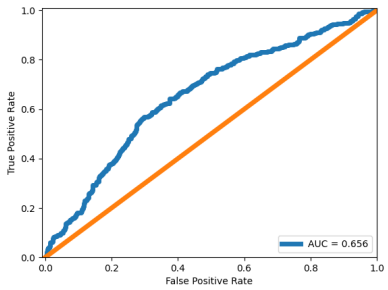
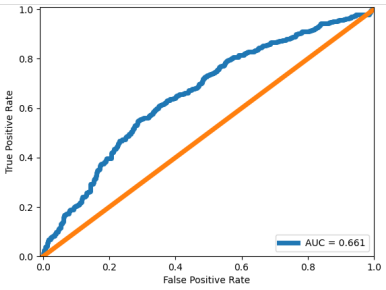
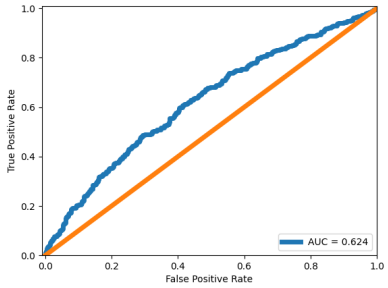
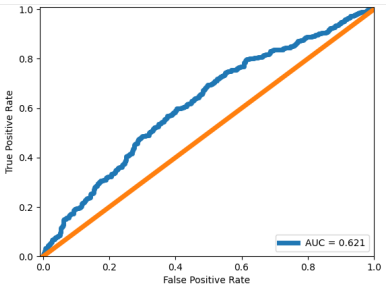
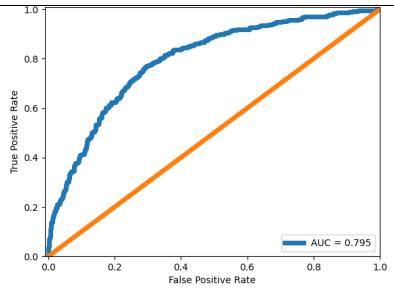
 Bonafide vs A15	 Bonafide vs A16
 Bonafide vs A17	 Bonafide vs A18
 Bonafide vs A19	

Table 5: ROC curves obtained from the MSEs computed from the U-Net trained with the modules of the bicoherences extracted from bonafide speech, one for each fake class.

In Figure 3 is shown the histogram of all the mean MSEs, while Tables 4 and 5 show the ROC curves related to each spoofed class of speech. Finally, in Figure 4 can be found the final ROC curve, that gives a measure of how well bonafide and spoofed speech can be distinguished using the MSEs extracted using the UNET trained with bonafide instances.

Clearly, we can say that this method achieves poor performances, since both bonafide and spoofed instances of the modules of the bicoherences are perfectly reconstructed by the network.

C Confusion matrixes of all the closed set classification of MK, UNET and MKU features

In this final Appendix we are going to show the confusion matrices of all the closed set classifications we have performed. Some of them have already been shown, while others did not find their place in Chapter 4. For each set of features (MK, UNET and MKU), in this Appendix are present the following results: three different confusion matrices associated to multiclass classifications (CSM) performed with support vector machines, logistic regressions and a random forest (see Chapter 1); three confusion matrices associated to binary (bonafide versus all the fake classes) classifications (CSBVAS) performed using the same algorithms as for the multiclass case; a series of confusion matrices associated to binary (bonafide versus each one of the fake classes) classifications (CSBVSS) performed using a SVM. The matrices associated to classifications whose performance reach 100% accuracy are omitted. Finally, we are going to show a Table with the accuracies of all the closed set classifications, including the ones performed by the convolutional neural network which can be found in Appendix A.

C.1 Confusion matrixes of the closed set classifications of MK features

We start from the MK features: in Figures 5, 6 and 7 are shown the confusion matrices associated to the multiclass classifications (CSM). The accuracies of these classifications are respectively 62%, 49% and 58%.

In Figures 8, 9 and 10 are shown the confusion matrices associated to the two-class classifications (bonafide versus all spoof) (CSBVAS). The accuracies of these classifications are respectively 76%, 67% and 79%.

Finally, in Tables 6 and 7 are shown the confusion matrices associated to the binary classifications which distinguish the bonafide class from each one of the fake classes (CSBVSS).

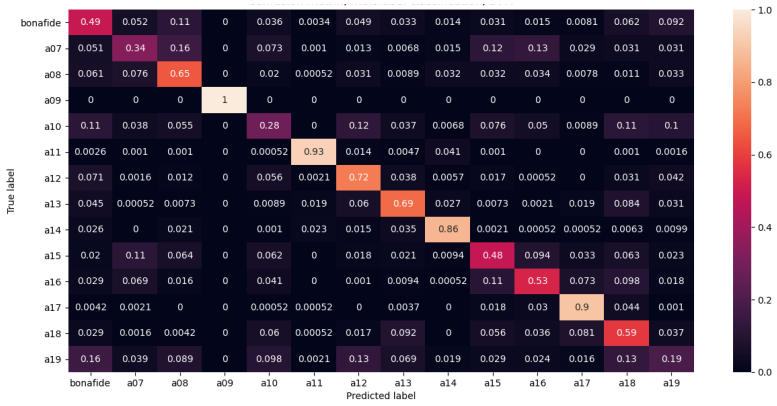


Figure 5: Confusion matrix of a multiclass classification (CSM) that involves the MK features, performed using a series of one-versus-the-rest support vector machines.

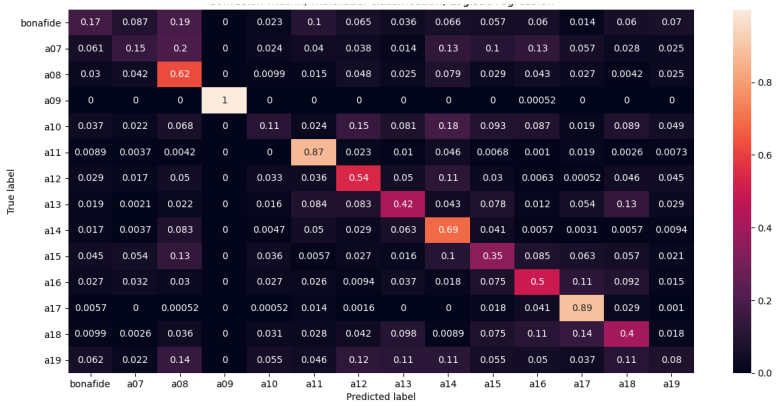


Figure 6: Confusion matrix of a multiclass classification (CSM) that involves the MK features, performed using a series of one-versus-the-rest logistic regressions.

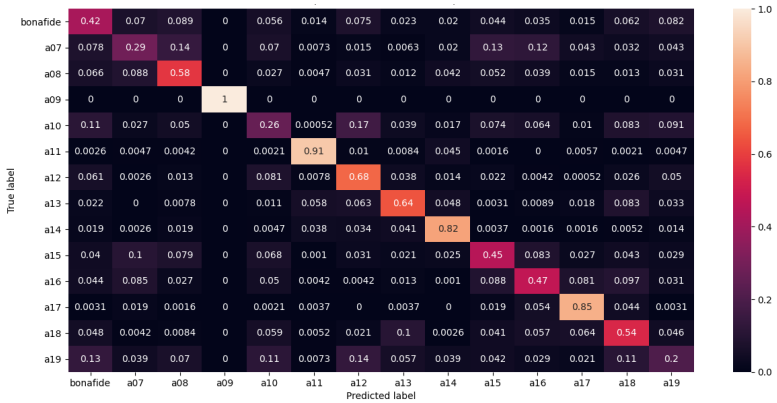


Figure 7: Confusion matrix of a multiclass classification (CSM) that involves the MK features, performed using a random forest.

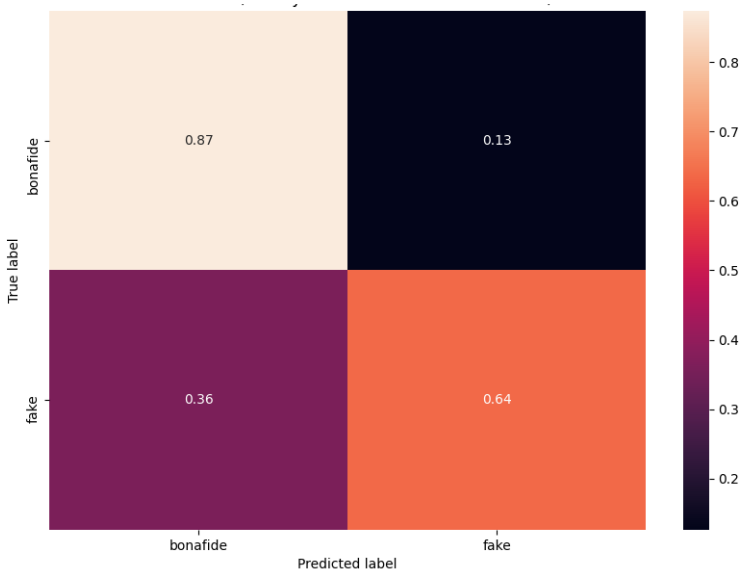


Figure 8: Confusion matrix of a binary (bonafide vs all spoof) (CSBVAS) classification that involves the MK features, performed using a support vector machine.

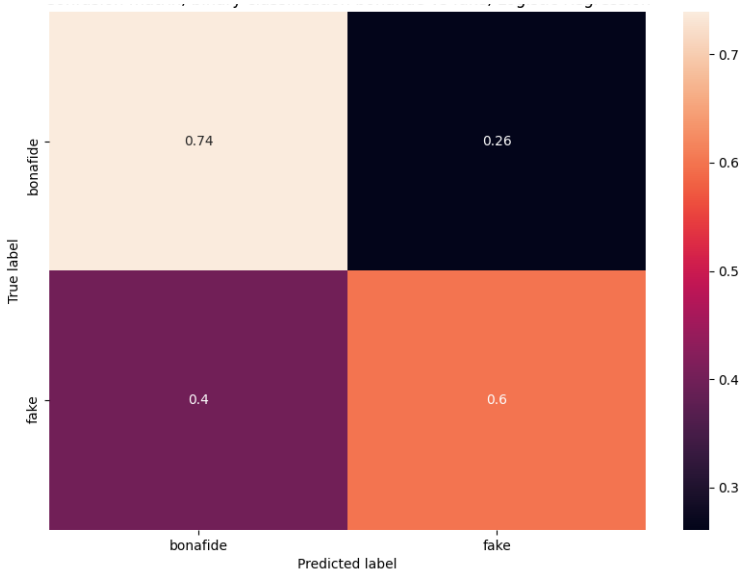
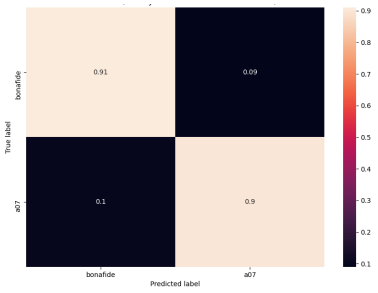
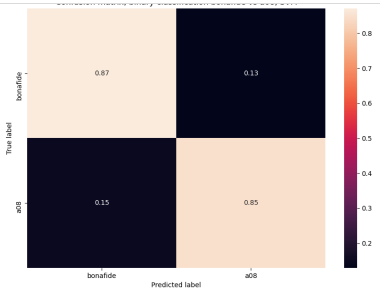


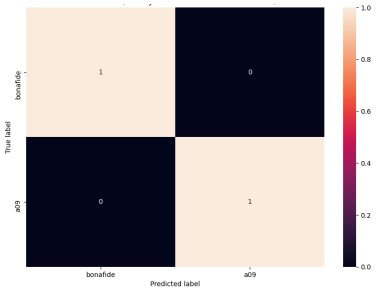
Figure 9: Confusion matrix of a binary (bonafide vs all spoof) (CSBVAS) classification involving the MK features using, performed a logistic regression.



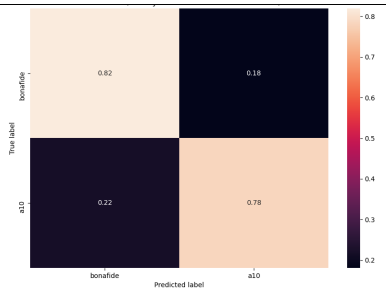
Bonafide vs A07



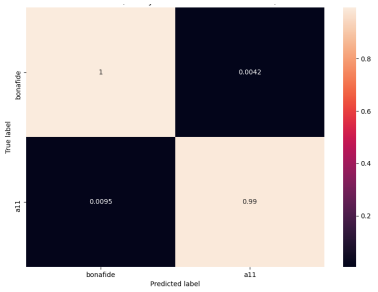
Bonafide vs A08



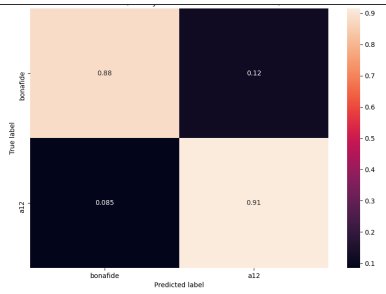
Bonafide vs A09



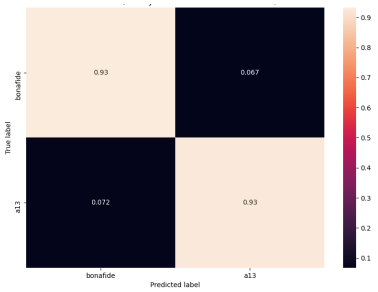
Bonafide vs A10



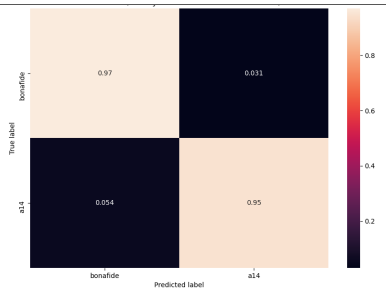
Bonafide vs A11



Bonafide vs A12

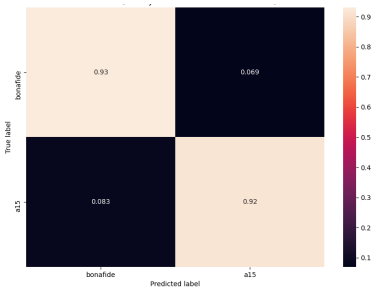


Bonafide vs A13

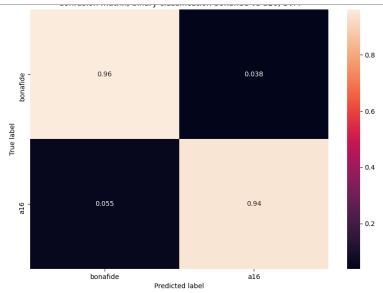


Bonafide vs A14

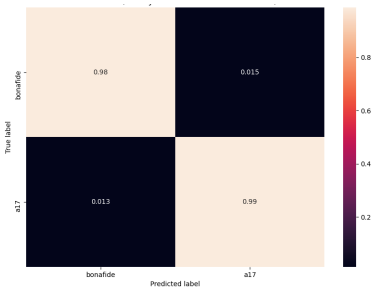
Table 6: Binary classifications (Bonafide vs each one of the fake classes) (CSBVSS) involving the MK features, performed using a support vector machine.



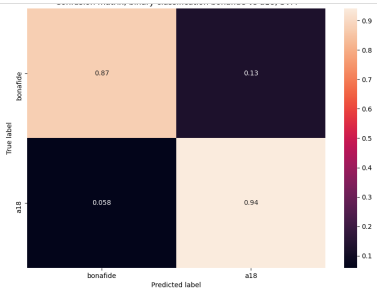
Bonafide vs A15



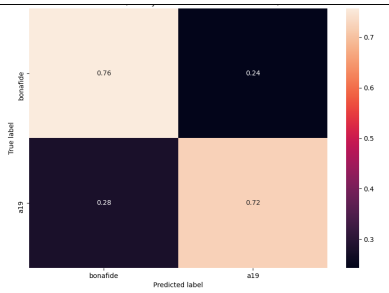
Bonafide vs A16



Bonafide vs A17



Bonafide vs A18



Bonafide vs A19

Table 7: Binary classifications (bonafide vs each one of the fake classes) (CSBVSS) that involve the MK features, performed using a support vector machine.

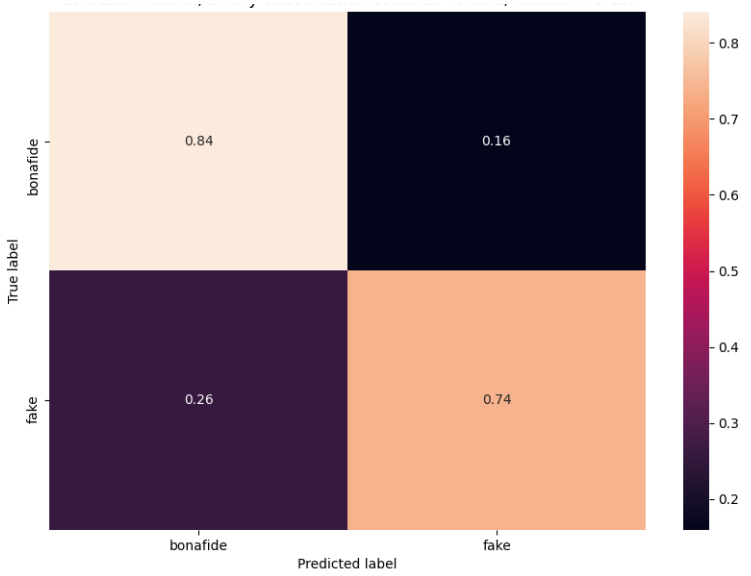


Figure 10: Confusion matrix of a binary (bonafide vs all spoof)(CSBVAS) classification that involves the MK features, performed using a random forest.

C.2 Confusion matrices of the closed set classifications of UNET features

We switch now to the UNET features.

Figure 11: Confusion matrix of a multiclass classification (CSM) that involves the UNET features, performed using a series of one-versus-the-rest support vector machines.

In Figures 11, 12 and 13 are shown the confusion matrices of the multiclass classifications (CSM). The accuracies associated to these classifications are respectively 35%, 30% and 32%.

The confusion matrices associated to the binary classifications that distinguish bonafide from fake speech (CSBVAS and CSBVSS) performed using the UNET features are omitted here, since the accuracy of these classifications is always 100%.

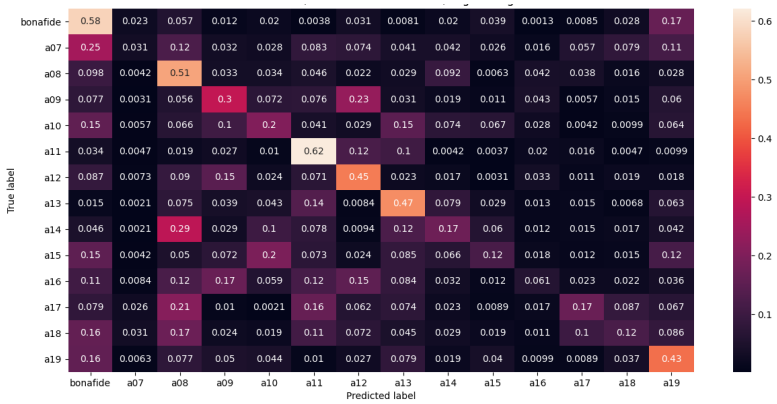


Figure 12: Confusion matrix of a multiclass classification (CSM) that involves the UNET features, performed using a series of one-versus-the-rest logistic regressions.

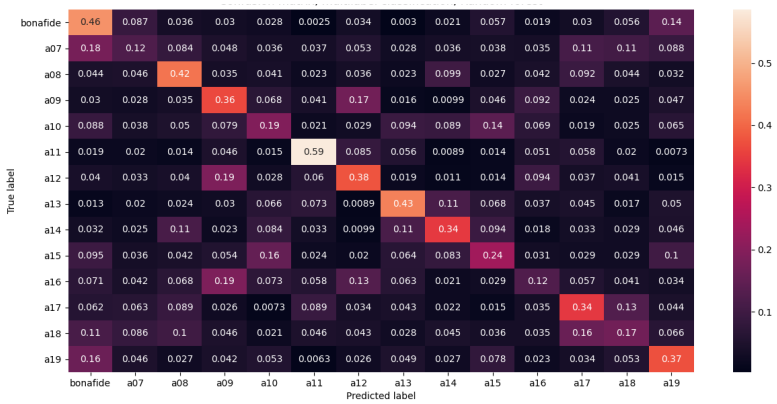


Figure 13: Confusion matrix of a multiclass classification (CSM) that involves the UNET features, performed using a random forest.

C.3 Confusion matrices of the closed set classifications of MKU features

Finally, we take care now of the mean-kurtosis-UNET (MKU) features, which are the ones that perform better in all the experiments that we have performed.

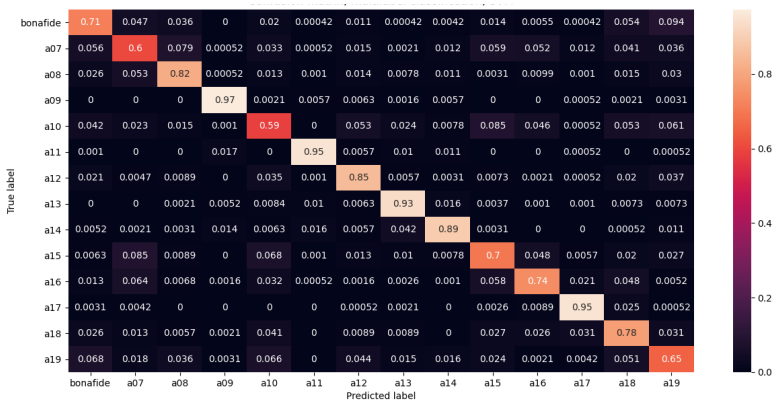


Figure 14: Confusion matrix of a multiclass classification (CSM) that involves the MKU features, performed using a series of one-versus-the-rest support vector machines.

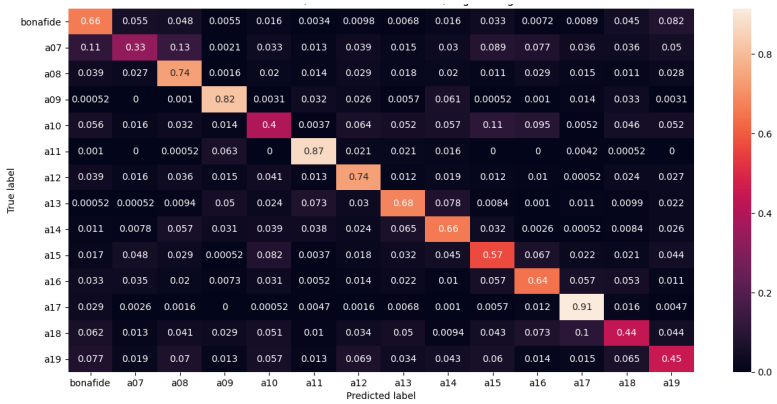


Figure 15: Confusion matrix of a multiclass classification (CSM) that involves the MKU features, performed using a series of one-versus-the-rest logistic regressions.

In Figures 14, 15 and 16 are shown the confusion matrices of the multiclass classifications (CSM). The accuracies of these classifications are respectively 80%, 64% and 67%

The confusion matrices associated to the binary classifications are omitted, since they all achieve 100% accuracy.

To end this thesis, we show in Table 8 the classification accuracies of all the closed set classifications.

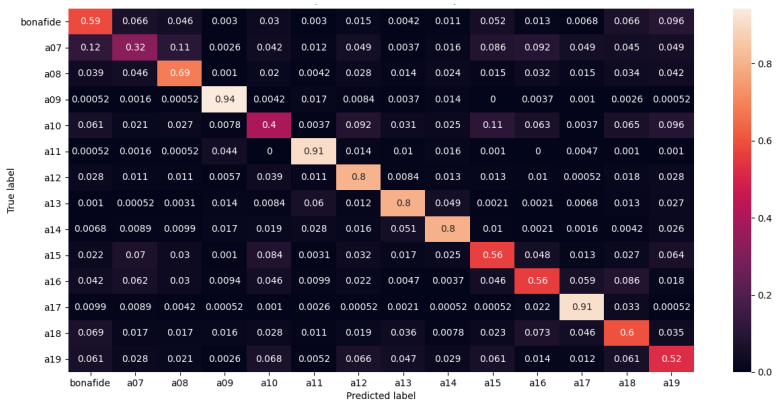


Figure 16: Confusion matrix of a multiclass classification (CSM) that involves the MKU features, performed using a random forest

Classification	MK features	Neural Networks	UNET features	MKU features
Multiclass	0.6178 / 0.485 / 0.5793	0.7386	0.3507 / 0.3034 / 0.3245	0.795 / 0.6364 / 0.6714
Bonafide vs all fake	0.755 / 0.67 / 0.79	0.775	1 / 1 / 1	1 / 1 / 1
Bonafide vs a07	0.905	0.88	1	1
Bonafide vs a08	0.86	0.985	1	1
Bonafide vs a09	1.0	0.99	1	1
Bonafide vs a10	0.8	0.67	1	1
Bonafide vs a11	0.995	0.99	1	1
Bonafide vs a12	0.895	0.705	1	1
Bonafide vs a13	0.93	0.965	1	1
Bonafide vs a14	0.96	0.99	1	1
Bonafide vs a15	0.925	0.88	1	1
Bonafide vs a16	0.95	0.665	1	1
Bonafide vs a17	0.985	0.995	1	1
Bonafide vs a18	0.905	0.91	1	1
Bonafide vs a19	0.74	0.955	1	1
Mean a07 - a19	0.9115	0.8908	1	1

Table 8: Summary of all the closed set classification accuracies. For the cells of the table with three different accuracies, these are associated to classifications performed using respectively a series ofsupport vector machines, a series of logistic regressions and a random forest.

Bibliography

[1] H. F. E. AlBadawy, S. Lyu, "Detecting ai-synthesized speech using bispectral analysis," *Computer vision foundation*, 2019.

[2] M. Reynolds, "Courts and lawyers struggle with growing prevalence of deepfakes," *Abajournal*, 2020.

[3] J. Solsman, "Deepfakes' threat to the 2020 us election isn't what you'd think," *c-net*, 2020.

[4] H. Farid, "Detecting digital forgeries using bispectral analysis," *Massachusetts Institute of Technology*, 1999.

[5] D. Tian, "A review on image feature extraction and representation techniques," *International Journal of Multimedia and Ubiquitous Engineering*, 2013.

[6] J. C. T. Jolliffe, "Principal component analysis: a review and recent developments," *Royal Society*, 2016.

[7] A. Hyvärinen, "Survey on independent component analysis," *Neural computing surveys 2*, 1999.

[8] G. H. L. van der Maaten, "Visualizing data using t-sne," *Journal of Machine Learning Reseach 9*, 2008.

[9] V. Nasteski, "An overview of the supervised machine learning methods," *Horizons*, 2017.

[10] J. M. M. Alloghani, D. Al-Jumeily, *A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science*, ch. 1, pp. 3–21. Springer, 2020.

[11] A. B. R. Sutton, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[12] D. M. Hawkins, *Identification of outliers*. Springer, 1980.

[13] S. C. R. Chalapaty, A. Krishna Menon, "Anomaly detection using one class neural networks," *cs.LG*, 2019.

[14] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[15] T. Mitchell, *Machine Learning.* McGraw Hill Higher education, 1997.

[16] L. Breiman, "Random forests," *Machine Learning 45*, 2001.

[17] S. C. R. Chalapathy, A.K. Menon, "Robust, deep and inductive anomaly detection," *Machine Learning and Knowledge Discovery in Databases*, 2017.

[18] T. B. O. Ronneberger, P. Fischer, "U-net: Convolutional networks for biomedical image segmentation," *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015.

[19] P. B. P. Ribeiro Mendes Junior, L. Bondi, "An in-depth study of open-set camera model identification," *Digital Object Identifier*, 2019.

[20] P. M. Hassen, "Learning a neural-network based representation for open set recognition," *Proceedings of the 2020 SIAM International Conference on Data Mining*, 2018.

[21] J. B. I. Cox, M. Miller, *Digital Watermarking.* Morgan Kaufmann, 2001.

[22] F. B. M. Barni, "Watermarking systems engineering: Enabling digital assets security and other applications," *Signal Processing and Communications*, 2004.

[23] J. B. I. Cox, M. Miller, *Digital Water-Marking and Steganography.* Morgan Kaufmann, 2008.

[24] L. A. R. Rivest, A. Shamir, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM vol. 21*, 1978.

[25] P. O. A. Menezes, S. Vanstone, *Handbook of Applied Cryptography.* CRC Press, 1996.

[26] H. Farid, "Image forgery detection," *IEEE Signal Processing Magazine, vol. 26*, 2009.

[27] A. Piva, "An overview on image forensics," *Hindawi Publishing Corporation, ISRN Signal Processing, Volume 2013, Article ID 496701, 22 pages*, 2013.

[28] M. K. L. Van, S. Emmanuel, "Identifying source cell phone using chromatic aberration," *Proceedings of the IEEE International Conference on Multimedia and Expo*, 2007.

[29] K. W. K. Choi, E. Lam, "Automatic source camera identification using the intrinsic lens radial distortion," *Optics Express, vol. 14*, 2006.

[30] N. M. A. Dirik, H. Senear, "Digital single lens reflex camera identification from traces of sensor dust," *IEEE Transactions on Information Forensics and Security, vol. 3*, 2008.

[31] H. F. M. Johnson, "Exposing digital forgeries through chromatic aberration," *Proceedings of the 8th workshop on Multimedia and Security*, 2006.

[32] H. H.-O. I. Yerushalmy, "Digital image forgery detection based on lens and sensor aberration," *International Journal of Computer Vision, vol. 92*, 2011.

[33] M. G. T. Filler, J. Fridrich, "Using sensor pattern noise for camera model identification," *Proceedings of the International Conference on Image Processing*, 2008.

[34] R. d. Q. Z. Fan, "Maximum likelihood estimation of jpeg quantization table in the identification of bitmap compression history," *Proceedings of the International Conference on Image Processing*, 2000.

[35] R. d. Q. Z. Fan, "Identification of bitmap compression history: Jpeg detection and quantizer estimation," *IEEE Transactions on Image Processing, vol. 12*, 2003.

[36] C. K. K. Q. M. S. J. Dittmann, "A context model for microphone forensics and its application in evaluations," *SPIE Conference on Media Watermarking, Security, and Forensics*, 2011.

[37] J. H. M. Kajstura, A. Trawinska, "Application of the electrical network frequency (enf) criterion: A case of a digital recording," *Forensic Science International vol. 155*, 2005.

[38] A. J. Cooper, "The electric network frequency (enf) as an aid to authenticating forensic digital audio recordings an automated approach," *Audio Engineering Society Conference: 33rd International Conference: Audio Forensics-Theory and Practice*, 2008.

[39] H. F. H. Malik, "Audio forensics from acoustic reverberation," *Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing*, 2010.

[40] J. R. Hopgood and P. J. Rayner, "Blind single channel deconvolution using nonstationary signal processing," *IEEE Transactions on Speech and Audio Processing, vol. 11*, 2003.

[41] "Rewind (reverse engineering of audio-visual content data)," 2011.

[42] A. J. Cooper, "Detecting butt-spliced edits in forensic digital audio recordings," in *Proceedings of the 39th International AES Conference Audio Forensics: Practices and Challenges*, 2010.

[43] H. K. N. Nitanda, M. Haseyama, "Audio-cut detection and audio-segment classification using fuzzy c-means clustering," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2004.

[44] I. P. M. Kyperountas, C. Kotropoulos, "Enhanced eigen-audioframes for audiovisual scene change detection," *IEEE Transactions on Multimedia, vol. 9*, 2007.

[45] J. H. R. Yang, Z. Qu, "Detecting digital audio forgeries by checking frame offsets," in *Proceedings of the 10th ACM Workshop on Multimedia and Security*, 2008.

[46] J. A. D. Nicolalde, "Evaluating digital audio authenticity with spectral distances and enf phase change," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2009.

[47] D. Barchiesi and J. Reiss, "Reverse engineering of a mix," *Journal of Audio Engineering Society, vol. 58*, 2010.

[48] J. Y. e. a. X. Wang, "Asvspoof 2019: a large scale public database of synthetic, converted and replayed speech," *Computer Speech and Language 64*, 2020.

[49] T. Dutoit, *An introduction to Text-To-Speech Synthesis*. Springer, 1999.

[50] J. Y. K. tokuda, T. Toda, "Speech synthesis based on hidden markov models," *Proceedings of the IEEE*, 2013.

[51] P. Taylor, *Text-To-Speech Synthesis*. Cambridge University Press, 2009.

[52] H. Z. A. van den Oord, S. Dieleman, "Wavenet: a generative model for raw audio," *arxiv*, 2016.

[53] V. S. A. Mecwan, "Voice conversion algorithm," *International Conference on Advances in Computing, Communication and Control*, 2009.

[54] M. G. H. Silén, E. Helander, "Voice conversion," *International Conference on Advances in Computing, Communication and Control*, 2012.

[55] B. B. T., "Toward robust audio spoofing detection: A detailed comparison of traditional and learned features," *Digital Object Identifier 10*, 2019.

[56] F. D. A. Lieto, D. Moro, ""hello, who am i talking to?" a shallow cnn approach for human vs. bot speech classification," *IEEE*, 2019.

[57] A. G. F. Pedregosa, G. Varoquaux, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research 12*, 2011.

[58] K. O. M. Morise, F. Yokomori, "World: A vocoder-based high-quality speech synthesis system for real-time applications," *IEICE Transactions on Information and Systems*, 2016.

[59] Y. Agiomyrgiannakis, "Vocaine the vocoder and applications in speech synthesis," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[60] V. M. A. Sharma, P. Kumar, "Fast griffin lim based waveform generation strategy for text-to-speech synthesis," *Multimedia Tools and Applications Journal*, 2020.

[61] H. Kawahara, "Straight, exploitation of the other aspect of vocoder: Perceptually isomorphic decomposition of speech sounds," *Acoustical Science and Technology*, 2006.

[62] J. C. M. Abadi, P. Barham, *TensorFlow: A system for large-scale machine learning*, ch. 12, pp. 265–283. USENIX Association, 2016.

[63] N. Ketkar, *Introduction to Keras*, ch. 4, pp. 95–109. Apress, 2017.