

# **Privacy-Preserving Speech Emotion Recognition**

**Miguel Ângelo Rodrigues Dias**

Thesis to obtain the Master of Science Degree in  
**Electrical and Computer Engineering**

Supervisors: Prof. Isabel Maria Martins Trancoso  
Prof. Alberto Abad Gareta

## **Examination Committee**

Chairperson: Prof. João Fernando Cardoso Silva Sequeira  
Supervisor: Prof. Isabel Maria Martins Trancoso  
Members of the Committee: Prof. Carlos Nuno da Cruz Ribeiro

**November 2017**



# Acknowledgments

Foremost, I would like to thank my thesis supervisor, Prof. Isabel Trancoso, not only for her constant support and invaluable scientific insights throughout the fulfillment of this thesis, but also for the trust and freedom she bestowed upon me in difficult times. I would also like to thank my thesis co-supervisor, Prof. Alberto Abad, for his invaluable suggestions regarding this thesis topics, and for helping me solve implementation issues along the way. To them, I am more than grateful.

A special thanks to Prof. Bhiksha Raj of Carnegie Mellon University for the discussion we had regarding the motivation of this thesis, which was extremely helpful, and to Abelino Jiménez for proof reading some of my earlier work.

I am grateful to the Cryptography Research Group of Microsoft Research, specially Kim Laine and Ran Gilad-Bachrach, for their availability and for their advice on how to use their fully homomorphic encryption library.

I would also like to express my gratitude to my friend Francisco Teixeira, for our discussions regarding this thesis and for proof reading most of my work.

I would like to express my heartfelt appreciation to my friends that always believed in me and for helping to relieve myself of stress by forcing me to take breaks.

Finally, I am deeply grateful to family for their unyielding support and encouragement throughout my whole academic journey. A special thanks to my father for his valuable life lessons and to my mother for caring too much. The same goes to both my sisters for always having my back.

To each and every one of you – Thank you.



# Abstract

Machine learning classification and data mining tasks are used in numerous situations nowadays, for instance, quality control, eHealth, banking, and homeland security. Due to lack of large training sets and machine learning tools, it is often effective to outsource the inferring of datasets to foreign parties that hold accurate predictive models (e.g. cloud-based outsourcing). This outsourcing, however, raises major privacy concerns. The situation becomes even more dire when datasets are composed of irrevocable biometric data, implying the need for data to remain confidential along the whole testing process.

In this thesis, we base ourselves mainly on the employment of privacy-preserving schemes in a two-class speech emotion recognition task, as a proof of concept that could be extended to other speech analytics tasks. Our aim is to prove that the implementation of privacy-preserving speech mining schemes in challenging tasks involving paralinguistic features are not only feasible, but also efficient and accurate. In a first approach we use distance-preserving hashing techniques in a support vector machine. Afterwards, in a second approach, a fully homomorphic encryption scheme is employed in a light neural network to preserve the privacy of the recording. For each approach, small, but crucial modifications to the baseline model were applied, allowing for an efficient protection of sensitive data, in both training and inferring stages, with little to no degradation regarding the accuracy of state-of-the-art predicative models.

## Keywords

Data Privacy; Cryptography; Speech Mining; Emotion Recognition; Paralinguistics.



# Resumo

Uma grande variedade de tarefas de aprendizagem automática e de prospeção de dados são, hoje em dia, usadas em várias circunstâncias, como por exemplo, em controlo de qualidade, eHealth, comércio bancário e segurança nacional. Devido à falta de grandes conjuntos de treino e de outras ferramentas é, normalmente, eficiente recorrer a outros partidos, com modelos preditivos precisos, para o teste de certos conjuntos de dados. Todavia, este método levanta sérias questões de privacidade. A situação torna-se ainda mais grave quando os conjuntos são compostos por dados irrevogáveis, como dados biométricos. É devido a este facto que é extremamente necessário que os dados em questão permaneçam confidenciais ao longo do seu teste no modelo preditivo.

Nesta dissertação, baseámo-nos maioritariamente no desenvolvimento de esquemas de preservação de privacidade numa tarefa de reconhecimento de emoções com duas classes, como prova de conceito que poderá ser estendida a outras tarefas de análise de fala. O nosso objetivo é provar que a implementação de esquemas de preservação de privacidade de prospeção de voz em difíceis tarefas paralinguísticas não só é factível, mas também eficiente e precisa. Numa primeira abordagem, foram usadas técnicas de preservação de distâncias entre *hashes* numa máquina de suporte de vetores. Seguidamente, numa segunda abordagem, foi utilizado um esquema de encriptação homomórfica total numa pequena rede neuronal, de maneira a eficientemente preservar a confidencialidade da gravação. A implementação destes dois esquemas promove uma proteção eficiente de dados sensíveis e pouca degradação em termos de precisão do modelo.

## Palavras Chave

Privacidade de Dados; Criptografia; Mineração de Fala; Reconhecimento de Emoções; Paralinguística.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Speech Mining . . . . .	3
1.2	Security and Privacy . . . . .	3
1.3	Motivation . . . . .	4
1.4	Structure of the Document . . . . .	6
<b>2</b>	<b>State of the Art</b>	<b>7</b>
2.1	Speech and Data Mining . . . . .	9
2.1.1	Hand-Engineered Feature Extraction . . . . .	9
2.1.1.1	Spectral Features . . . . .	10
2.1.1.2	Prosodic Features . . . . .	11
2.1.2	Automatic Feature Extraction . . . . .	11
2.1.3	Classification . . . . .	12
2.1.3.1	Gaussian Mixture Models . . . . .	13
2.1.3.2	Support Vector Machines . . . . .	14
2.1.3.3	Neural Networks . . . . .	15
2.2	Privacy-Preserving Computation . . . . .	17
2.2.1	Secure Multi-Party Computation . . . . .	17
2.2.1.1	Homomorphic Encryption . . . . .	18
2.2.1.2	Zero Knowledge Proof . . . . .	19
2.2.1.3	Oblivious Transfer . . . . .	20
2.2.1.4	Garbled Circuits . . . . .	21
2.2.2	Differential Privacy . . . . .	22
2.2.3	Nearest-Neighbour with Randomized Embedding . . . . .	23
2.2.3.1	Locality-Sensitive Hashing . . . . .	23
2.2.3.2	Secure Binary Embedding . . . . .	24
<b>3</b>	<b>Privacy-Preserving Emotion Classifier Using Distance-Preserving Hashing</b>	<b>29</b>
3.1	Introduction . . . . .	31
3.2	Baseline of Emotional Classifier using SVMs . . . . .	31
3.2.1	Feature Extraction and Classification . . . . .	32

3.2.2	Experimental Setup . . . . .	32
3.2.3	Discussion . . . . .	33
3.3	Emotion Classification Using SBE . . . . .	34
3.3.1	Secure Binary Embedding . . . . .	34
3.3.2	Experimental Setup . . . . .	35
3.3.3	Results . . . . .	35
3.3.3.1	Distance . . . . .	35
3.3.3.2	Training and Classification . . . . .	37
3.3.4	Discussion . . . . .	38
3.4	Emotion Classification Using SMH . . . . .	39
3.4.1	Secure Modular Hashing . . . . .	39
3.4.2	Experimental Setup . . . . .	39
3.4.3	Results . . . . .	40
3.4.3.1	Probability . . . . .	40
3.4.3.2	Distance . . . . .	41
3.4.3.3	Leakage . . . . .	42
3.4.3.4	Training and Classification . . . . .	43
3.4.4	Discussion . . . . .	43
3.5	Privacy Analysis . . . . .	44
3.6	Case Example . . . . .	44
3.7	Summary . . . . .	47
<b>4</b>	<b>Privacy-Preserving Emotion Classifier Using Fully Homomorphic Encryption</b>	<b>49</b>
4.1	Introduction . . . . .	51
4.2	Baseline of Emotion Classifier Using ANNs . . . . .	51
4.2.1	Experimental Setup . . . . .	52
4.2.2	Training and Classification . . . . .	52
4.2.3	Discussion . . . . .	53
4.3	Emotion Classification with Cryptonets . . . . .	54
4.3.1	Cryptographic Neural Networks . . . . .	54
4.3.2	Simple Encrypted Arithmetic Library . . . . .	54
4.3.3	Approximations and Modifications . . . . .	56
4.3.3.1	Polynomial Activation Function . . . . .	56
4.3.3.2	Pre-Activation Normalization . . . . .	56
4.3.3.3	Encryption Noise . . . . .	59
4.3.4	Experimental Setup . . . . .	60
4.3.5	Results . . . . .	61
4.3.6	Discussion . . . . .	62

4.4	Privacy Analysis . . . . .	64
4.5	Case Example . . . . .	64
4.6	Summary . . . . .	66
<b>5</b>	<b>Conclusions and Future Work</b>	<b>67</b>
5.1	Conclusions . . . . .	69
5.2	Contributions . . . . .	70
5.3	Future Work . . . . .	71
<b>A</b>	<b>Corpora</b>	<b>79</b>
A.1	Let's Go / LEGO . . . . .	79
<b>B</b>	<b>End-to-end Emotion Classifier</b>	<b>83</b>
B.1	Introduction . . . . .	83
B.2	Deep Convolutional Recurrent Network . . . . .	84
B.2.1	Experimental Setup . . . . .	84
B.2.2	Results and Discussion . . . . .	86
<b>C</b>	<b>SEAL Library</b>	<b>89</b>
C.1	Introduction . . . . .	89
C.2	Encryption Parameters . . . . .	90
C.3	Encoding/Decoding . . . . .	91
C.4	Relinearization . . . . .	92
C.5	Noise . . . . .	92



# List of Figures

1.1	An example of Secure Machine Learning as a Service applied to speech signals. A user sends encrypted speech data to a cloud holding a predicative model. The cloud takes the encrypted speech data and evaluates it. The provided outputs are then sent to the user. .	5
2.1	Speech Mining Basic Scheme with $N$ Features. . . . .	9
2.2	Spectral Feature Extraction Scheme . . . . .	10
2.3	Training and classification phases of a traditional machine learning model. . . . .	12
2.4	Two distinct examples for the use of Gaussian Mixture Models in different dimensions. . .	14
2.5	Mapping features to a space where the data is linearly separable, using the kernel function $\phi$ . . . . .	15
2.6	Sample architecture of a deep neural network with several hidden layers and units. . . . .	17
2.7	Abstract example of a Zero Knowledge Proof (ZKP) with a <i>prover</i> , Peggy, and a <i>verifier</i> , Victor. . . . .	20
2.8	2D Example using an Locality-Sensitive Hashing (LSH) projection. . . . .	24
2.9	Representation of the Universal Scalar Quantizer. This non-monotonic quantization function $Q(\cdot)$ allows for universal rate-efficient scalar quantization and provides information-theoretic security [1]. . . . .	25
2.10	2D example using an Secure Binary Embedding (SBE) projection. . . . .	26
2.11	Behaviour of SBE using different parameters [1]. . . . .	26
3.1	Euclidean distance between each training and validation set features for different values of bits per coefficient (bpc) and $\Delta$ . . . . .	36
3.2	Cosine distance between each training and validation set features for different values of bpc and $\Delta$ . . . . .	36
3.3	SMH example using two hashes with $k = 3$ [2]. . . . .	40
3.4	Comparison between theoretical and estimated probability $P(h(\mathbf{x}) = h(\mathbf{y}))$ , and euclidean distance for different values of $k$ , $mpc = 64$ and $\Delta = 3$ . Both derived from the hashed Let's Go datasets used in all other previous experiments. . . . .	41

3.5	Comparison between expected and actual values of the hamming distance, and euclidean distance for different values of $k$ , $m_{pc} = 64$ and $\Delta = 3$ . Both derived from the hashed Let's Go datasets used in all other previous experiments. . . . .	42
3.6	Protocol of a Privacy-Preserving Speaker Verification using an SBE approach. A user sends in a recording with his/her voice and receives a positive or negative response for the server. . . . .	45
3.7	Protocol of a Multiparty Privacy-Preserving Speech Emotion Recognition using SBE/Secure Modular Hashing (SMH) approach for training model improvement. Secret Keys $\mathbf{A}$ and $\mathbf{w}$ are shared between all data providers. The trusted party receives only the hashed data and trains a joint model. . . . .	46
4.1	Multilayer Perceptron architecture used in the baseline experiments. . . . .	53
4.2	SEALs encoding and encryption process. . . . .	55
4.3	Illustration of the polynomial approximations to the Rectified Linear Unit (ReLU) using the analytical functions from Table 4.2. . . . .	57
4.4	Modified Multilayer Perceptron (MLP) architecture compliant with an Fully Homomorphic Encryption (FHE) scheme. . . . .	60
4.5	Case example of a client using a cloud-based application to predict the emotion label of a given recording. . . . .	65
B.1	Evolution of the evaluation results on the validation set against the number of training steps.	86

# List of Tables

3.1	Results of the tests performed on the different sets using different features. The features show which features were used to train and classify the sets. Correct/Total Instances represents the number of correct predictions within the total amount of recordings in the set. Finally, accuracy represents the accuracy classification for each set. . . . .	33
3.2	Accuracy results for different values of bpc and leakage. Cells presented as a crossed line represent the lack of information leaked necessary for an accurate classifier. The most suitable values that allow for a rather accurate and private classifier are highlighted in bold. . . . .	38
3.3	SMH privacy threshold values for each $x \bmod k$ , empirically set from plots in Figure 3.5.	42
3.4	Leakage results for different parameters of $\Delta$ and $k$ , with $mpc = 32$ . . . . .	42
3.5	Accuracy results for different parameters of $\Delta$ and $k$ , with $mpc = 32$ . . . . .	43
4.1	Test results of the MLP baseline for both validation and test set. . . . .	53
4.2	Approximation of the ReLU function by polynomials. $p$ represents the polynomial degree of the approximation and the approximated polynomial is the approximation of the ReLU by a polynomial [3]. . . . .	56
4.3	Cryptonets Encoding and Encryption Parameters. The parameter name refers to the name of the parameter as it appears in the Simple Encrypted Arithmetic Library (SEAL) library while the value represents the actual numerical value chosen for the experiments. Parameters that are not mentioned, but are still crucial to the experiments, are left as default. . . . .	60
4.4	Accuracy results of the tests performed on the unencrypted test set for different values of $p$ , with and without normalization. The normalization refers to the pre-activation normalization, serving to adjust the inputs to a more viable interval. The $p$ represents the degree of the polynomial approximation of the ReLU activation function. . . . .	62
4.5	Accuracy results of the tests performed on the encrypted test set for different values of $p$ , with and without normalization. The normalization refers to the pre-activation normalization, serving to adjust the inputs to a more viable interval. The $p$ represents the degree of the polynomial approximation of the ReLU activation function. . . . .	62

4.6	Default pairs $(n, q)$ and their estimated security levels [4]. . . . .	64
A.1	Statistics of the two corpora LEGO and LEGOext and of the combined corpus LEGOv2 and the actual used corpus in this thesis (LEGOv2.1). Shown are the reference, the total number of calls, the number of calls with emotional labels, the total number of exchanges , the number of exchanges with emotional labels, the average dialogue length in number of exchanges and the inter-rater agreement (Adapted [5]). . . . .	81
C.1	Default pairs $(n, q)$ and their estimated security levels [4]. . . . .	90
C.2	Noise estimates for homomorphic operations in SEAL [4]. . . . .	93

# Acronyms

<b>ANN</b>	Artificial Neural Network
<b>bpc</b>	bits per coefficient
<b>CNN</b>	Convolutional Neural Network
<b>DCRN</b>	Deep Convolutional Recurrent Network
<b>DMFCC</b>	Delta Mel-Frequency Cepstral Coefficient
<b>DNN</b>	Deep Neural Network
<b>FHE</b>	Fully Homomorphic Encryption
<b>FV</b>	Fan-Vercauteren
<b>GC</b>	Garbled Circuit
<b>GMM</b>	Gaussian Mixture Model
<b>HE</b>	Homomorphic Encryption
<b>HMM</b>	Hidden Markov Model
<b>IQ</b>	Interaction Quality
<b>kNN</b>	K-Nearest Neighbours
<b>LPC</b>	Linear Prediction Coefficient
<b>LPCC</b>	Linear Prediction Cepstrum Coefficient
<b>LPCMCC</b>	Linear Prediction Coefficients and Mel Cepstrum Coefficient
<b>LSH</b>	Locality-Sensitive Hashing
<b>LSTM</b>	Long Short-Term Memory
<b>MFCC</b>	Mel-Frequency Cepstral Coefficient

<b>MLaaS</b>	Machine Learning as a Service
<b>MLP</b>	Multilayer Perceptron
<b>mpc</b>	measurements per coefficient
<b>OT</b>	Oblivious Transfer
<b>PLP</b>	Perceptive Linear Predictive
<b>PPDM</b>	Privacy-Preserving Data Mining
<b>PPSM</b>	Privacy-Preserving Speech Mining
<b>RBF</b>	Radial Basis Function
<b>ReLU</b>	Rectified Linear Unit
<b>RLWE</b>	Ring Learning with Errors
<b>RNN</b>	Recurrent Neural Network
<b>SBE</b>	Secure Binary Embedding
<b>SEAL</b>	Simple Encrypted Arithmetic Library
<b>SMH</b>	Secure Modular Hashing
<b>SMLaaS</b>	Secure Machine Learning as a Service
<b>SJL</b>	Secure Joint Learning
<b>SMPC</b>	Secure Multi-Party Computation
<b>STPC</b>	Secure Two-Party Computation
<b>SVM</b>	Support Vector Machine
<b>TOP</b>	Trusted Outsourcing Party
<b>UBM</b>	Universal Background Model
<b>USQ</b>	Universal Scalar Quantization
<b>ZKP</b>	Zero Knowledge Proof

# 1

## Introduction

### Contents

---

1.1	Speech Mining . . . . .	3
1.2	Security and Privacy . . . . .	3
1.3	Motivation . . . . .	4
1.4	Structure of the Document . . . . .	6

---



## 1.1 Speech Mining

A person's voice is a biometric trait which conveys information about his/her anatomical and behavioural traits. Along with other biometrics, such as fingerprints, face, iris and even handwritten signatures, a variety of data mining tasks can be performed on a speech signal. These tasks mainly include identity verification, nativity recognition, emotional and health state recognition and also anatomical traits such as age, weight and height.

Speech mining is a particular field of data mining that aims to extract relevant information from speech. In fact, due to modern advances in communication technology, hours and hours of audio data have been made available, making speech mining an area of great interest to researchers. From speech alone, the average individual can distinguish human traits like gender, age group, accent and emotion from acoustic/prosodic features. For example, while talking on the phone, one can distinguish the gender of the person based on the pitch and other characteristics of the voice. The developing technologies and knowledge about this subject allows the transition from this human-based speech mining to an automatic and machine-based speech mining [6].

The traditional process for performing speech mining is pretty straightforward. First, temporal and spectral features, such as prosodic features, Mel-Frequency Cepstral Coefficients (MFCCs), Perceptive Linear Predictives (PLPs), Linear Prediction Cepstrum Coefficients (LPCCs) and Linear Prediction Coefficients and Mel Cepstrum Coefficients (LPCMCCs) need to be extracted from the speech signal. Normally, MFCC features are used due to their high performance in audio classification. After obtaining these features a great variety of machine learning techniques can be used to perform speech mining. These techniques include the widely used Hidden Markov Models (HMMs)/Gaussian Mixture Models (GMMs), Support Vector Machines (SVMs), deep learning or even end-to-end learning approaches, which both eliminates the need for engineered feature extraction and outperforms the traditional approaches, thus becoming the state-of-the-art. Even though GMMs continue to be the standard model for speech recognition [7], Deep Neural Networks (DNNs) with many hidden layers, that are trained using new methods, have been shown to outperform GMMs on a variety of speech recognition benchmarks, sometimes by a large margin [8], making DNNs suitable substitutes for GMMs in some cases.

## 1.2 Security and Privacy

The increasing use of biometrics in data mining tasks and the permanent link the user has with them has raised concerns about their privacy and security. Besides this immutability with the user, biometrics are also public and irrevocable, which means they are basically accessible to anyone and cannot be replaced. A good example of a biometric security breach is the disclose, either by leaking or hacking, of biometric templates from a database resulting in a permanent loss of security. Furthermore, attackers will be able to break through a security system using spoofing/mimicry and other methods [9], which

may lead to issues like identity theft and fraud. Attacks to biometric databases have already been known to happen. A good example is the hack against the U.S. Office of Personnel Management [10] compromising millions of fingerprints.

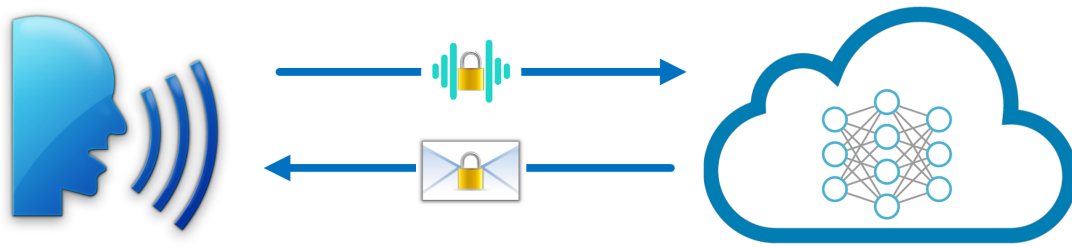
With the vast datasets available in modern times, outsourcing the inference of test sets to parties that hold very accurate models is becoming more and more common. Usually, it is more expensive and time-consuming to design, implement and test a whole new machine learning algorithm from scratch than just paying some party to test the data for us. However, there are no guarantees that the outsourcing party is trustworthy. It may try to uncover valuable and sensitive information about the data sent. Therefore, it is not enough to protect the biometric in a database. It is of the utmost importance to also protect it along the whole testing process.

Although there are several implementations in the literature about privacy-preserving machine learning, the literature regarding privacy-preserving machine learning in speech signals is very scarce. The situation for privacy-preserving paralinguistic speech tasks is even less mentioned in the literature. Nonetheless, several privacy-preserving schemes can be implemented using cryptographic primitives such as, homomorphic encryption, garbled circuits, oblivious transfer, among others. A Secure Multi-Party Computation (SMPC) may use any of these primitives to construct of a fully secure privacy-preserving protocol. Other techniques based on nearest-neighbour search (SBE or SMH) and on differential privacy may also be used for the construction of said protocols.

## 1.3 Motivation

In today's world, privacy is a very important matter. As technology advances, people's privacy tends to decrease. People either share information willingly (e.g. Facebook or other social media) or unwillingly (e.g. biometrics), which may lead to major security issues. Biometrics are very reliable for authentication and data analysis, but also very dangerous since they are permanently linked to the user [9]. Hence, a compromised template cannot be replaced, resulting in a permanent loss of security. Template privacy is very useful in encrypted matching, for example, in speaker verification, however, there are still major privacy issues when using biometric data for prediction models.

This thesis will focus mainly on the classification of emotions [11, 12], specifically frustration. To begin with, an individual might not want to share his emotional status and other paralinguistic features. Furthermore, as it was referred in Section 1.1, the voice of a person conveys a great deal of information (paralinguistic and non-linguistic), and an attack to a voice biometric might be problematic. The attacker can perform his/her own speech mining tasks to uncover relevant information about the individual or what he actually said, which can later be used for a variety of crimes such as fraud, identity theft and blackmail. However, if the speech is made private and the tasks at hand are computed in a domain where it is simply impossible to have access to the true biometric template, most of these privacy issues are resolved.



**Figure 1.1:** An example of Secure Machine Learning as a Service applied to speech signals. A user sends encrypted speech data to a cloud holding a predicative model. The cloud takes the encrypted speech data and evaluates it. The provided outputs are then sent to the user.

Nowadays, with the development of cloud services and machine learning, Machine Learning as a Service (MLaaS) is getting more and more recognition. Still, sending someone's biometric data to a cloud so it can be "mined" is extremely dangerous. There is usually no assurance that the cloud is trustworthy, and thus, something needs to be done regarding the protection of the biometric traits present in a recording. Nonetheless, this problem is solvable with the implementation of privacy-preserving machine learning schemes. Unlike speaker verification, which uses matching with an encrypted database, privacy-preserving paralinguistic mining aims to train and test a model using confidential paralinguistic features, which is not a trivial process. The scheme must allow computations, made in the encrypted domain, to be preserved in the non-encrypted domain, in order for the machine learning algorithm to work, or at least, for some information about the true features to be leaked while still maintaining privacy. Figure 1.1 illustrates a simple protocol for Secure Machine Learning as a Service (SMLaaS) applied to speech signals.

Privacy-preserving emotion recognition can be branched out and adjusted to fulfill other speech analytics tasks in a wide variety of fields. These include, but are not limited to, eHealth, banking and homeland security. However, most of the current privacy-preserving schemes available are not mature enough in terms of performance and efficiency for a large scale deployment [9] and, due to this very fact, a lot of research still has to be done about this subject. In fact, for most schemes, there is a trade-off relationship between utility, efficiency and privacy. Therefore, nowadays, to have a good private scheme, either utility or efficiency has to be sacrificed.

To sum up, by performing privacy-preserving emotion recognition we aim to fulfill the following statements:

- Perform emotion recognition in an encrypted domain while obtaining little to no degradation in the classification results.
- Effectively secure sensitive information conveyed by the user's voice.
- Derive the best compromise of the trade-off relationship.
- Broaden the spectrum in which privacy-preserving schemes can be applied.

- Contribute to advancements in the field of SMLaaS.

There is a vast amount of literature related to privacy-preserving and biometric protection schemes. However, most of the literature points to biometric authentication and verification using other types of biometrics, such as fingerprints, face or iris. The lack of literature regarding private data mining tasks in speech signals using paralinguistic features, calls for an increase in research about this very topic. Moreover, this thesis can be seen as an extension of J. Portêlo's work [13] where keyword spotting, music matching and speaker recognition tasks are thoroughly investigated in a privacy-preserving domain.

## 1.4 Structure of the Document

This thesis is organized as follows: Chapter 1 is composed by a brief introduction to the topic of speech mining and privacy-preserving computation, as well as the main motivations of this thesis. The most recent and up to date work, as well as other state-of-the-art techniques regarding speech mining and privacy-preserving schemes, are thoroughly detailed in Chapter 2. In Chapter 3, experiments using Support Machine Vectors with Secure Binary Embeddings and Secure Modular Hashing are performed and analyzed on an emotional database, while in Chapter 4, experiments using Artificial Neural Networks with Fully Homomorphic Encryption are performed and analyzed, on the same emotional database. Finally, Chapter 5 drafts the most important conclusions taken over the course of this thesis. It also refers the main contributions of this thesis and some interesting topics for future work.

# 2

## State of the Art

### Contents

---

2.1 Speech and Data Mining . . . . .	9
2.2 Privacy-Preserving Computation . . . . .	17

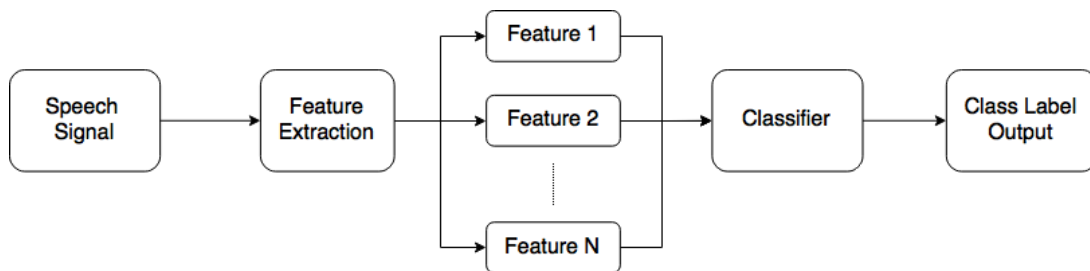
---



This chapter describes the most up to date literature about the subjects required to fulfill this thesis. Section 2.1 describes the techniques that can be used to perform data mining in speech signals. Methods and techniques on how to preserve the privacy while performing certain tasks among one or more untrustworthy parties are presented in Section 2.2.

## 2.1 Speech and Data Mining

Data Mining is the process of scraping data in order to locate latent patterns that might convey valuable information. This process can be applied to the majority of signals and, most importantly for this thesis, speech signals, which are processed in a way that features can be extracted. This feature extraction can either be performed by human-based pre-processing or in a more end-to-end fashion, through an automatic machine learning feature extraction phase [14]. In human feature pre-processing, different types of features, mostly prosodic or spectral, may be used by a specific classifier to classify new data. This classifier, usually based on machine learning algorithms, is previously trained with a training set containing a significant amount of data with the respective class labels. The accuracy of a classifier is proportional to the amount of training it was subjected to, i.e., it increases with the amount of training performed. Figure 2.1 displays a basic speech mining scheme with a trained classifier.

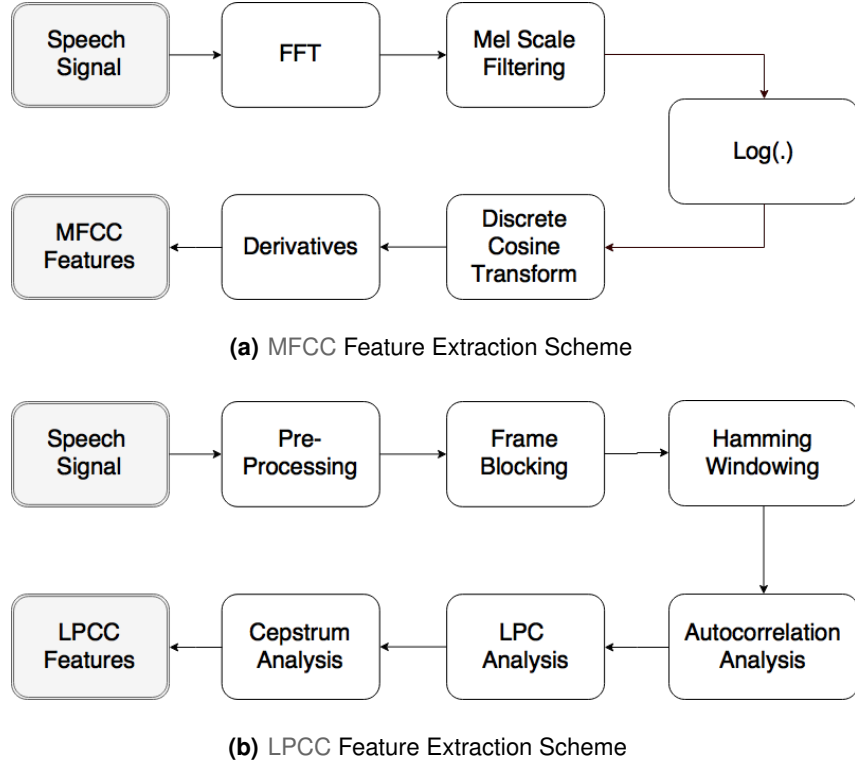


**Figure 2.1:** Speech Mining Basic Scheme with  $N$  Features.

### 2.1.1 Hand-Engineered Feature Extraction

A classification task is as good as the features that are used to execute it. Therefore, choosing which features to use is a very important step to acquire good classification results. In order for features to be adequate for speech classification tasks, they should have the following attributes:

- *Informative:* Features should be chosen in a way that relevant information may be retrieved and irrelevant information discarded.
- *Practical:* The extraction and measurement of features should not be too computationally demanding and features should occur naturally and frequently in a speech signal.
- *Robust:* They should be constant over time and not prone to relevant changes provided by outside factors (e.g. noise).



**Figure 2.2:** Spectral Feature Extraction Scheme

Broadly, hand-engineered feature extraction techniques are classified as temporal analysis and spectral analysis. In temporal analysis, the speech waveform itself is used for analysis while in spectral analysis, a spectral representation of the speech signal is used for analysis. A great deal of state-of-the-art tools, such as HTK, openSMILE, GeMAPS and PRAAT are available for performing feature extraction in speech signals.

### 2.1.1.1 Spectral Features

Spectral features such as, MFCCs and LPCCs, are shown to have great performance in most speech mining tasks [11, 15, 16]. On one hand, the Mel-cepstra takes short-time spectral shape to retrieve important data about the quality of voice and production effects [11]. The reason for computing the short-term spectrum is that the cochlea of the human ear performs a quasi-frequency analysis. The analysis in the cochlea takes place on a non-linear frequency scale (known as the Bark scale or the Mel scale) [17]. In fact, the first 13 MFCCs have been shown to yield a very good performance, serving as the standard for the feature extraction stage [18]. On the other hand, the basic idea behind linear prediction is that the current sample can be predicted, or approximated, as a linear combination of the previous samples, which would provide a more robust feature against sudden changes [18]. Figure 2.2 explains schematically how to obtain the most important spectral features from speech signals.

Although MFCCs and LPCCs are used in a bigger scale, other spectral features such as Linear

Prediction Coefficients (LPCs), PLPs, Delta Mel-Frequency Cepstral Coefficients (DMFCCs), and even hybrids (e.g. LPCMCCs [19]) may also be used to train and test classifiers. PLPs analysis of speech is quite similar to the Mel analysis where the short-term spectrum is modified. Still, MFCCs are based on a Mel scale while PLPs are based on the Bark scale. On another note, PLPs have been known to outperform MFCCs in specific conditions [17]. Using each feature type on their own, to train/test classifiers, is not the best approach since it will lead to poor results. However a joint use of these spectral features, as well as with other prosodic and time-domain features, leads to better classification results [19]. Despite all the advances in this scientific area, the MFCCs are still the state-of-the-art when it comes to hand-engineered features.

#### **2.1.1.2 Prosodic Features**

The main prosodic features are the Fundamental Frequency (F0), Energy and Duration. These might be useful for some speech (paralinguistic) mining tasks, such as, emotional classification [19], Parkinson's disease detection [16] and foreign accent detection [20]. In emotion classification, the use of prosody is quite important since each human emotion can be characterized with different variations in pitch, energy and speaking rate. Nonetheless, for emotions like anger and surprise, only using prosodic cues becomes an issue since both have high pitch and energy [15]. A way around it may be the joint use of both prosodic and spectral features. Parkinson's disease affects all components of speech production, thus, the speaker may exhibit a monotonous pitch, inappropriate pauses, variable speech rates, harsh voice, unusual shimmers (variation in amplitude) and jitters (variation in period), among others [16].

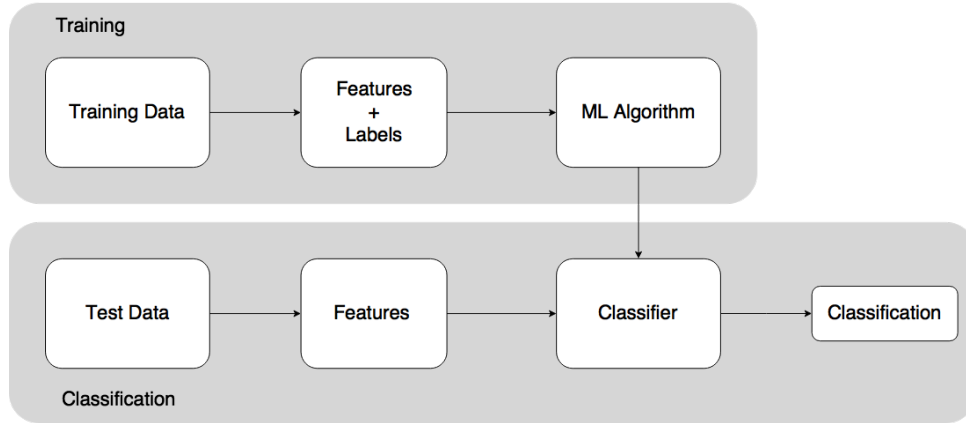
On their own, prosodic features still present a lack in classification performance when compared to spectral features. Although, the use of both features (spectral and prosodic) actually boosts the performance of the classifiers when compared with using each feature type individually [19].

There are a lot of techniques to extract prosodic features in the literature [19,21], specifically, short-term correlation, maximum likelihood and frequency domain techniques.

### **2.1.2 Automatic Feature Extraction**

Even though hand-engineered features are easier to process and implement in state-of-the-art machine learning approaches, and actually hold rather respectable accuracy scores, they are still human-based, which means they are always prone to slight human-based errors. Errors that, in most state-of-the-art machine learning models, are propagated, resulting in a significant loss of accuracy. Due to this fact, new end-to-end approaches [14] are being thoroughly researched to both provide an automatic machine-based feature learning and to decrease the client's role in a speech mining algorithm, i.e., to, for instance, only needing the user to speak into a microphone, forcing the inputs of the model to be raw speech signals which then eliminated the need for a pre-processing of the speech by the user.

In recent related works, experiments were made in large-vocabulary speech recognition (LVCSR) [22],



**Figure 2.3:** Training and classification phases of a traditional machine learning model.

based on log-Mel filterbank energies by using a deep convolutional recurrent neural network, which uses both convolutional layers and a specific type of recurrent layers, Long Short-Term Memorys (LSTMs). The convolutional layer effectively helps reducing the temporal variation, and another convolutional layer allows to preserve locality and reduce frequency variation. The LSTM layers serve for a contextual modelling of the speech signal. Moreover, the features learned between the first two convolutional layers appear to model phone-specific spectral envelopes of the sub-segmental speech signal, which makes the model more resilient to noise. Other works using a partial end-to-end feature extraction along with a low-dimensional Mel filterbank feature vector were performed in the field of paralinguistics [23, 24]. Finally a full end-to-end learning using a Deep Convolutional Recurrent Network (DCRN) was developed for an emotion recognition task [14].

Instead of relying on previously human-engineered features, features are learned along the deep neural network, being gradually updated to be best suited for the task at hand. In fact, end-to-end learning derives a representation of the input signal directly from the raw data, which will, ultimately, allow the network to learn an intermediate representation of the raw input signal that better suits the task, leading to better performances and accuracy scores [14].

Automatic feature extraction not only allows for better performances and accuracy scores, but also, minimizes the amount of signal processing the user has to perform. This will be very helpful in devising privacy-preserving machine learning as a service protocols, namely one referring to speech signals, where the client will only need to speak into a microphone in order to perform predictions. In these privacy-preserving protocols, the only processing done by the user, if automatic feature extraction is present, is the encryption of his or her speech.

### 2.1.3 Classification

Although choosing which features to extract for each task is a very important step towards good performance, a good classification will always depend on the classifier used. Classifiers have to be previously trained by, typically, a machine learning algorithm. This algorithm will collectively sweep

through the training data, labelled with the respective classes, and create a prediction/classification model. A scheme with the usual process of training and testing can be inspected in Figure 2.3.

In speech mining, GMM [25, 26] (or a mix between GMM and Universal Background Model (UBM) (GMM-UBM) [7, 13]) and SVM [13, 15, 19] based classifiers have been extensively used by the speech processing community due to their good performance and easy implementation. However, in some recent works [8, 27], a DNN implementation has shown better performances for speech mining tasks than a GMM implementation, due to its ability to train better with a limited size training set [27]. Moreover, end-to-end approaches [14] were also proposed by combining recurrent neural networks with LSTM networks, outperforming most traditional approaches for speech and emotion recognition. Still, the GMM is the most common model in speech pattern recognition problems [13]. Other classification algorithms, such as K-Nearest Neighbours (kNN) [28], MLPs, and Maximum Likelihood Bayesian Classifiers, have also been recently used by researchers [19]. A great deal of software and tools is available for the public to perform speech modelling, such as SPEAR <sup>1</sup> and MATLAB (GMMs) and LIBSVM <sup>2</sup> and Weka <sup>3</sup> (SVMs).

### 2.1.3.1 Gaussian Mixture Models

GMMs are probabilistic models that represent single normally distributed subgroups of data within an overall distribution model. This technique falls within the unsupervised learning category of machine learning algorithms, since the subgroup assignment is unknown. The expectation maximization algorithm is the traditional technique used to estimate the mixture model's parameters. This algorithm is basically a numerical technique for maximum likelihood estimation for updating the model parameters, making the gaussian distributions fit the data in question. Two simple examples<sup>4</sup> for the use of GMMs are illustrated in Figure 2.4.

The introduction of the expectation maximization algorithm made possible the training of HMMs. With it, it became possible to develop speech recognition tasks with the richness of the GMM to represent the relationship between HMM states and the acoustic input. These systems usually use MFCCs and PLPs and are designed to discard irrelevant information in waveforms while discriminating relevant information [8, 13].

GMMs have the advantage of being able to, with enough components, model probability distributions to any required level of accuracy and it is fairly easy to fit data when using the expectation maximization algorithm. Even though there are a lot of ways to improve this system, GMMs are very successful for acoustic modelling and it is very challenging to come up with a new model that can outperform such an efficient and simple model [8, 13] and so, for the majority of tasks, it remains the state-of-the-art in speech recognition [7]. However, an alternative using DNNs was shown to outperform common GMMs

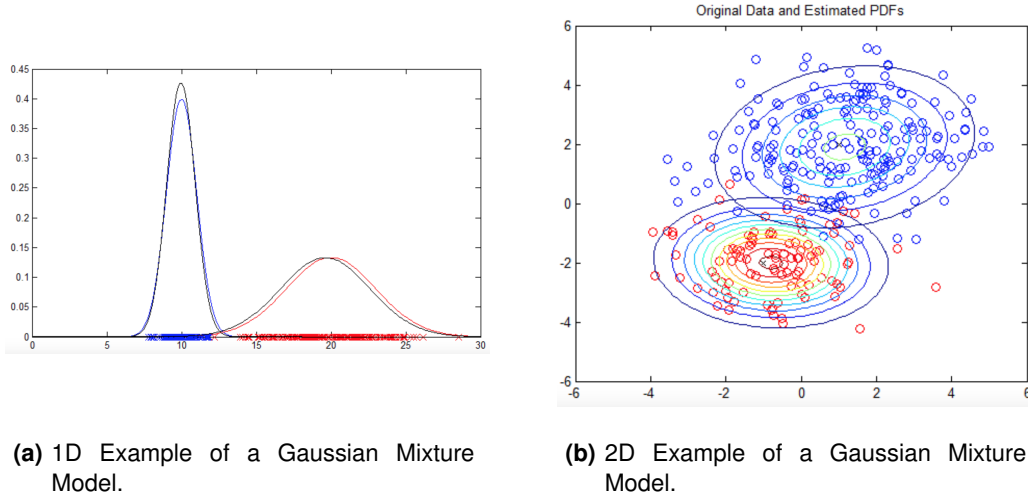
---

<sup>1</sup>SPEAR Toolkit: <https://github.com/guker/spear>

<sup>2</sup>LIBSVM Toolkit: <https://www.csie.ntu.edu.tw/~cjlin/libsvm/>

<sup>3</sup>Weka Software: <https://www.cs.waikato.ac.nz/ml/weka/>

<sup>4</sup>Images reference: <http://mccormickml.com/2014/08/04/gaussian-mixture-models-tutorial-and-matlab-code/>



**Figure 2.4:** Two distinct examples for the use of Gaussian Mixture Models in different dimensions.

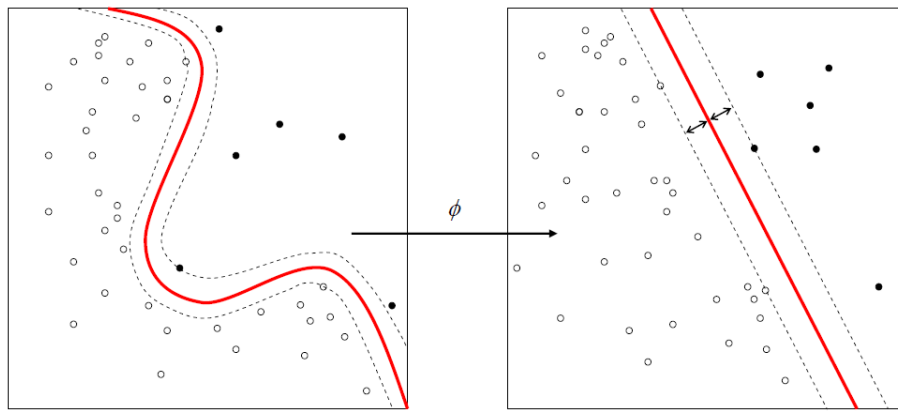
in speech recognition tasks, though it is much harder to make good use of large cluster machines to train DNNs on massive datasets [8].

The UBM is a GMM trained with a set of sample features from the general population and subsequently the specific model for each class is obtained using a maximum a posteriori adaptation from the UBM [7]. Typically used in speaker recognition, the GMM-UBM can be expanded to other tasks such as emotion recognition [7] or Parkinson's disease detection [29, 30].

Based on the GMM-UBM, new vector-based methods have emerged, such as GMM supervector approach, and the joint factor analysis or total variability based compensation methods. The GMM supervector is used to map each speech utterance to a high-dimensional vector space. An SVM can later be used to perform the classification of speaker vectors within the same space. In total variability modelling, the speaker and the channel variabilities of high-dimensional supervectors are jointly modelled as a single low rank total variability space. The low-dimensionality total variability factors extracted from a speech segment form i-vectors, which represent speech segments in a very compact and efficient way. Following the extraction, a channel compensation needs to be performed. Total variability modelling has shown to be a very powerful technique, which made it become the current standard.

### 2.1.3.2 Support Vector Machines

SVMs [31] are supervised learning models with associated algorithms that analyze data used for classification, pattern recognition and regression analysis. Given a set of training examples, each of them labelled to a specific class, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. The examples of the separate categories are divided by a clear gap that is as wide as possible and, the further the example is to that gap, the more confident is the classification. New examples are then mapped into that same



**Figure 2.5:** Mapping features to a space where the data is linearly separable, using the kernel function  $\phi$ .

space and predicted to belong to a category based on which side of the gap they fall.

However, the SVM only works if the data is linearly separable and yet, it can efficiently perform a non-linear classification by means of kernel functions (e.g. Linear, Polynomial and Radial) which map the original input set to a higher dimensional space, also known as *feature space*, making it linearly separable in that space. An illustration<sup>5</sup> of the mapping from an input space to a feature space may be inspected in Figure 2.5.

SVMs have the advantage of being simple and efficient learning models that can work with limited data. Moreover, if the proper kernel is used, it can separate any kind of data, although precautions need to be made against overfitting. Altogether, SVMs can have a very good classification performance compared to other classifiers [19].

### 2.1.3.3 Neural Networks

Neural networks are a computational model used in machine learning and other research fields. It consists of an interconnected group of artificial neurons that communicate with each other to process information.

DNNs are neural networks that contain many layers of hidden units between the input and the output (see Figure 2.6)<sup>6</sup>. These are typically designed as feedforward networks but were, later, successfully applied to recurrent neural networks, specially LSTM networks [32], for language modelling and processing. A recurrent network topology allows not only for data to move linearly along the network (feedforward propagation) but also backwards making them models with a bidirectional data flow, increasing its processing power. Moreover, the combination between LSTM networks, recurrent neural networks and end-to-end learning has been shown to give state-of-the-art results in several speech related tasks, such as speech recognition [33] and emotion recognition [14]. In fact, the state-of-the-art for emotion recognition is an end-to-end emotion recognition approach using DCRN [14], which is based on a combination of:

<sup>5</sup>Image reference: [https://en.wikipedia.org/wiki/Support\\_vector\\_machine](https://en.wikipedia.org/wiki/Support_vector_machine)

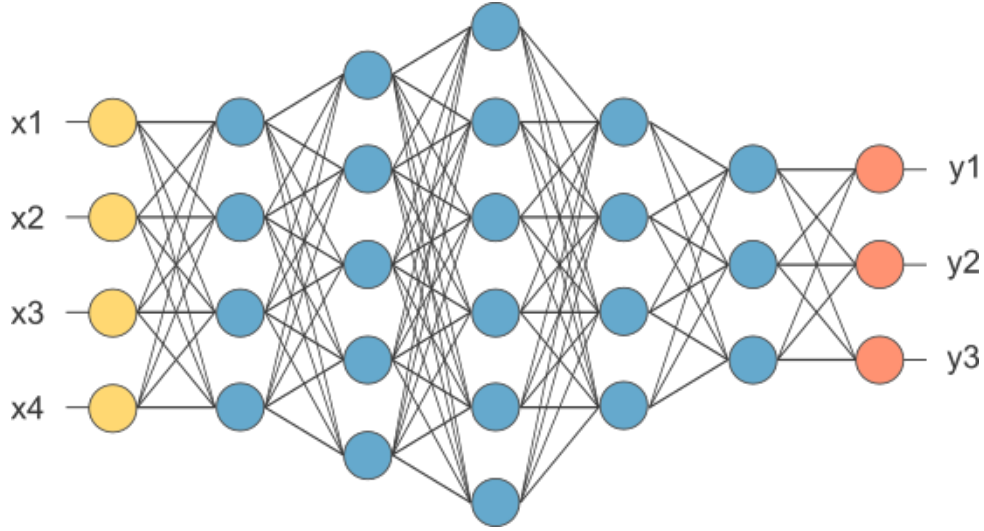
<sup>6</sup>Image reference: <http://www.opennn.net>

*Convolutional Neural Networks:* Modified Artificial Neural Networks (ANNs), used mostly in image recognition and, lately, to construct acoustic models of audio signals. In image recognition, it has been shown that Convolutional Neural Networks (CNNs) are able to directly classify raw pixels into high-level concepts without actually using hand-engineered feature extraction algorithms, which will result in less human interaction and better performance [34]. This fact may be also applied to feature learning in speech signals, making CNNs invaluable to end-to-end learning [14]. CNNs have hidden layers to extract features from the raw input signals without any pre-processing. These layers generate the features that will serve as input of other, succeeding, hidden layers, providing a much more accurate model for a specific task. Each feature extraction layer is composed of 1-d convolutions and max pooling. The convolution process consists in chains of layers and filters that are convoluted with the input signals (Equation 2.1), generating features which will, later, be processed by a max pooling operation, which essentially collects the maximum and most adequate feature from all adjacent outputs. In this case, it is used max pooling, but other types of pooling may also be used. Thus, the implementation of CNNs not only removes background noise, but also enhances specific and relevant bits of the input speech signal.

*Long Short-Term Memory Networks:* In a standard Recurrent Neural Network (RNN), the hidden layers of the network are recurrently connected, allowing the network to "remember" what happened in previous time steps. However, standard RNNs cannot, in practice, handle more complex problems that have long-term dependencies. Following this case, the LSTM was designed. LSTM is a recurrent neural network architecture [32] that has been designed in order to solve the long-term dependencies of the Convolutional RNNs, i.e. they are able to store information for long periods of time. In short, an LSTM is able to process arbitrary inputs by using information stored in its memory, allowing the decision process to be more accurate. Moreover, it knows what it should "forget" and what it should "remember", in order to minimize the loss function. Furthermore, the LSTM allows inputs to have variable length, by breaking the signal down into smaller bits and connecting them via recurrent links between units, which is a great advantage for speech recognition and other related tasks.

$$(f \circledast h)(t) = \sum_{k=-T}^T f(t) \cdot h(t - k). \quad (2.1)$$

This new DCRN approach makes the need for extracting hand-engineered features from speech to become obsolete. Instead, the features are learned in a process called *feature learning*, where features will be gradually updated and improved along the hidden layers of the network. With this, is it possible to automatically learn an accurate representation of the speech directly from the raw input signal that better suits the task at hand leading to better accuracies and computational times, outperforming all the traditional feature extraction approaches in terms of performance.



**Figure 2.6:** Sample architecture of a deep neural network with several hidden layers and units.

## 2.2 Privacy-Preserving Computation

Although biometric template protection schemes heighten the security of a given unique biometric characteristic, in a database, by transforming or encrypting it, the security of the biometric for some prediction tasks is severely compromised. Normally, in a matching phase, the stored template is decrypted at some point to perform the comparison, which means that even though the template is secure against attacks to the storage, it is still vulnerable in a matching phase. Regarding speech emotion recognition, the biometric (voice) has to be encrypted and secured throughout the emotion prediction model. For this reason, in order to preserve security at all times, privacy-preserving solutions, such as encrypted testing using Homomorphic Encryption (HE), Oblivious Transfer (OT), Garbled Circuits (GCs), differential privacy or even distance-preserving hash functions may be implemented to ensure protection and security throughout the whole process.

Currently, several privacy preservation methods for data mining are available. These include K-anonymity, classification, clustering, association rule, distributed privacy preservation, L-diverse, randomization, taxonomy tree, condensation, and cryptographic primitives [35]. Moreover, popular classification algorithms, such as Naïve Bayes, Decision Trees and Linear Discriminant Classifiers have been successfully developed to perform classifications over encrypted medical data [36].

### 2.2.1 Secure Multi-Party Computation

Secure Multi-Party Computation (SMPC) is a sub-field of cryptography that enables  $n$  parties, each with their own set of private data, to jointly compute a given function without any of them being able to retrieve any data from the other parties. The SMPC protocol considers that each party  $p_1, p_2, \dots, p_n$  with respective input  $x_1, x_2, \dots, x_n$  will be able to use other parties private data in a joint function  $f(x_1, x_2, \dots, x_n) = (y_1, y_2, \dots, y_n)$  and collect its own private output. It is important to note that each

party  $p_i$  only has access to its private input  $x_i$  and output  $y_i$  [37]. It is an ideal protocol for collaboration between untrustworthy parties.

There is a particular case of SMPC, namely the Secure Two-Party Computation (STPC), where one party has data but not the function and the other has the function but not the data. In an example, let us assume two parties:

- *Alice*: Has a certain function  $f_A(\cdot)$  and is willing to provide it as a service.
- *Bob*: Wishes to know the outcome of the function  $f_A(X_B)$ , which he does not possess, using his database  $X_B$ . Moreover, Bob does not trust Alice with his data, so he wants to compute the function without making his database public to Alice.

With STPC, Alice can perform  $f_A(X_B) = Y_{AB}$  without knowing anything about Bob's data and Bob will obtain the expected output. This is extremely useful for quality control services in callcenters, statistical analysis of banks finances, health diagnostics, homeland security, cloud computing and Privacy-Preserving Speech Mining (PPSM) tasks. The confidentiality present in the STPC is achieved through the exchange and computation between partial results in complex protocols that employ HE, OT and GC [13].

There has been extensive work in the development of SMPC protocols for machine learning related to Privacy-Preserving Data Mining (PPDM) in the literature [38]. However, the amount of literature regarding PPSM is scarce comparing to PPDM, which is one of the main motivations of this thesis.

### 2.2.1.1 Homomorphic Encryption

Homomorphic Encryption, first proposed by Rivest et. al. [39], is a public-key cryptosystem that allows for specific algebraic operations to be performed indirectly in the plaintext, by performing the same algebraic operations in the encrypted data or ciphertext [13]. Particularly, operations between two different plaintexts  $x$  and  $y$  are possible by manipulating their corresponding ciphertexts in the following way:

$$E(x) \oplus E(y) = E(x \otimes y) \quad (2.2)$$

However, a high degree of polynomial computations requires the use of large parameters in the scheme, which results in larger encrypted messages and slower computation times [40], not to mention that for every bit product operation the noise doubles in size, decreasing the accuracy for the tasks at hand.

A Fully Homomorphic Encryption (FHE) can perform both multiplication and addition operations on ciphertexts. Since all computations can be expressed by multiplications and additions, any kind of function can be computed in the encrypted domain. The biggest setback is that most FHE schemes implemented to this day are not very efficient and practical for large deployment. Nonetheless, FHE

remains a very attractive field of research and faster and more secure schemes are, at this moment, being developed. In 2009, Craig Gentry proposed the first real somewhat efficient FHE scheme [41] and, following this scheme, several FHE libraries have been implemented that manage to be efficient enough for FHE-based privacy-preserving researching purposes. These include YASHE [42] (Yet Another Somewhat Homomorphic Encryption), HELIB [43] (Homomorphic Encryption Library), and more recently, SEAL [4] (Simple Encrypted Arithmetic Library).

Despite this, there are still very practical and efficient partially homomorphic cryptosystems, such as RSA [44] and Pailler [45], and implementations of a series of joint techniques of machine learning, cryptography and software engineering (e.g. CryptoNets [40]), that may be helpful in devising new privacy-preserving schemes.

RSA is a very famous and one of the first practical public-key cryptosystems. Being a multiplicatively homomorphic scheme, RSA allows multiplication on plaintexts by multiplying ciphertexts. This multiplication computation is shown in Equation 2.3. On the other hand, a Pailler cryptosystem is an additively homomorphic scheme and, therefore, is able to only compute the addition of two plaintexts by multiplying two ciphertexts (Equation 2.4).

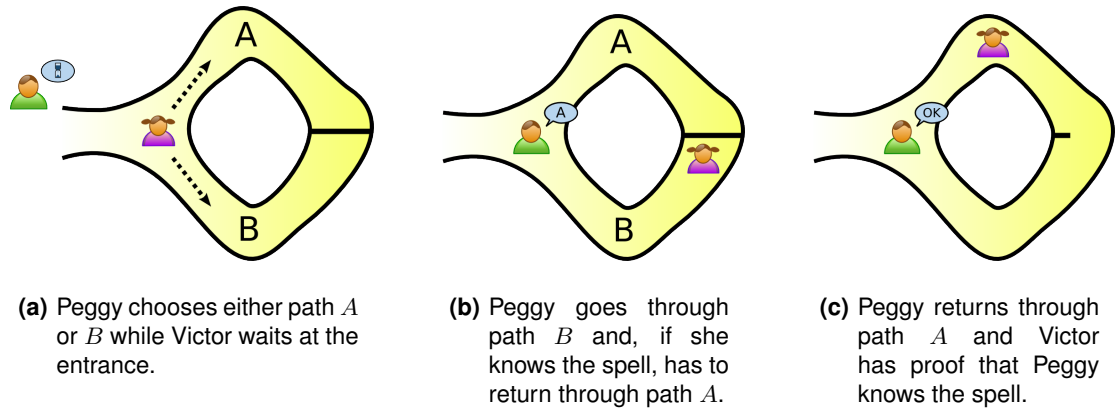
$$E(x) \times E(y) = E(x \times y) \quad (2.3)$$

$$E(x) \times E(y) = E(x + y) \quad (2.4)$$

Graepel et al. [46] suggested the use of HE in machine learning algorithms for binary classification that could be trained over encrypted data. However, since the computation of HE in a high polynomial degree is very slow, they used mainly low degree polynomials like linear discrimination classifiers. Later, Aslett et al. [47, 48] continued the work by presenting new ways to train machine learning models with Bayes classifiers and random forests. Finally, a Cryptonet scheme was introduced by Dowlin et al. [40]. In their work, they propose a new method to convert learned neural networks into cryptonets, which are neural networks that allow testing to be done over encrypted data. This method differs from the previous due to its superior accuracy and the fact that it imposes fewer requirements on the data owner. However, this method does not support training on the encrypted data itself.

### 2.2.1.2 Zero Knowledge Proof

Introduced by Goldwasser et al. [49], ZKP is defined as a proof of knowledge where the *prover* has some private data and needs to ensure a *verifier* that he or she actually has the information without revealing the actual content of the private data. In a simple example [50], a *prover*, Peggy, has a spell that allows her to open the door that separates two connected paths *A* and *B*. Victor wants to pay Peggy to teach him the spell, however, he does not want to pay her until he is sure she has, in fact, the spell she claims to have. On the other hand, Peggy will not teach Victor unless she gets paid first. To solve this problem, Peggy will choose either path *A* or *B* while Victor stands at the entrance of both paths.



**Figure 2.7:** Abstract example of a ZKP with a *prover*, Peggy, and a *verifier*, Victor.

Since Peggy knows the spell to open the door, she will return to Victor through a path different from the one she initially took. This way, Victor can be certain that Peggy knows the spell and is finally willing to pay. An illustration<sup>7</sup> for this example is provided in Figure 2.7.

A valid ZKP has to always satisfy the following requirements [37]:

- *Completeness*: The verifier should always accept the proof if the statement is true.
- *Soundness*: If the statement is false, the verifier has a negligible probability of accepting that statement.
- *Zero Knowledge*: The verifier should only learn about the veracity of the statement and nothing else.

In authentication, a user has an encrypted answer and sends it to a server to decrypt it. The server decrypts it and returns it to the user. The problem is that the user is not sure that the decrypted data he received from the server is the decryption of encrypted data he sent. A server may then perform a ZKP to assure the user that what he received was, in fact, his decrypted data. Besides authentication, ZKPs can be used in a variety of applications such as HE and other cryptosystems.

### 2.2.1.3 Oblivious Transfer

Oblivious Transfer (OT) is a cryptographic protocol in which a party sends different sets of information to another without actually knowing what was sent. The first protocol to be introduced was the one-out-of-two OT ( $OT_1^2$ ) [51] which used RSA primitives. In  $OT_1^2$ , Alice would send one of two bits to Bob, each 1/2 probability. Alice has no knowledge of the bit that was sent to Bob while Bob knows which one he received. The  $OT_k^n$  is a more generalized version of the  $OT_1^2$ . In this protocol, Alice sends  $k$  of  $n$  different data values with equal probability to Bob, having no knowledge which data values were sent.

<sup>7</sup>Image reference: [https://en.wikipedia.org/wiki/Zero-knowledge\\_proof](https://en.wikipedia.org/wiki/Zero-knowledge_proof)

There also exists the  $OT_1^n$  which works in a similar way to  $OT_1^2$  but Alice sends 1 from  $n$  data values instead.

Oblivious Transfer can be used to create STPC protocols though with less efficiency than with HE [37]. Despite this, OTs are very useful in other privacy preserving applications, including the selling of digital goods, solving the list intersection problem, exchanging mutually authenticated keys, among others [13].

#### 2.2.1.4 Garbled Circuits

Garbled Circuits is one of the most popular approaches to STPC allowing two parties, let us assume Alice and Bob, holding inputs  $a$  and  $b$ , respectively, to evaluate an arbitrary function  $y = f(a, b)$  without leaking any information about the inputs. The main idea is to encrypt the nodes and transition paths of a boolean circuit so that the party who evaluates only follows one path [13].

Let us take into account a random boolean circuit that corresponds to the function  $f(a, b)$ , which is known to both Alice and Bob. Alice generates the circuit and associates two random cryptographic keys  $K_i^0$  and  $K_i^1$  for each wire  $i$  of the circuit, where  $K_i^0$  and  $K_i^1$  represent the encoding of the bit equal to 0 and 1, respectively. Then, the encrypted output, lets call it  $R$ , for each logic gate  $g$  with inputs  $i$  and  $j$  is obtained the following way [52]:

$$EN_{K_i^{b_i}, K_j^{b_j}}^R (K_R^{g(b_i, b_j)}) \quad (2.5)$$

where  $b_i$  and  $b_j$  are the respective 1-bit or 0-bit for the  $i$  and  $j$  wire. After computing the Equation 2.5 for each combination of bits, four ciphertexts are obtained, which corresponds to the four possible encrypted outputs of the gate  $g$  and thereby to the garbled truth table of that logic gate. Finally, Alice sends the resulting GC along with her cryptographic keys to Bob so he can then evaluate it. It is important to note that the transfer of data between them is made through Oblivious Transfer (OT) to allow the evaluator to obtain the input keys while remaining oblivious to which one he received [52]. Finally, Bob can compute a key for the output wire of a specific gate by decrypting the four ciphertexts obtained where only one will be the correct decryption of the logic gate output. In fact, the circuit is designed to make the incorrectness of the decryptions apparent [13]. One of the most popular techniques that provides this useful characteristic is the Fairplay [53] technique, where given the mappings from the output wire keys to bits, the actual output of the desired function can be computed. Finally, Bob has the option to share the results of the computation with Alice if he wishes to do so, which might generate privacy issues [52].

In short, a garbled circuit protocol undergoes the following steps:

1. A boolean circuit describes the function to be computed by both parties. Both parties have the same knowledge about the circuit.
2. Alice, or the *garbler*, encrypts the circuit.

3. Alice sends the encrypted circuit with her own encrypted data.
4. Bob, or the *evaluator*, receives the encrypted inputs from Alice through OT.
5. Bob decrypts the circuit and obtains the encrypted outputs of the circuit.
6. Decrypting the output requires communication between both Alice and Bob. If one chooses not to share the information, none of them can learn about the output.

Some optimizations have been investigated and applied to garbled circuits in order to improve its efficiency and running times. Assuming a *semi-honest* model, Kolensikov and Schneider [54] rely on the Random Oracle assumption to create a one-round protocol which allows the use of XOR logic gates for "free", i.e., with no significant computational cost. Thus, XOR and XNOR do not need to be garbled. Instead, the evaluation is based on XOR-ing its garbled input values. Additional reductions in truth tables, execution time, bandwidth and energy have also been implemented [55, 56].

Since the first real and practical application of GCs in Fairplay [53], a great variety of other practical and efficient implementations took place [13, 52, 57].

### 2.2.2 Differential Privacy

Differential privacy constitutes a strong privacy-guaranteed standard for algorithms on aggregate databases. It is defined in terms of the application-specific concept of adjacent databases [58]. When having two almost identical databases (one with some specific information, and another without it), differential privacy ensures that the probability that a statistical query will return the same result is nearly identical in both databases. Usually, differential privacy can be achieved if the party is willing to add random noise to the outcome. This is necessary to produce a slightly different result, masking the contents of any given row in a database. Keep in mind that every time a query is performed, the noise will grow along with the total leakage of information, and may never decrease. It is also important to note that once the data is leaked, it may never be recovered. Moreover, in order to minimize the amount of leaked data, a compromise between accuracy and privacy must be met. The total amount of leakage allowed is named *privacy budget* and it determines how many queries are allowed to be performed on the database, before the amount of leaked information becomes big enough to challenge the privacy of the scheme. If the *privacy budget* is too high, then the accuracy will be better at the cost of leaking too much sensitive data. On the other hand, if the *privacy budget* is too low, the data is, in fact, very secured, but the accuracy will be significantly lower.

Abadi et. al. [58] successfully implemented a deep learning neural network under a modest privacy budget, and at a manageable cost in software complexity, training efficiency, and model quality. In fact, by modifying the Stochastic Gradient Descent algorithm with a norm clipping and noise addition to protect privacy, and by keeping track of the privacy and moments at each step, high accuracy scores for both MNIST (97%) and CIFAR-10 (73%) datasets were obtained.

### 2.2.3 Nearest-Neighbour with Randomized Embedding

Embedding techniques are techniques in which the high dimensionality vectors of data is reduced into a low dimensional one such that the distance between the original vector data points is preserved. Cryptographically speaking, if the distance between two data points is smaller than a certain threshold, one can assume that their corresponding hashes are close to each other. On the other hand, if the distance is greater than the threshold, no information can be obtained regarding the distance between the corresponding hashes. Locality-sensitive hashing, SBE, and SMH are very useful and simple to implement privacy-preserving techniques based on nearest-neighbour methods [59].

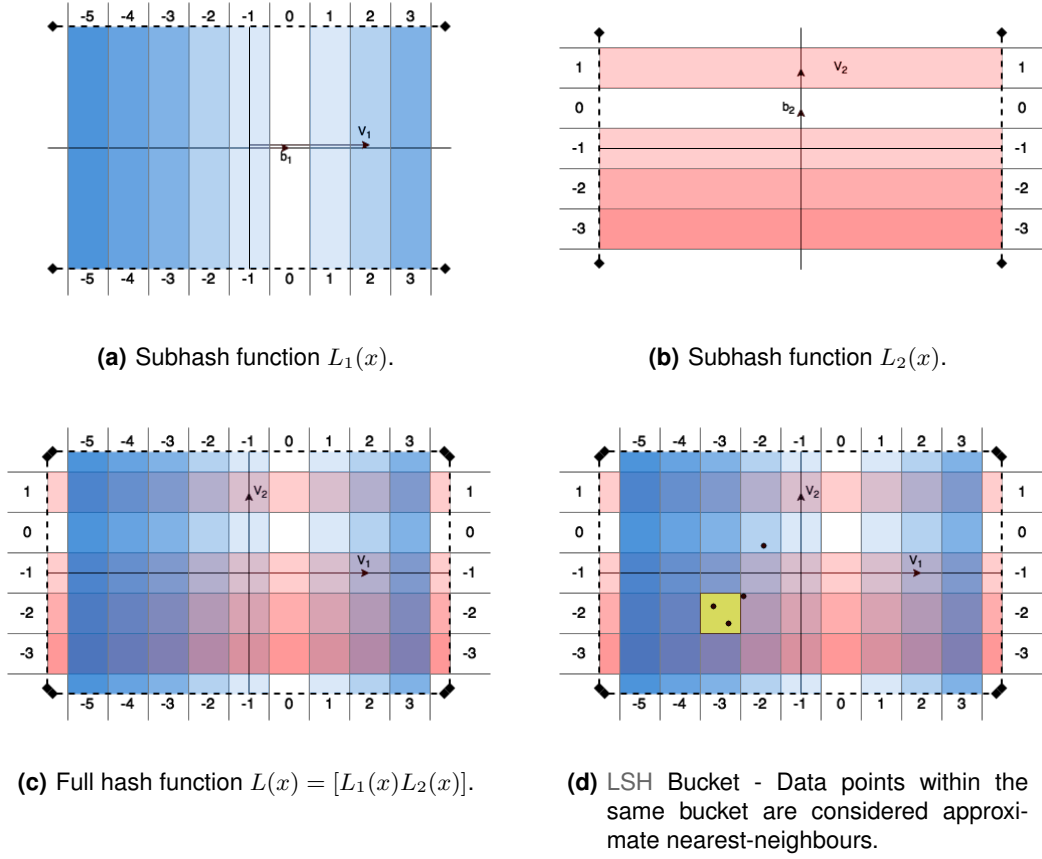
#### 2.2.3.1 Locality-Sensitive Hashing

LSH [13, 59] is a method for performing efficient approximate nearest-neighbour search by reducing the dimensionality of high-dimensional data into a lower dimension one, while preserving the distance between corresponding hashes. LSH applies a locality-sensitive function  $L(\cdot)$  on a vector  $x$  and projects it into a vector  $L(x)$  in a lower dimensional space, to which we refer as LSH key or bucket. A set of data points that map to the same bucket are considered to be approximate nearest-neighbours and their corresponding hash keys identical with high probability. The use of a set LSH functions is usually considered since it forms a smaller bucket. A smaller bucket allows for an even bigger probability of two data points, that map into the same bucket, being nearest-neighbours. However, the excessive use of multiple LSH functions also makes it more likely to miss valid data points. Considering an LSH function composed of  $n$  LSH functions  $L(x) = [L_1(x), L_2(x), \dots, L_n(x)]$  and the use of the Euclidean norm, each individual  $L_i$  is computed by

$$L_i(x) = L_i(x; V_i; b_i) = \left\lfloor \frac{x^T V_i + b_i}{w} \right\rfloor \quad (2.6)$$

where  $V_i$  is a random vector drawn from a normal distribution,  $b_i$  is a random number between zero and  $w$ , and  $w$  is the quantization width. One of the main advantages of using LSH is its efficiency. By pre-computing the keys, the approximate nearest-neighbour search can be done in time, sub-linear to the number of instances in the data set [59].

In order to better understand the concept of locality-sensitive hashing, a simple 2D example is presented in Figure 2.8.  $L_1(x)$  is the hashing function defined by vector  $V_1$  and scalar  $b_1$  while  $L_2(x)$  is the hash function defined by vector  $V_2$  and scalar  $b_2$ . It is important to note that, in these hash functions, the regions are perpendicular to the orientation of the vector  $V$  and the reference region is defined by the scalar  $b$ . Figure 2.8(c) illustrates an LSH function  $L(x) = [L_1(x), L_2(x)]$ , with 2 projections, where  $L_1(x)$  and  $L_2(x)$  correspond to the subhashing functions in Figure 2.8(a) and Figure 2.8(b), respectively. The function  $L(x)$  creates LSH buckets defined by the two hash components  $[L_1(x), L_2(x)]$ . All data points within each bucket are considered to be nearest-neighbours and will be represented by the same hash key. As it can be observed in Figure 2.8(d), the bucket  $L(x) = [-3, -2]$ , highlighted in yellow, contains two data points that are considered, with high probability, approximate nearest neighbours while



**Figure 2.8:** 2D Example using an LSH projection.

the data points in  $L(x) = [-2, -2]$  and  $L(x) = [-2, 0]$  have a very low probability of being approximate nearest-neighbours. On a final note, the concept of LSH can be associated with the concept of information leakage. This concept states that, if hashes are close enough, information regarding the original features can be extracted which might raise privacy concerns.

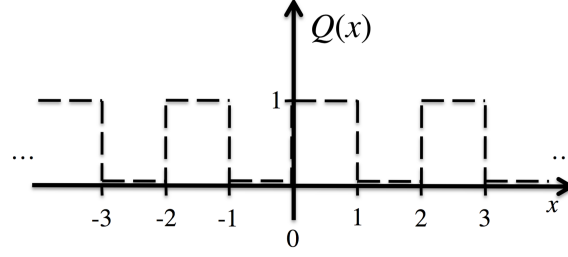
In a more generalized manner, a locality sensitive hashing function is a map  $h$  from  $\mathcal{M}$  to a universe  $\mathcal{U}$  such that, given any two vectors  $x_1, x_2 \in \mathcal{M}$ ,

- if  $d(x_1, x_2) \leq R$ , then  $h(x_1) = h(x_2)$  with probability at least  $P_1$ .
- if  $d(x_1, x_2) \geq cR$ , then  $h(x_1) = h(x_2)$  with probability at most  $P_2$ .

for some radius of interest  $R$  and some constant  $c$ . Thus, vectors that are less than a distance  $R$  apart have a high probability of being hashed to the same value, while they are highly improbable to do so if they are more than  $cR$  apart [2].

### 2.2.3.2 Secure Binary Embedding

SBE [1, 59] is a scheme that, unlike LSH, guarantees security on its own without actually requiring a cryptographic protocol as a wrapper. SBE converts vectors to bit sequences using band-quantized



**Figure 2.9:** Representation of the Universal Scalar Quantizer. This non-monotonic quantization function  $Q(\cdot)$  allows for universal rate-efficient scalar quantization and provides information-theoretic security [1].

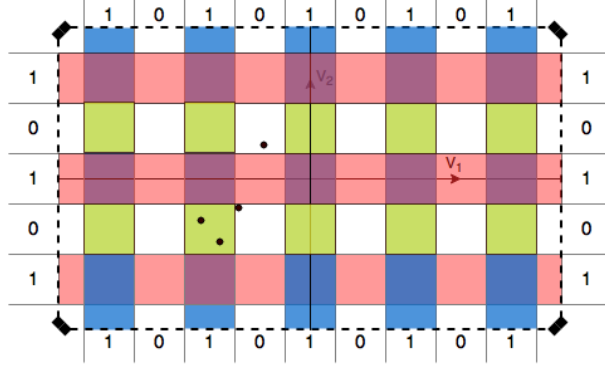
random projections. The general idea of this scheme consists on the fact that if the Euclidean distance between two vectors is lower than a threshold, then the Hamming distance between their hashes is proportional to the Euclidean distance between the vectors. On the other hand, if the distance is higher than the threshold, the hashes will not provide any kind of information regarding the distance between the vectors. These embeddings provide information-theoretic privacy on their own, which is not present in LSH, making them very attractive for privacy-preserving nearest-neighbour tasks. Other techniques based on SBE were recently introduced, such as SMH [2] which can be seen as a modular variant or generalization of SBE.

The SBE approach is based on the concept of Universal Scalar Quantization (USQ) [60], which redesigns the quantizer to have non-contiguous quantization regions. A complete USQ in vector form is computed by

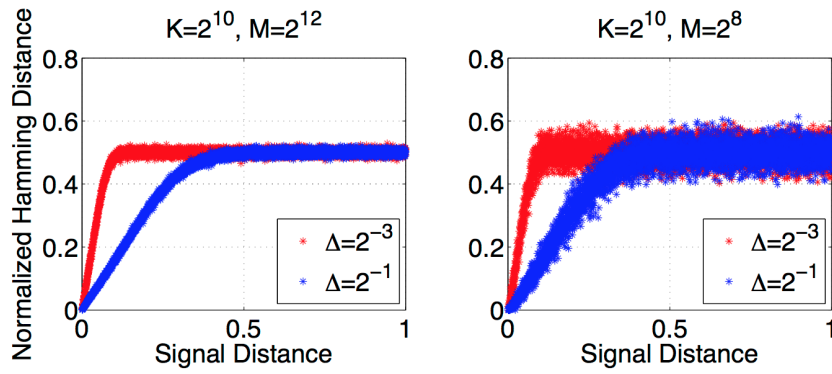
$$f(\mathbf{x}) = Q(\lfloor \Delta^{-1}(\mathbf{A}\mathbf{x} + \mathbf{w}) \rfloor) \quad (2.7)$$

where  $f(\mathbf{x})$  is a  $M$ -bit binary vector, which can be referred to as the hash of  $\mathbf{x}$ ,  $\mathbf{A}$  is a random  $M \times L$  matrix with normally distributed and i.i.d elements, where  $L$  is the number of samples,  $\Delta^{-1}$  is a scaling factor,  $\mathbf{w}$  is dither, uniformly distributed in  $[0, \Delta]$  and  $Q(\cdot)$  is the scalar quantizer, presented in Figure 2.9, operating element-wise on the input vector which can also be represented as  $Q(x) = [x \bmod 2]$ . This function is equivalent to a multi-bit quantization, where only the least significant bit is preserved and all other bits are discarded. In order to preserve security and privacy, the projection matrix  $\mathbf{A}$  and the dither  $\mathbf{w}$  should be treated as a secret key and not be revealed to the untrusted party.

This technique can be applied to the previous example in Figure 2.8 by replacing the present quantization function with the universal scalar quantization [60]  $Q(\cdot)$  in both subhash functions  $L_1$  and  $L_2$  in Figure 2.8(a) and Figure 2.8(b), respectively. In Figure 2.10 it is possible to observe the corresponding result to using an SBE approach on the previous example from Figure 2.8. It is important to note that all other cells with  $L(x) = [1, 0]$  are also highlighted in yellow since these are indistinguishable in their SBEs. Thus, all data points within any of these buckets are considered approximate nearest-neighbours with high probability.



**Figure 2.10:** 2D example using an SBE projection.



**Figure 2.11:** Behaviour of SBE using different parameters [1].

The binary hash generated by the USQ of the Equation 2.7 satisfies

$$|d_H(f(\mathbf{x}), f(\mathbf{y})) - g(\|\mathbf{x} - \mathbf{y}\|_2)| \leq \epsilon \quad (2.8)$$

where  $d_H(\cdot, \cdot)$  is the Hamming distance in the embedding space, and  $g(d)$  is a distance map which is approximately linear for small  $d$  but converges exponentially fast for a large  $d$  greater than a distance threshold. This threshold and the slope of the linear section are determined by the parameters  $\Delta$  and  $A$ . The behaviour of the SBE for different parameters can be observed in Figure 2.11.

By inspecting Figure 2.11, it is clear that as  $\Delta$  increases, the slope decreases and the linear section increases. This will allow more information to be leaked from signals that are more distant. Moreover, an increase in the number of bits  $M$  decreases the uncertainty in the mapping but limits the use of bandwidth.

On another note, for an  $M$ -bit embedding, the mutual information between the embeddings decays exponentially to zero as

$$I(f(x); f(y)|d) \leq 10M^{-cd^2} \quad (2.9)$$

where  $c$  is a constant that depends on the embedding parameters. This fact will make sure that a

potential eavesdropper or untrusted third party cannot reconstruct the signals or obtain any information regarding the distance between them.



# 3

## Privacy-Preserving Emotion Classifier Using Distance-Preserving Hashing

### Contents

---

3.1	Introduction . . . . .	31
3.2	Baseline of Emotional Classifier using SVMs . . . . .	31
3.3	Emotion Classification Using SBE . . . . .	34
3.4	Emotion Classification Using SMH . . . . .	39
3.5	Privacy Analysis . . . . .	44
3.6	Case Example . . . . .	44
3.7	Summary . . . . .	47

---



This chapter contains experiments performed on a couple of schemes using nearest neighbors with randomized embeddings on an emotional speech dataset. Section 3.1 introduces the topic and problem at hand. Section 3.2 presents the results of a speech emotion classifier baseline using the SVM learning model and hand-engineered features. A privacy-preserving approach using SBE is implemented and tested in Section 3.3 while, similarly, an SMH-based approach is implemented and tested in Section 3.4. Section 3.5 analyzes the privacy of both privacy-preserving schemes and, an example on how to use this scheme in a real world scenario is illustrated and detailed in Section 3.6. Finally, a small summary with the main conclusions of the chapter is present in Section 3.7.

## 3.1 Introduction

Emotion classification is still one of the most challenging tasks in speech processing. Its basic principle is to analyze the acoustic differences that occur when uttering the same thing under different emotional states [11]. Several techniques and approaches have already been widely investigated by the research community, most of them associated with acoustic/prosodic feature extraction and/or spectral feature analysis [11, 15]. The most common features for emotional classification are the MFCCs, PLPs and LPCCs. In terms of classifiers, the most common are based on HMMs/GMMs, SVMs, and on deep learning techniques. The most recent trend in emotion classification is based on end-to-end approaches, without any engineered feature extraction, directly from the raw input signal [14].

The automatic classification of emotions is a very active research topic in the Human-Computer Interaction field and it extends to a wide range of applications. These include quality long-distance teaching enhancement, quality control for call centers, clinical diagnosis, video games, among others [19].

With the rising concern for privacy and data security from parties with malicious intent, emerges the need for devising accurate privacy-preserving schemes using commonly used techniques for classification, regression, clustering, among others. Privacy-preserving LSH schemes such as, SBE and SMH, and other approaches, namely SMPC using cryptographic primitives have been researched throughout recent years allowing users to effectively perform computations over hashed or encrypted data with a high security assurance.

A large variety of software may be employed to perform the feature extraction, namely HTK, PRAAT and openSMILE/GeMAPS, and to perform the speech modelling, such as SPEAR, MATLAB, LIBSVM and WEKA.

## 3.2 Baseline of Emotional Classifier using SVMs

As mentioned in Section 2.1, a Support Vector Machine [31] (SVM) is a simple supervised machine learning algorithm useful in a variety of tasks. It is able to not only classify data linearly, but also, non-linearly, with the help of different kernel functions. They are very efficient when working with limited

training samples, although precautions against overfitting are always in order. Nonetheless, this algorithm is quite simple to implement to suit any task and gives fairly accurate results.

### 3.2.1 Feature Extraction and Classification

In the scope of this thesis, privacy will be implemented into the baseline, which means that one party will perform emotion classification on some encrypted data provided by another source or party.

For the implementation of the baseline, the following sets of features were considered:

- *openSMILE* [61]: a novel open-source feature extractor for incremental processing, consisting of audio low-level descriptors, loudness, MFCCs, LPCs, line spectral frequencies, fundamental frequency and formant frequencies, which sum up to a total of 6374 features.
- *GeMAPS* [62]: a basic standard acoustic parameter set, consisting of 56 features, that, in contrast to a large brute-force parameter set, such as openSMILE, presents only the most important and relevant voice parameters. Large brute-force feature sets are well known to foster over adaptation of classifiers to the training data in machine learning problems, reducing their generalization capabilities to unseen data. These minimalistic parameter sets might reduce this danger and lead to better generalization in cross corpus experiments and ultimately in real-world scenarios.
- *eGeMAPS* [62]: an extended version of the GeMAPS feature set that, besides implementing prosodic, excitation, vocal tract and spectral descriptors, also contains a small set of cepstral descriptors, such as the first four MFCCs, the spectral flux and the additional second and third formant bandwidth, all of which are known to increase the accuracy of automatic affect recognition over a pure prosodic and spectral parameter set. Adding these to the previous set, the eGeMAPS has a total of 88 features. Only low order MFCCs are used since these are the most relevant for affect and paralinguistic voice analysis tasks. On the other hand, high order MFCCs are better to reflect fine grained energy distributions, which are more relevant to identify phonetic content than non-verbal voice attributes.

### 3.2.2 Experimental Setup

The corpus considered for all experiments is the latest version of "Let's Go" corpus. "Let's Go" [63] is a public dialogue system that provides the bus schedule information for Pittsburgh's Port Authority buses during off-peak hours. The corpus used on the baseline of the emotional classifier was the *LEGOv2.1* corpus, which is an updated version of the original "Let's Go" Corpus [5,63,64] with emotional labels provided by an expert rater. It contains a total of 547 calls to the callcenter, where 302 of those have emotional annotations, and a total of 13,836 system-user exchanges, where 4,243 of those have emotional annotations. The emotional labels are provided as a binary class representation, where each exchange is either classified as *neutral* or *angry*, and a multiclass representation, where each exchange

**Table 3.1:** Results of the tests performed on the different sets using different features. The features show which features were used to train and classify the sets. Correct/Total Instances represents the number of correct predictions within the total amount of recordings in the set. Finally, accuracy represents the accuracy classification for each set.

Features	Set	Correct/Total Instances	Accuracy
<i>openSMILE</i>	Validation	544/657	82.80 %
	Test	462/581	79.52 %
<i>GeMAPS</i>	Validation	582/657	88.58 %
	Test	481/581	82.79 %
<i>eGeMAPS</i>	Validation	590/657	89.80 %
	Test	481/581	82.79 %

is classified with *friendly*, *neutral*, *slightlyAngry* or *veryAngry*. More information about this corpus is present in the Appendix A.

Three arranged sets with labelled emotional information were used. A training set with a total of 3,005 audio files is used to train an SVM model with the LIBSVM toolkit, which is an integrated software for support vector classification, regression and distribution estimation. A validation set made up of 657 audio files is mainly used to check the progress and the improvement of the baselines if changes are made to it. And finally, a test set made up of 581 audio files which is used to check the accuracy of the baseline. All of these three sets are labelled with its corresponding emotional state (*neutral* or *angry*), previously classified by an expert rater.

Features corresponding to each parameter set were extracted using the openSMILE toolkit for the training, validation and test sets. The results for the validation and test sets, using the different parameter sets discussed are presented in Table 3.1.

### 3.2.3 Discussion

As it can be depicted from Table 3.1, the classification using *eGeMAPS* provides the best accuracy. Moreover, the introduction of cepstral descriptors allows for a better accuracy in automatic affect recognition over pure prosodic and spectral parameter sets such as *GeMAPS*. On another note, the huge amount of features in the openSMILE parameter set will result in large computation times, which will make the training of models much more computationally demanding.

The amount of training data provided is always one big factor that influences the accuracy of the classifier and, the larger the amount of training data, the better is the accuracy of the classifier. Nevertheless, training a classifier with an immense amount of training data in a realistic time period is very challenging, computationally-wise and so, a compromise solution is often adopted. Another thing worth mentioning is that the labelling, by raters, is based on the rater's biased opinion, which means that some of the labelled exchanges might be challenging for the rater to classify leading to, sometimes, non-agreeable ratings across the three raters and thus, incompatible classifications between the classifier and the raters. The choice of the best machine learning algorithm for a certain task as well as

the use of other state-of-the-art approaches has also been shown to increase the accuracy score of the classifier [14, 19].

### 3.3 Emotion Classification Using SBE

The preservation of privacy in data mining has emerged as one of the biggest fields of research in machine learning. The increase usage of the internet and other machine-based ways of communication to perform exchanges of classified or sensitive information between untrustworthy parties has lead to an intense pursuit for new and practical ways of hiding and securing data over communication channels, as described in the previous chapter.

#### 3.3.1 Secure Binary Embedding

Secure Binary Embedding [1, 59] (SBE), discussed in Section 2.2.3.2, is a technique that hides and secures data by converting vectors into bit sequences using band-quantized random projections. SBEs are based on the fact that the euclidean or cosine distance between two vectors is proportional to the hamming distance between their corresponding hashes. If the euclidean or cosine distance between the features is lower than a certain threshold, controlled mainly by a scaling factor  $\Delta$ , then the hashes leak information about the true distance between the vectors, otherwise no information is leaked. This means that the true distance between feature vectors is proportional to the hamming distance between their hashes if and only if the true distance between the features vectors is smaller than a certain threshold. Furthermore, these embeddings provide information-theoretic privacy on their own which makes them very attractive for privacy-preserving nearest-neighbour tasks.

The process for computing the SBE hashes is based on the Universal Scalar Quantizer (USQ) and is calculated as follows:

$$\begin{aligned} h(\mathbf{x}) &= Q(\lfloor \Delta^{-1}(\mathbf{A}\mathbf{x} + \mathbf{w}) \rfloor) = \\ &= \lfloor \Delta^{-1}(\mathbf{A}\mathbf{x} + \mathbf{w}) \rfloor \mod 2, \end{aligned} \quad (3.1)$$

where  $h(\mathbf{x})$  is a  $M$ -bit binary vector, which can be referred to as the hash of  $\mathbf{x}$ ,  $\mathbf{A}$  is a random matrix with normally distributed and i.i.d elements,  $A \sim \mathcal{N}(0, 1)$ ,  $\Delta$  is a scaling factor,  $\mathbf{w}$  is the dither, uniformly distributed in  $[0, \Delta]$ .  $\mathbf{A}$  and  $\mathbf{w}$  are independent.  $Q(\cdot)$  is the scalar quantizer, presented in Figure 2.9, operating element-wise on the input vector and allows for a binary representation of the hashes.

It is important to note that  $\mathbf{A}$  and  $\mathbf{w}$  are treated as secret keys, meaning that only the individual in possession of these variables is able to learn about the true distance between the original feature vectors and recover  $\mathbf{x}$ . A potential impostor or eavesdropper, not having access to  $\mathbf{A}$  and  $\mathbf{w}$ , cannot learn or infer anything about feature vector  $\mathbf{x}$  from its embedding  $h(\mathbf{x})$ . Even if the impostor was able to

obtain  $\mathbf{A}$  and  $\mathbf{w}$ , it is computationally hard to derive  $\mathbf{x}$  from  $h(\mathbf{x})$  unless the imposter has some apriori knowledge about  $\mathbf{x}$ .

### 3.3.2 Experimental Setup

For the implementation of this SBE approach, the *eGeMAPS* baseline was considered, not only because it provided the best accuracy for the sets considered but also due to the amount of parameters it has. As the number of parameters or features grows, so does the execution time for the generation of the SBE hashes and its hamming distance computation. Both  $\Delta$  and  $M$  affect the behaviour of the SBE, however,  $M$  by itself is not useful since different values of  $L$  (dimensionality of the supervector) require different values of  $M$ . Knowing this, an alternative bpc is used, which is computed as  $M/L$  and controls the variance of the universal quantizer [13]. Hence, the use of a large set of features results in a much greater  $M$  for the same bpc as another parameter set with much fewer features and, therefore, slower computation times.

The SBE hashes were computed using MATLAB software for different percentages of leakage and different values of bpc in order to get a better understanding of how these parameters affect the accuracy of the classification. Leakage will be defined as the amount of hashes that leak information and is computed by

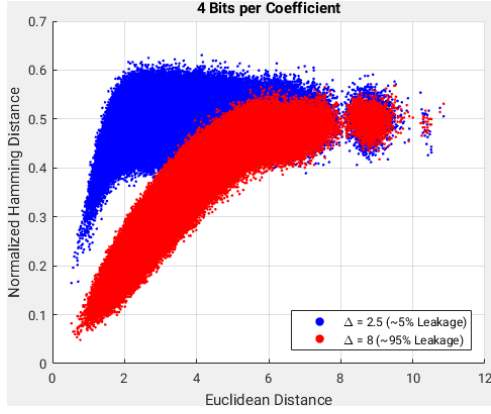
$$Leakage = \frac{\#leaks}{\#hashes}, \quad (3.2)$$

where  $\#leaks$  is the total number of hashes that have their normalized hamming distance below a certain privacy threshold, which is empirically set at 0.475, and  $\#hashes$  is the total number of hashes. The privacy threshold is set by closely inspecting the relationship plot between hamming distance and euclidean distance, present in Figure 3.1. The privacy threshold will be the hamming distance where the proportionality between both distances ends, i.e., where the hashes stop leaking information. Even though the leakage is affected by both  $\Delta$  and bpc, it varies almost exclusively with the scaling factor  $\Delta$ .

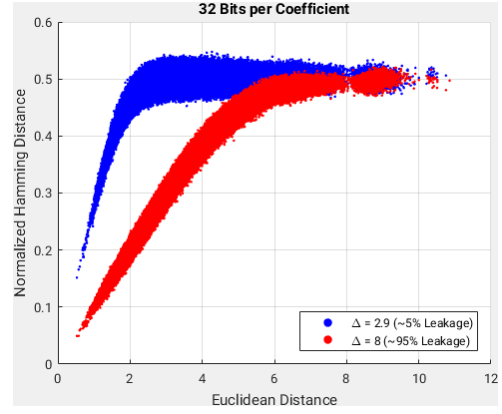
### 3.3.3 Results

#### 3.3.3.1 Distance

The main idea behind secure binary embeddings is that the true distance of the features of a signal, namely euclidean or cosine distance, is proportional to the hamming distance of their corresponding hashes until a certain threshold. This effect can be easily seen in Figure 2.11 back in Section 2.2.3.2. It is important to note that by itself, the hamming distance only preserves the euclidean distance. In order to also preserve the cosine distance, there is a need to normalize the feature vector before applying the hash. To be sure that the hashes computed for this task are correct and only leak information about the true distance of the signal if it is below a certain threshold, controlled by the parameter  $\Delta$ , the true distance between each training and validation set features was calculated, namely the euclidean

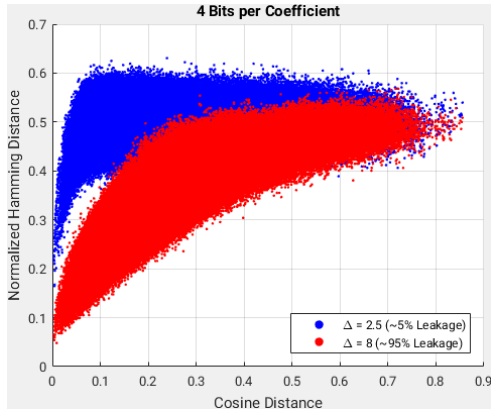


(a) 4 Bits Per Coefficient.

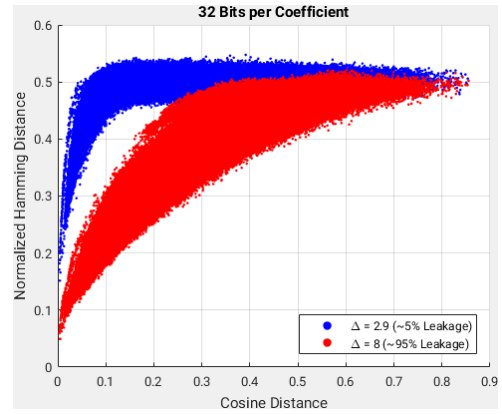


(b) 32 Bits Per Coefficient.

**Figure 3.1:** Euclidean distance between each training and validation set features for different values of bpc and  $\Delta$ .



(a) 4 Bits Per Coefficient.



(b) 32 Bits per coefficient.

**Figure 3.2:** Cosine distance between each training and validation set features for different values of bpc and  $\Delta$ .

distance and the cosine distance. Later, following the SBE hash computation of the sets, the normalized hamming distance between the hashes from the training set and the validation set was also computed and plotted against the cosine and euclidean distance between the true features, using different values of leakage and bpc. The behaviour of the SBEs when using euclidean distance for the feature vectors and hamming distance for the hashed vectors, using 4 and 32 bpc with  $\sim 5\%$  and  $\sim 95\%$  leakage, respectively, is presented in Figure 3.1. In addition, the same parameters were chosen when using cosine distance instead of euclidean. The behaviour of the hashes using the cosine distance, for the same parameters as in Figure 3.1, can be inspected in Figure 3.2.

From Figures 3.1 and 3.2, one can clearly state that the SBE hashes computed for the feature vectors do leak information about the true distance of the vectors. It can also be depicted that the amount of information leaked is mainly dependent of the  $\Delta$  parameter. In other words,  $\Delta$  allows us to adjust the distance threshold until which the Hamming distance is informative. Furthermore, the bpc seems to decrease the variance of the hamming distance, which results in a better accuracy in terms of the true distance of the original feature vectors.

### 3.3.3.2 Training and Classification

In our experiments, an SVM based classifier is trained to distinguish if a target speaker is angry or emotionally neutral. To do this, a feature supervector derived from a given recording is classified by an SVM classifier using a Radial Basis Function (RBF) kernel, which is given by

$$k(\mathbf{x}, \mathbf{y}) = e^{-\gamma \cdot d^2(\mathbf{x}, \mathbf{y})}, \quad (3.3)$$

where  $d^2(\mathbf{x}, \mathbf{y})$  corresponds to the euclidean distance between  $\mathbf{x}$  and  $\mathbf{y}$ , and  $\gamma$  corresponds to the scaling factor.

The fact that the hamming distance between SBE hashes  $h(\mathbf{x})$  and  $h(\mathbf{y})$  is proportional to the euclidean distance between  $\mathbf{x}$  and  $\mathbf{y}$  if and only if the euclidean distance is smaller than a certain privacy threshold, controlled mainly by  $\Delta$ , allows for a straightforward implementation of the SBE approach by simply training an SVM classifier with a modified RBF that, instead of analyzing the euclidean distance between the hashes, analyzes their normalized hamming distance. The RBF-Hamming kernel would then be computed as

$$k(\mathbf{x}, \mathbf{y}) = e^{-\gamma \cdot d_H^2(h(\mathbf{x}), h(\mathbf{y}))}, \quad (3.4)$$

where  $d_H(h(\mathbf{x}), h(\mathbf{y}))$  corresponds to the normalized hamming distance between the hashes  $h(\mathbf{x})$  and  $h(\mathbf{y})$ , and  $\gamma$  corresponds to the scaling factor. Note that for a given  $\mathbf{A}$  and  $\mathbf{x}$ , the modified kernel closely approximates the conventional RBF for small  $d(\mathbf{x}, \mathbf{y})$ , but varies significantly when  $d(\mathbf{x}, \mathbf{y})$  becomes larger.

Afterwards, having computed all hashes for different bpc and leakage, several SVM-based models using the modified RBF kernel were computed for different values of the cost ( $C$ ), and  $\gamma$  parameters.

**Table 3.2:** Accuracy results for different values of `bpc` and leakage. Cells presented as a crossed line represent the lack of information leaked necessary for an accurate classifier. The most suitable values that allow for a rather accurate and private classifier are highlighted in bold.

<code>bpc</code> \ Leakage	~5%	~25%	~50%	~75%	~95%
4	—	77.45 %	81.24 %	81.58 %	82.44 %
8	78.83 %	79.52 %	82.44 %	82.44 %	81.41 %
16	79.86 %	80.38 %	82.27 %	82.62 %	82.44 %
32	79.35 %	<b>82.62 %</b>	81.93 %	81.93 %	82.44 %

The reason for doing this comes from the fact that by altering the `bpc` and leakage, the conditions of the experiment will vary, and, therefore, several models with distinct parameters have to be computed and each used to predict and classify the hashed validation set. Hence, for each value of `bpc` and leakage, the prediction for the hashed validation set will be maximum for a particular pair of RBF parameters ( $C$  and  $\gamma$ ), which will then be used to classify the hashed test set. The results obtained for the prediction of the test set using SBE hashes derived from recordings of the Let's Go corpus [5, 63, 64] is shown in Table 3.2. All experiments were performed using the LIBSVM toolkit.

### 3.3.4 Discussion

From a close inspection of the results in Table 3.2, it is possible to notice a strong relationship between the amount of information leaked by the hashes regarding the true distance of the feature vectors with the accuracy of the classifier, which is expected, specially for smaller `bpcs`. As a matter of fact, by common sense, the accuracy of a classifier is as great as the total amount of knowledge known about the true distance of the feature vectors, hence, the bigger is the amount of information leaked by the SBE hashes, the better is the accuracy of the classifier. Furthermore, in a more theoretical way, by inspecting Figures 3.1 and 3.2, one can clearly see the proportionality of knowledge leaked, which is controlled by the scaling factor  $\Delta$ . On the other hand, it seems that `bpc` has a slightly weaker relationship with the accuracy of the classifier, specially when the leakage is higher. This effect might be explained by the fact that there is a clear separation between the utterances that classify as either *angry* or *neutral*, which means the classification is only slightly affected by the noise introduced when generating the SBE hashes [13]. Also, from Figures 3.1 and 3.2, it is possible to see the difference that higher `bpcs` make. Higher `bpcs` effectively diminish the "noise" introduced in the hash generation and allow for more viable information about the distance to be leaked.

The best result using the SBE hashes would be when the `bpc` is 32 and the leakage is approximately 25% (highlighted in Table 3.2) due to the fact that the SBE hashes only leak a total of 25% of information regarding the true distance of the hashes and because the bigger the `bpc`, the less noise is introduced, resulting in a slightly better accuracy. It is important to highlight the fact that each result was computed with different random keys  $A$  and  $w$ , and therefore, may cause small deviations regarding accuracy results. Even so, these deviations are not great enough to cause difficulties in an analysis of the accuracy

of the private emotional classifier.

The best accuracy results using SBE hashes, 82.62%, is very similar to the results of the non-private baseline using euclidean distance, 82.79%. This is probably the most important result of this experiment. It means that using an SBE approach for data hiding and secure machine learning offers little degradation of accuracy results and, therefore, is very promising for upcoming new techniques in data hiding and other privacy-preserving techniques for speech processing tasks.

## 3.4 Emotion Classification Using SMH

### 3.4.1 Secure Modular Hashing

Secure Modular Hashing, or just SMH [2], is a generalization of the SBE technique. Both schemes rely on the notion that if two encrypted values  $h(x)$  and  $h(y)$  are close enough to each other, then the true distance between  $x$  and  $y$  can be guessed with high probability. SMH introduces a generalized modular parameter to the hash generation function which controls the behaviour of the quantizer used in SBE. The quantizer used in the SBE scheme, presented back in Figure 2.9, which can also be represented as  $Q(x) = x \bmod 2$ , is generalized in the SMH approach with the introduction of a new parameter  $k$  that controls the modulo of the hash generation, modifying the quantizer into  $Q(x) = x \bmod k$ . Hence, the hashes in the SMH scheme are computed as

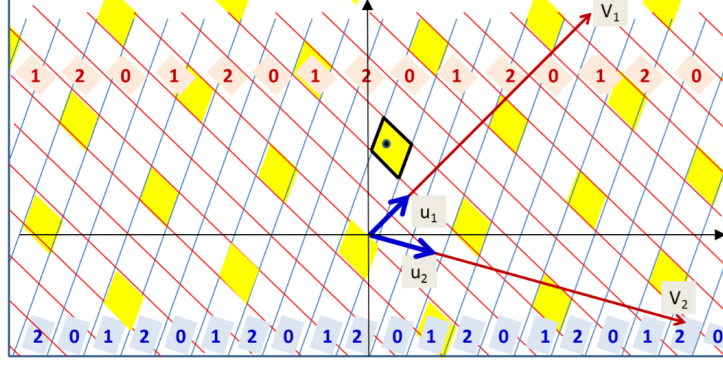
$$\begin{aligned} h(\mathbf{x}) &= Q(\lfloor \Delta^{-1}(\mathbf{A}\mathbf{x} + \mathbf{w}) \rfloor) = \\ &= \lfloor \Delta^{-1}(\mathbf{A}\mathbf{x} + \mathbf{w}) \rfloor \bmod k, \end{aligned} \quad (3.5)$$

where  $h(\mathbf{x})$  is the hashed product of the vector  $\mathbf{x}$ ,  $\mathbf{w} \sim \text{unif}(0, k)$ , also known as dither, and  $A \sim \mathcal{N}(0, 1)$  independent matrix.  $\Delta$  is the scalar factor of the SMH that controls the variance of  $A$ . It is noticeable that, one can simply set the value of  $k = 2$  to have an SBE scheme.

To have a better understanding of the SMH approach, an illustrated example, such as the one in Figure 2.10 in Section 2.2.3.2, may be created. While in Figure 2.10, there are only four ( $2^2$ ) types of hash buckets due to the behaviour of the quantizer of the SBE, in SMH there will be  $k^2$  different hash buckets, where  $k$  represents the modulus operation ( $x \bmod k$ ). This example is presented in Figure 3.3. As can be depicted from Figure 3.3, any hashes that fall within the same coded hash bucket, in this case,  $[h(\mathbf{x}) \ h(\mathbf{y})] = [1 \ 1]$ , highlighted in yellow, are close enough and, therefore, will reveal information about the true distance of vectors.

### 3.4.2 Experimental Setup

The same experimental setup used in the SBE approach was also considered for all SMH experiments. Similar to SBE, both the scalar factor  $\Delta$  and the number of samples  $M$  influence the behaviour



**Figure 3.3:** SMH example using two hashes with  $k = 3$  [2].

of the SMH scheme, although this new approach introduces a new variable  $k$  modulus, that significantly influences the distance in which hashes of two vectors reveal information about the real vectors. The number of samples  $M$  does not influence the behaviour by itself, hence, a new variable computed as  $M/L$ , where  $L$  is equal to the dimensionality of the supervector [13]. We will name this new variable measurements per coefficient (mpc). Notice that, for this approach, we use mpc instead of bpc since, depending on  $k$ , we will have a  $k$ -modular representation of the hashes, and not always a binary representation.

The SMH hashes were computed using MATLAB software for different values of  $\Delta$  and  $k$ . A static value of 32 was considered for the mpc since the leakage varies very little with the mpc and we wish to study the influence of the  $k$  parameter on the the number of leaks provided by the hashes and, therefore, the accuracy of the emotion classifier. The leakage is calculated as in Equation 3.2, although the privacy threshold varies with  $k$ . The value for this threshold has to be empirically set from plotting  $d_H(h(\mathbf{x}), h(\mathbf{y}))$  against  $\|\mathbf{x} - \mathbf{y}\|$ , as depicted in Figure 3.5.

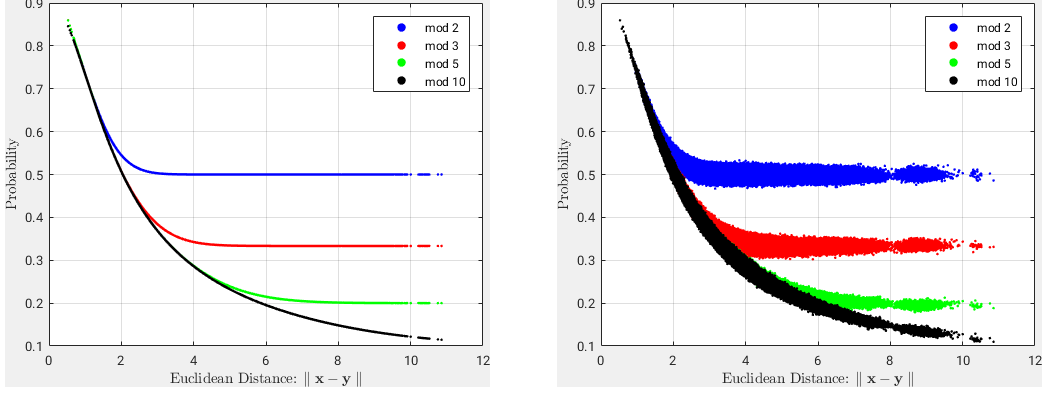
### 3.4.3 Results

#### 3.4.3.1 Probability

The probability that hashes derived from two distinct vectors  $\mathbf{x}$  and  $\mathbf{y}$  take the same value depends extremely on the euclidean distance between the correspondent non-hashed vectors. This probability decreases exponentially with the euclidean distance  $\|\mathbf{x} - \mathbf{y}\|$ . In fact, the theoretical probability can be computed against the euclidean distance the following way [2]:

$$P(h(\mathbf{x}) = h(\mathbf{y})) = \frac{1}{k} \left( 1 + 2 \sum_{i=1}^{\infty} \text{sinc}^2\left(\frac{i}{k}\right) e^{-2\left(\frac{\pi\|\mathbf{x}-\mathbf{y}\|i}{\Delta k}\right)} \right). \quad (3.6)$$

The theoretical probability, as well as the estimated probability derived from the feature vectors,  $P(h(\mathbf{x}) = h(\mathbf{y}))$  for different values of  $k$  is presented in Figure 3.4. Figure 3.4(a) is the theoretical probability in which two hashed vectors take the same value depending on the euclidean distance of



(a) Theoretical probability  $P(h(\mathbf{x}) = h(\mathbf{y}))$  as a function of the euclidean distance  $\|\mathbf{x} - \mathbf{y}\|$ , derived from the Equation 3.6.

(b) Estimated probability  $P(h(\mathbf{x}) = h(\mathbf{y}))$  as a function of the euclidean distance  $\|\mathbf{x} - \mathbf{y}\|$ , derived from Equation 3.6, using the hashed values of the datasets.

**Figure 3.4:** Comparison between theoretical and estimated probability  $P(h(\mathbf{x}) = h(\mathbf{y}))$ , and euclidean distance for different values of  $k$ ,  $mpc = 64$  and  $\Delta = 3$ . Both derived from the hashed Let's Go datasets used in all other previous experiments.

non-hashed vectors, while Figure 3.4(b) is the estimated probability, computed as  $1 - d_H(h(\mathbf{x}), h(\mathbf{y}))$ .

From a quick inspection of Figure 3.4, one can easily realize that  $P(h(\mathbf{x}) = h(\mathbf{y}))$  decreases exponentially with  $\|\mathbf{x} - \mathbf{y}\|$  and saturates in  $1/k$ . This means that, as soon as the probability saturates, the SMH stops leaking information.

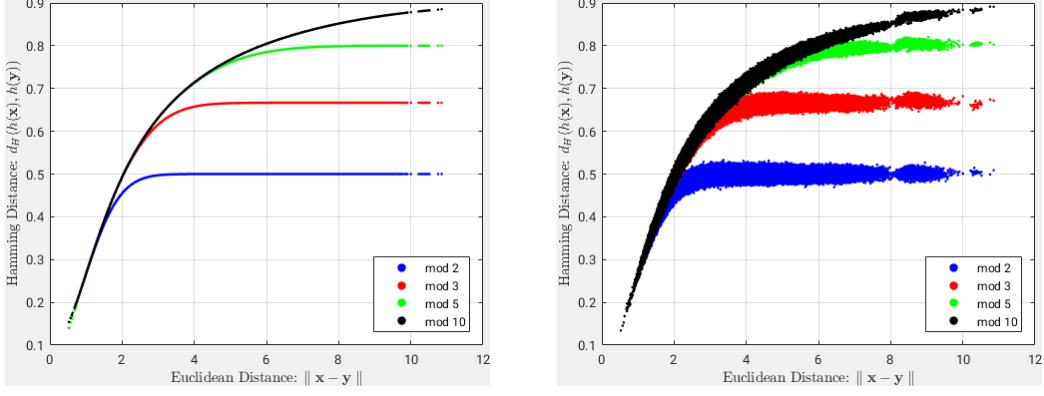
### 3.4.3.2 Distance

The expected value of the hamming distance  $E[d_H(h(\mathbf{x}), h(\mathbf{y}))]$  as a function of  $\|\mathbf{x} - \mathbf{y}\|$  is given by [2]

$$1 - P(h(\mathbf{x}) = h(\mathbf{y})), \quad (3.7)$$

since the hamming distance between two hashes is defined as the average number of mismatches of a collection of hashes, i.e.,  $P(h(\mathbf{x}) \neq h(\mathbf{y}))$ . The expected hamming distance  $d_H(h(\mathbf{x}), h(\mathbf{y}))$ , as well as the actual one, as a function of euclidean distance  $\|\mathbf{x} - \mathbf{y}\|$  for different values of  $k$  are depicted in Figure 3.5.

Reviewing both plots in Figure 3.5, a relation between leakage and  $k$  is observable. The region in which the hashes reveal information about the true vectors, also known as leaky region, is bounded by a threshold located right before the hamming distance saturates. This privacy threshold has to be empirically set for each value of  $k$ , from a close inspection of Figure 3.5(b). The privacy thresholds for each  $k$  are provided in Table 3.3.



(a) Expected value of the hamming distance  $d_H(h(x), h(y))$  as a function of the euclidean distance  $\|x - y\|$ , derived from Equation 3.7.

(b) Actual value of the hamming distance of  $d_H(h(x), h(y))$  as a function of the euclidean distance  $\|x - y\|$ , derived from the hashed values of the datasets.

**Figure 3.5:** Comparison between expected and actual values of the hamming distance, and euclidean distance for different values of  $k$ ,  $mpc = 64$  and  $\Delta = 3$ . Both derived from the hashed Let's Go datasets used in all other previous experiments.

**Table 3.3:** SMH privacy threshold values for each  $x \bmod k$ , empirically set from plots in Figure 3.5.

$k = 2$	$k = 3$	$k = 5$	$k = 10$
0.475	0.65	0.775	0.875

### 3.4.3.3 Leakage

The leakage is defined as the amount of pairs of hashed vectors that have its hamming distance lower than the privacy threshold, previously set in Table 3.3. In a SBE approach, this leakage was mostly influenced by the scaling factor  $\Delta$ , although  $bpcs$  also slightly influenced the leakage. In an SMH approach, the scaling factor  $\Delta$  and the modulus  $k$  are the parameters that mostly influence the leakage, as shown in Table 3.4.

**Table 3.4:** Leakage results for different parameters of  $\Delta$  and  $k$ , with  $mpc = 32$ .

Leakage	$\Delta = 2$	$\Delta = 2.5$	$\Delta = 3$	$\Delta = 3.5$
$k = 2$	0.15 %	1.38 %	6.07 %	16.25 %
$k = 3$	27.69 %	56.58 %	77.02 %	88.96 %
$k = 5$	75.00 %	91.54 %	97.11 %	99.06 %
$k = 10$	98.78 %	99.86 %	99.98 %	99.99 %

For each individual experiment, a suitable  $mpc$ ,  $\Delta$  and  $k$  must be carefully chosen in order to provide maximum privacy and accuracy. From an empirical point of view, the best parameters for good secure classifier would be  $mpc = 32$ ,  $\Delta = 2$  and  $k = 3$ , since there is only 27.69% of leakage, which, not only allows for a good secure scheme, but also for fairly accurate classification results with little degradation.

### 3.4.3.4 Training and Classification

The accuracy results of a private SVM classifier based on an SMH approach are strongly related with the amount of information leaked by the SMH hashes. Thus, it is fair to assume that, using the parameters from Table 3.4 the classification will be better as the leakage increases. Table 3.5 presents the classification results for the different parameters used in Table 3.4 with optimized  $C$  and  $\gamma$  parameters for training phase of the SVM classifier using the modified RBF kernel in Equation 3.4.

**Table 3.5:** Accuracy results for different parameters of  $\Delta$  and  $k$ , with  $mpc = 32$ .

Accuracy	$\Delta = 2$	$\Delta = 2.5$	$\Delta = 3$	$\Delta = 3.5$
$k = 2$	—	—	79.52 %	80.21 %
$k = 3$	80.56 %	81.58 %	81.75 %	82.80 %
$k = 5$	82.27 %	82.44 %	81.93 %	82.79 %
$k = 10$	82.61 %	82.61 %	82.61 %	82.79 %

As expected, for greater leakage values, the classification results tend to approximate the non-private *eGeMAPS* baseline results, 82.79%, computed back in Table 3.1. Note that, for small leakages (less than 5%), the classifier is not able to classify any of the utterances, since the amount of hashes that provide any information about the real distance of the feature vectors is insignificant to allow for the training of an adequate SVM model.

### 3.4.4 Discussion

The introduction of the parameter  $k$  allows for a better control of the leakage provided by any two hashed vectors. From Figures 3.4(b) and 3.5(b), it is possible to observe the importance of this new parameter  $k$ . It shows that  $P(h(\mathbf{x}) = h(\mathbf{y}))$  decreases exponentially with  $\|\mathbf{x} - \mathbf{y}\|$  and saturates in  $1/k$ . Furthermore, the hamming distance  $d_H(h(\mathbf{x}), h(\mathbf{y}))$  is a reliable predictor for the euclidean distance for small values of  $\|\mathbf{x} - \mathbf{y}\|$ . At larger values, however, it rapidly saturates in  $1 - 1/k$ , at which point it becomes uninformative [2]. Thus, an increasing  $k$  will expand the leaky region, providing better accuracy results but at the cost of privacy, which decreases with the amount of information leaked.

From both Tables 3.4 and 3.5, we note not only the relation between  $k$  and the leakage, which is exponentially proportional, but once again, the relation between leakage and classifier accuracy. This accuracy significantly increases with the information revealed by the hashes. A careful selection of  $mpc$ ,  $\Delta$  and  $k$  is essential to achieve the best possible classification results while still providing a strong but efficient security of the data in question. From the experiments performed in Tables 3.4 and 3.5, a good secure classifier would be one with  $mpc = 32$ ,  $\Delta = 2$  and  $k = 3$ , since only 27.69% of the hashes leak information about  $\|\mathbf{x} - \mathbf{y}\|$ , which is sufficient for adequate security, and provides 80.56% accuracy, which presents approximately 2% degradation from the *eGeMAPS* baseline of 82.79%, which is an acceptable degradation for an implementation of a privacy-preserving scheme.

### 3.5 Privacy Analysis

As it was mentioned in Section 2.2.3.2, the main source of security of the SBE/SMH approach relies on the fact that the matrix  $\mathbf{A}$  and the dither  $\mathbf{w}$  are randomly generated by the owner of the data and are completely private and exclusive to him. Hence, a potential impostor or eavesdropper will not be able to learn anything about the true distance of the feature vectors and therefore, no information regarding the biometric aspects of the recorded individual's voice can be obtained.

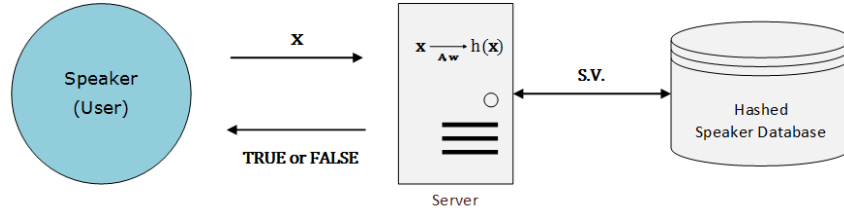
In case the impostor is able to, somehow, get a hold of the private keys  $\mathbf{A}$  and  $\mathbf{w}$ , he still has to compute  $\mathbf{x} = (\Delta \cdot h(\mathbf{x}) + \mathbf{w})\mathbf{A}^{-1} \bmod k$  and, although there is no actual proof that  $\mathbf{A}$  is non-invertible, it would still be too computationally demanding to invert it, unless the attacker has some prior knowledge about the true feature vector and uses it as a starter point [13], specially if the bpc is large, which generates a much larger matrix  $\mathbf{A}$ . Moreover, there is no actual need for the attacker to compute the modulo, he/she may compute a pseudo inverse of  $\mathbf{A}$  or even just use a part of  $\mathbf{A}$  and  $\mathbf{w}$  to get a fraction of the information contained in the feature vector. Nonetheless, the user should always safeguard the private keys  $\mathbf{A}$  and  $\mathbf{w}$ , since they may need to be reused for other machine learning tasks. Let us assume that the user trains a model using hashes generated with a certain set of private keys  $\mathbf{A}$  and  $\mathbf{w}$ . In order for a posterior prediction of a test set to be correctly computed, the user has to generate SBE/SMH hashes for that test set with the same private keys used to train the model.

The SBE/SMH approach has a very simple but strong security. Still, hashes are based on the idea the information about the true features is leaked, which raises the concerns that an eavesdropper might be able to use this leaked information to get useful information about the recording. As it was mentioned several times before,  $\Delta$  and  $k$ , in case of SMH scheme, are the main parameters that control the amount of leaked information. To preserve privacy, the threshold for leakage in which SBE/SMH hashes start to be not secure was empirically set at 0.475 in terms of hamming distance, which means that all hashes with hamming distance superior to 0.475 will not leak any information. [13]. The same idea was applied to the SMH scheme and the privacy thresholds for the different  $k$  can be inspected in Table 3.3.

In summary, an SBE/SMH approach offers a very primitive and strong security while still allowing operations to be computed in the encrypted domain, i.e., it is ideal for privacy-preserving machine learning tasks due to the fact that models can be trained using SBE/SMH hashes in a very trivial way. There is a trade-off between privacy and accuracy, although for these techniques this trade-off is not very harsh. Although its security might be sufficient in the majority of cases, it is not perfect as a motivated adversary might be able to obtain valuable information from it [13].

### 3.6 Case Example

There is little to no point in using an SBE or SMH approach to perform emotion recognition on secured data unless there is an actual real case scenario in which its implementation is useful. The



**Figure 3.6:** Protocol of a Privacy-Preserving Speaker Verification using an SBE approach. A user sends in a recording with his/her voice and receives a positive or negative response for the server.

main objective is to improve a training model by jointly computing it using all available labelled data from different untrustworthy sources.

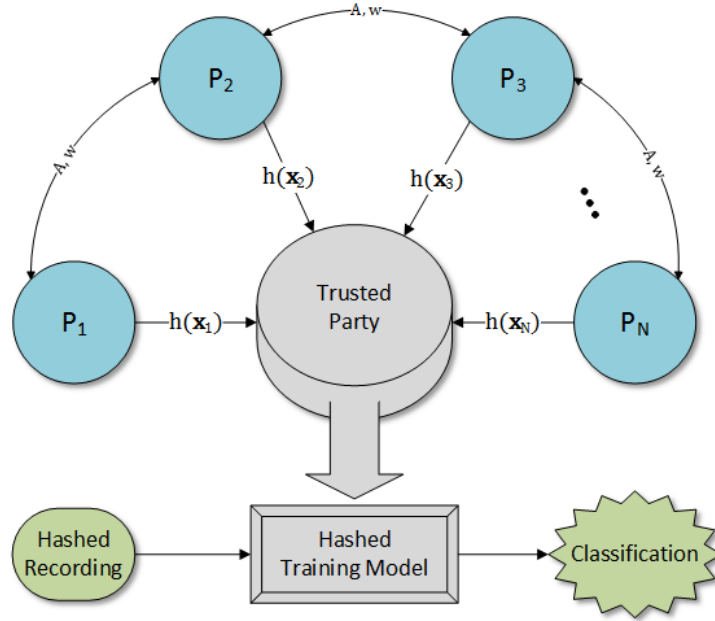
First, let us take into account previous works using an SBE approach to perform private speaker verification [13]. In these works, a simple assignment of a hashed recording to a database, with several recordings from different speakers, is conducted. The protocol for privacy-preserving speaker verification is as follows:

1. A user interacts with a trusted server by speaking into a microphone, allowing the server to have access to the biometric content of his/her voice.
2. The server generates SBE hashes with its, previously assigned, private keys,  $A$  and  $w$ .
3. The server performs speaker verification by comparing the hashed input with its hashed database and returns TRUE if it finds a match or FALSE, otherwise.

Note that all hashed vectors in the database have to be hashed with the same private keys as the ones used to hash the input recording, otherwise, the speaker recognition will fail due to the incompatibility between hashes. This protocol is illustrated in Figure 3.6.

This protocol is adequate for a speaker verification task, since the biometric content of each speaker needs to be private and, thus, it is hashed and stored in a database. An imposter will not be able to learn anything if an attack on the database is made. Moreover, if, somehow, the imposter, by attacking the server, is able to obtain the secret keys, he/she later needed to perform another attack on the database and then decrypt it, which is also very computationally challenging, even knowing the secret keys. Having this in mind, it is possible to conclude that this privacy-preserving protocol has, in fact, a fairly good security.

An emotion recognition case is quite different from the speaker verification one. If we were to provide a two party protocol, similar to the one in Figure 3.6, both secret keys needed to be shared amongst both parties, in order for both inputs to be compatible. Then, either one of the parties would receive the total amount of hashes to create the joint training model. This would lead to a major privacy issue, since the party that received the hashes is also in possession of the secret keys needed to decrypt the hashes from the other party. Hence, due to these facts, a two-party implementation using SBE/SMH is quite challenging and will most definitely lead to serious privacy issues.



**Figure 3.7:** Protocol of a Multiparty Privacy-Preserving Speech Emotion Recognition using SBE/SMH approach for training model improvement. Secret Keys  $A$  and  $w$  are shared between all data providers. The trusted party receives only the hashed data and trains a joint model.

To solve this, the introduction of a Trusted Outsourcing Party (TOP) is required. By adding a trusted party, the private keys and the corresponding hashes can be separated in a way that it is impossible for each party to decrypt other parties data. The protocol for privacy-preserving emotion recognition is, then, as follows:

1. Both parties, let us call them  $P_1$  and  $P_2$ , agree on the secret keys,  $A$  and  $w$ .
2. Each party generates the hashes for their own data,  $h(x_1)$  and  $h(x_2)$  and sends them to the TOP.
3. The TOP builds the training model from the labelled hashed data from both parties. The TOP does not have access to the secret keys. These are only shared between the data provider parties.
4. Once the model is trained, either party may send a hashed input, hashed with the previously agreed private keys, to the TOP, in order to be classified.

A multiparty version for the privacy-preserving emotion recognition protocol is illustrated in Figure 3.7. This example not only proves the possibility of encrypted testing using SBE/SMH hashes, but also allows for a Secure Joint Learning (SJL), where data provided from different parties is jointly used in devising a more accurate training model.

### Privacy Analysis

The privacy for the said protocol relies mainly on the fact that no party may have both the secret keys and the hashed data from other parties. Each individual party has only access to their own hashed data

and the secret keys. The TOP has access to all hashed data but not the secret keys. This means that a party will not be able to learn anything about the other parties data since it does not have access to that hashed data. Similarly, even if the TOP is untrustworthy, it would also not be able to decrypt the data because it does not have access to the private keys. Furthermore, it is quite difficult for the TOP to use this data to create other models, since the hashed data will only have labels corresponding to the specific task. Thus, no single party is able to obtain any knowledge about other parties data and the protocol is, therefore, secure.

With this, and in the scope of emotion recognition in callcenters, a quality control study can be performed among different callcenters. For instance, the National Institute of Statistics, acting as the TOP, wishes to perform a quality control study among the main callcenters in a specific country that are unwilling to share real customer information. This privacy-preserving emotion recognition protocol using SBE or SMH may be implemented to improve an emotion recognizer's accuracy. The TOP is, then, able to obtain a much more reliable quality control for all callcenters.

### 3.7 Summary

In this chapter, SBE/SMH hashes were used to hide and secure voiced audio data while being able to perform basic emotion classification on the hidden data. The privacy-preserving results obtained using SBE/SMH hashes and SVM-based classifiers showed little to no degradation, around 0.2% for SBE and 2.2% for SMH, compared to the non-secure baseline emotion classifier, which shows how promising SBE/SMH hashes are regarding privacy-preserving speech processing tasks. Besides, these models allow for secured joint learning among several untrustworthy parties. Its security is sufficient but not perfect, since the non-invertibility of  $\mathbf{A}$  is not yet proven. The introduction of the  $k$  parameter allows for a better control of the leakage. However, this parameter might prove useful and help to increase security in some other scenarios. Since SBE/SMH is based on hamming distance between hashes, other more sophisticated techniques for emotion classification such as end-to-end Deep Convolutional Recurrent Networks [14] that do not require any signal processing to classify emotions will be harder to implement using an SBE/SMH approach. Nonetheless, SBE/SMH is a new and original way to perform speech processing tasks indirectly on the parameters sets without actually having access to them.



# 4

## Privacy-Preserving Emotion Classifier Using Fully Homomorphic Encryption

### Contents

---

4.1	Introduction . . . . .	51
4.2	Baseline of Emotion Classifier Using ANNs . . . . .	51
4.3	Emotion Classification with Cryptonets . . . . .	54
4.4	Privacy Analysis . . . . .	64
4.5	Case Example . . . . .	64
4.6	Summary . . . . .	66

---



This chapter contains the implementation and testing of a FHE-based neural network on a speech emotional database. Section 4.1 introduces the topic and problem at hand. Section 4.2 presents the raw implementation and testing of a simple ANN (MLP), to be used as a baseline for future encrypted tests. A fully functional speech-based emotional cryptonet is implemented and tested in Section 4.3. An analysis of the level of privacy of the cryptonet is showed in Section 4.4 and, a real world scenario example for the use of this cryptonet is illustrated and outlined in Section 4.5. Finally, a brief summary about the main conclusions regarding the chapter is present in Section 4.6.

## 4.1 Introduction

The amount of data that can be stored is always limited by storing units, such as servers, hard drives, disks, among other. With the ever increasing technological market and due to new advances in this area, storing a tremendous amount of data is now possible. This allowed the creation of enormous datasets that may be applied as training for one of the biggest trends in machine learning at the moment, ANNs. The sudden interest in constructing ANN models is directly correlated with the amount of data accessible, since these can only surpass other machine learning techniques if the amount of training data available is suitable.

Although the training data used by the model is public, test data held by an eventual client may not be. A certain client might not want to have his or her or someone's sensitive information accessed by an unknown party. Thus, the need for private and secure evaluation and predicting on neural networks becomes a necessity. Some different privacy-preserving techniques using deep learning and neural networks have been researched, such as deep learning with differential privacy [58] and ANNs using Fully Homomorphic Encryption (FHE) [3, 40].

## 4.2 Baseline of Emotion Classifier Using ANNs

Inspired on the real biological neural networks of the brain, ANNs consist of interconnected units, also known as artificial neurons, that communicate with each other to process information. An ANN learns by first doing a feedforward sweep and, when it reaches the output, it checks the error between the prediction from the feedforward sweep and the actual output it should return. It then performs a backpropagation sweep, optimizing the weights and bias of each unit.

For emotion classification experiments, an MLP was considered for its simplicity and efficiency. An MLP uses hidden layers of many single perceptrons with non-linear activation functions, such as the sigmoid,  $f(x) = (1 + e^{-x})^{-1}$  or the ReLU,  $f(x) = \max(0, x)$ , among others. In an MLP, the output of a perceptron is  $f(\sum_i w_i x_i + b_i)$ , where  $f$  is the non-linear activation function,  $w$  is the weight and  $b$  is the bias of the  $i$ (th) neuron. The MLP consists of an input layer, one or more hidden layers and one output layer. Each node is connected to all nodes in the previous layer. The input layer contains as much nodes

as there are features, while the output layer contains as much nodes as there as different class labels. The prediction is given by the node index with maximum output.

### 4.2.1 Experimental Setup

The corpus considered for all experiments in this chapter is also the LEGOV2.1 corpus [5, 63, 64], which is an updated version of the original "Let's Go" Corpus [63] with emotional labels provided by an expert rater. It consists of three sets of data: training, validation and test set. The dimensions of these sets are 3005 recordings for training, 657 for validation and 581 for testing. The outputs for these experiments will be either *angry* or *neutral*. More information about this corpus is presented in the Appendix A. All experiments in this chapter were performed in a MacBook Pro, 2.7Ghz Intel Core i5 with 8Gb RAM DDR3.

The architecture of the ANN baseline follows the common MLP architecture with the following modules:

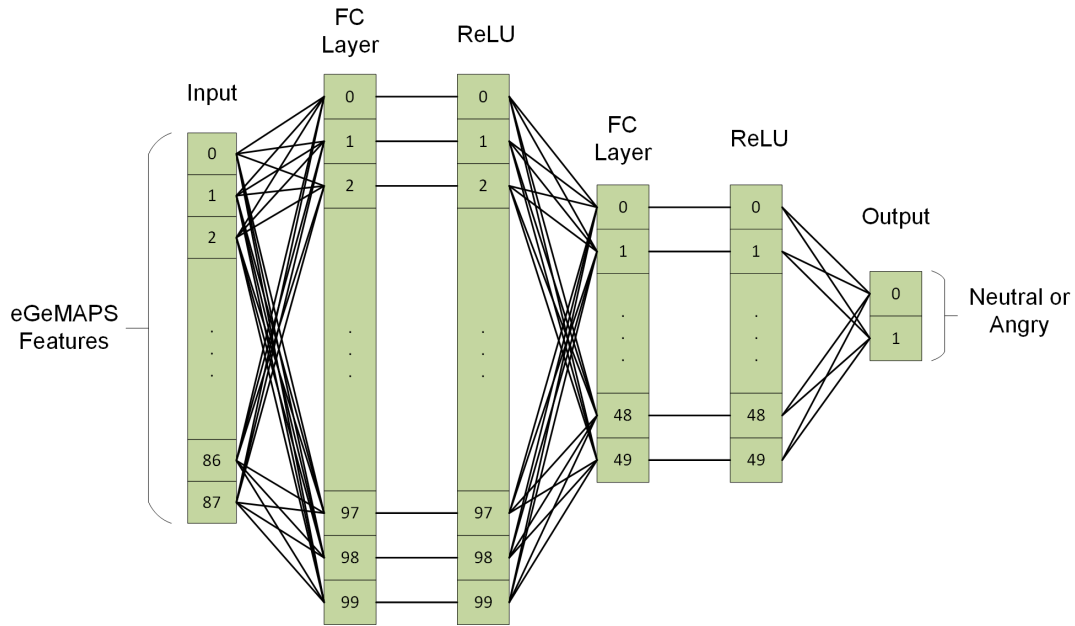
- *Input*: 88, previously processed and extracted, *eGeMAPS* features.
- *First Internal Layer*: Fully Connected layer with 100 hidden units, where each node is connected to all the nodes of the input layer.
- *Second Internal Layer*: Fully Connected layer with 50 hidden units, where each node is connected to all the nodes in the first internal layer.
- *Output*: 2 nodes, corresponding to the emotional labels, which is either 0 (neutral) or 1 (angry).

A ReLU activation function is present at the end of each fully connected layer. Before the fully connected layer generates the outputs that will be the inputs of the next layer, these need to pass through this activation function. The learning algorithm used was the gradient descent with a learning rate of 0.01 and a weight decay of 0. The loss function is the mean-squared error.

The whole implementation was done using C++ programming language. A visual representation of the architecture is shown in Figure 4.1.

### 4.2.2 Training and Classification

The training and classification of an MLP is pretty straightforward. The *eGeMAPS* features are fed to the input layer while the labels of the recordings are positioned in the output layer. In a training phase, the features will undergo a feedforward sweep of the whole network with some initial weights and biases. Once at the output layer, the mean squared error is calculated between the predicted and the actual labels. Afterwards, the weights and biases will be updated in a process called backpropagation, that evaluates the error of each unit, thus completing the training process for that input. This network uses the gradient descent learning algorithm to perform a backpropagation of the errors, updating and optimizing



**Figure 4.1:** Multilayer Perceptron architecture used in the baseline experiments.

**Table 4.1:** Test results of the MLP baseline for both validation and test set.

Accuracy	Validation Set	Test Set
Baseline	88.5845 %	82.6162 %

the weights and biases. The training takes approximately 0.274 seconds for 3005 recordings. It is important to notice that the training of a neural network with such a small training dataset is dangerous because it may lead to overfitting, which may cause poor results for different test sets. Having this in mind, we trained the network with 3005 training recordings and tested the network on a validation set. Small changes were made to the network in order to maximize the accuracy score on the validation set. Later, we used the modified network to retrain with the training data, and obtained 82.6162% accuracy in the test set. The results for both the validation set and test set can be inspected in Table 4.1.

### 4.2.3 Discussion

By inspecting the results in Table 4.1, it is possible to notice the similarity between these MLP baseline results and the SVM baseline results, obtained in Section 3.2. These baseline results can be considered good for testing with such an small amount of training data. It is, in fact, possible to boost the accuracy scores of the ANN, by using different methods and techniques such as, increasing the complexity of the network, usage of optimizers, like the Adam Optimizer, after each epoch of training and testing, increasing the training data, number of hidden layers and number of hidden units, and optimizing learning parameters. However, the main aim of these experiments was not to optimize the baseline results, but show that the encrypted model accuracy scores have the minimum degradation possible, when comparing with results from the baseline.

Overfitting is not a major issue, since the whole labelled data was previously divided into 3 smaller datasets. One for training, another for evaluation and validation, and the last one for actual testing. This effectively prevents overfitting due to the fact that all these smaller datasets contain independent data from one another.

## 4.3 Emotion Classification with Cryptonets

Researchers from different areas, having the need to follow the ANN and deep learning trend, have successfully studied and implemented privacy-preserving methods that may be applied to these new state-of-the-art machine learning techniques. Very recent examples are: deep learning with differential privacy [58], and ANNs using Fully Homomorphic Encryption (FHE) [3, 40]. SEAL [4], written in C++, is the FHE library used in all encrypted-based experiments in this chapter.

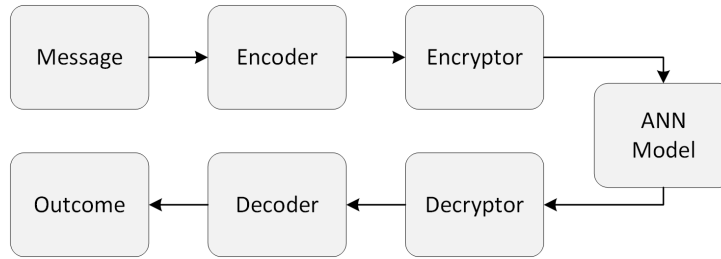
In this section, we modify the previously build MLP, transforming it into a cryptonet, which allows for encrypted testing on a learned model, i.e., allows for a client to encrypt processed speech, using FHE, and successfully predict the emotion of the subject without any privacy loss. Still, the combination of an ANN and a FHE scheme is not trivial. Since the FHE, described in more detail in Section 2.2, only allows for multiplications and additions to be preserved, some modifications and approximations need to be applied to the original network. Even though most operations in a simple MLP are multiplication and addition-based and are easily modified by switching typical multiplications and additions with FHE-based ones, some aspects, such as the non-linear activation function, need to be adapted to work with only FHE-friendly operations.

### 4.3.1 Cryptographic Neural Networks

Cryptographic Neural Networks, or simply cryptonets [3, 40], were first introduced by Microsoft researchers as a privacy-preserving framework for neural networks. The main idea of cryptonets is to unite the powerful deep learning models with a structure-preserving encryption scheme, such as FHE, which allows the manipulation of plaintexts through homomorphic operations on their corresponding ciphertexts. Thus, the encrypted inputs of a neural network can be easily manipulated within the neural network environment, providing encrypted predictions that may only be deciphered by the client or someone that holds the private key.

### 4.3.2 Simple Encrypted Arithmetic Library

The Simple Encrypted Arithmetic Library [4], or just SEAL, is an easy-to-use FHE library that not only contains the essential FHE multiplications and additions, but also some very useful tools to help optimize the encryption, security and speed of computations.



**Figure 4.2:** SEALs encoding and encryption process.

The encryption and decryption process is somehow different from what one should expect. Since SEAL is a polynomial-based FHE library, there is a need to first encode the actual message one wishes to encrypt into a polynomial, and then actually encrypt the polynomial. The decryption process is similar. When the encrypted message is decrypted, the result is, as expected, a polynomial. This polynomial needs to be, later, decoded into an actual number. SEAL contains implementations of the most common encoding/decoding schemes. The encryption/encoding process along a machine learning model is illustrated on Figure 4.2.

Four types of encoders are available to the user, namely a scalar encoder, an integer encoder, a fractional encoder and a CRT batching encoder. The fractional encoder seems to be the most logical encoder for our purposes, since we will most surely be working with floating point numbers.

The encryption parameters chosen are directly linked with the security and speed of the homomorphic computations. Moreover, as soon as a message is encrypted, there is a noise budget that decreases with each homomorphic operation performed on the ciphertext. If a homomorphic computation is performed while the budget is zero, the result becomes undecryptable, i.e., will decrypt the message into a completely unrelated number. If needed, SEAL has an automatic parameter selection function, that helps choosing the best and most viable encryption and encoding parameters for the computation of a given function. This will, in most cases, result in the best compromise between security, speed and noise budget. On a side note, after choosing the encryption parameters, both the secret key and the public key may be generated. These, along with the encryption parameters, will serve as inputs for the encoder, encryptor and decryptor. The secret key is mainly used in encrypting and decrypting while the public key is used for evaluation, i.e., homomorphic operations.

There are several homomorphic operations allowed by SEAL. The most common are the homomorphic multiplication and addition with either another ciphertext or a plaintext. Nonetheless, several other useful operations are available, such as the negation of a ciphertext, square, exponentiate, subtract, among others. More information about SEAL can be found in Appendix C.

### 4.3.3 Approximations and Modifications

#### 4.3.3.1 Polynomial Activation Function

A network with only fully connected layers and convolutional layers is only able to classify data linearly, thus, the activation function of a neural network is essential because it allows the neural network to classify data in a non-linear way. The most common activation functions used are the sigmoid,  $f(x) = (1 + e^{-x})^{-1}$  or the ReLU,  $f(x) = \max(0, x)$ , as it was previously mentioned.

Since, in the baseline, we used the ReLU activation function, a polynomial approximation of this function is in order. Note that, there is a trade-off between the polynomial degree of the approximation, i.e., how accurate the approximation is, and the speed of the FHE operations. Moreover, the encrypted calculation of a high degree polynomial is not only slow, but also increases the encrypted noise exponentially and, if the noise is greater than a certain noise budget, defined by the encryption parameters, then the signal is too noisy to be correctly decrypted, resulting in entirely wrong and uncorrelated decryptions.

Using the Python programming language with the *polyfit* function of the package *numpy*, polynomial approximations with different polynomial degrees may be computed. The analytical functions can be inspected in Table 4.2, while a visual representation of the same approximation is depicted in Figure 4.3.

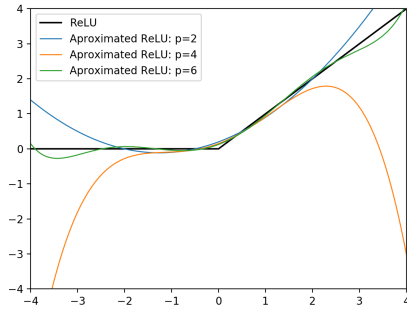
From Figure 4.3, it is possible to see the relationship between the degree of the polynomial and how good the approximated polynomial function is. The polynomial approximation of the ReLU function becomes better as the degree of the polynomial increases. Furthermore, it is also possible to realize that the approximation is only effective in approximately  $[-2, 2]$  and, outside this interval, the approximated polynomial deviates exponentially from the actual ReLU function. In order to solve this, either all the inputs that enter the activation function are already situated within this threshold interval, or a normalization of the activation function inputs is required.

#### 4.3.3.2 Pre-Activation Normalization

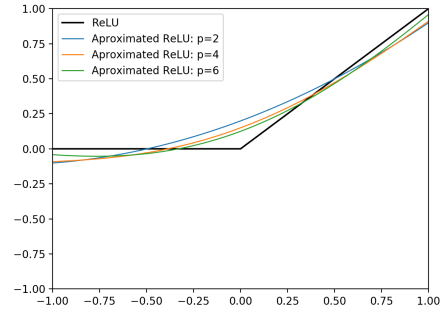
Normalization is, in this case, the mapping of the real values to another domain, usually a space interval previously chosen by the user. There are several ways to perform the normalization of a vector  $\mathbf{v} = [x_1, x_2, \dots, x_n]$ . The easiest and most trivial method is to swipe the entire vector to find the maximum and the minimum and, for each element  $x_i$  of the vector  $\mathbf{v}$ , compute

**Table 4.2:** Approximation of the ReLU function by polynomials.  $p$  represents the polynomial degree of the approximation and the approximated polynomial is the approximation of the ReLU by a polynomial [3].

$p$	Approximated Polynomial
2	$0.1992 + 0.5002x + 0.1997x^2$
4	$0.1500 + 0.5012x + 0.2981x^2 - 0.0004x^3 - 0.0388x^4$
6	$0.1249 + 0.5000x + 0.3729x^2 - 0.0410x^4 + 0.0016x^6$



(a) General approximations to the ReLU activation function with different degree polynomials.



(b) Detailed approximations, within a normalization environment between -1 and 1, to the ReLU activation function with different degree polynomials.

**Figure 4.3:** Illustration of the polynomial approximations to the ReLU using the analytical functions from Table 4.2.

$$\hat{x}_i = \frac{x_i - \min(\mathbf{v})}{\max(\mathbf{v}) - \min(\mathbf{v})}. \quad (4.1)$$

If the need arises, one can compute  $\hat{x}_i = (b-a)\hat{x}_i + a$  to normalize the values to an interval  $[a, b]$ . Here  $\hat{x}_i$  is the normalized element  $i$  of the vector  $\mathbf{v}$ . However, this method cannot be used in the encrypted domain due to the fact that one needs complete access to the real and unaltered vector in order to find its maximum and minimum. With the normalization occurring directly before the activation function, it is not possible to use this method because the server/cloud does not hold the private key and, therefore, cannot decrypt the vector to find the maximum and minimum.

The server or cloud, being only allowed to perform simple operations in ciphertexts, needs a way to perform normalization using just these simple ciphertext operations. This is achieved with *batch normalization* [65]:

$$\mu = \frac{1}{n} \sum_{i=1}^n x_i, \quad (4.2)$$

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2, \quad (4.3)$$

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}}, \quad (4.4)$$

$$y_i = \gamma \hat{x}_i + \beta, \quad (4.5)$$

where  $n$  is the dimension of  $\mathbf{v}$ ,  $\mu$  is the mean,  $\sigma^2$  is the squared variance and  $\hat{x}_i$  is the normalized element  $i$  of vector  $\mathbf{v}$ .  $\gamma$  and  $\beta$  are the scaling and shifting factors, respectively, which are learned by

the network.  $\epsilon$  is a small value to prevent the division by zero. The *batch normalization* [65] technique is composed of two major steps, namely the learning of the parameters  $\gamma$  and  $\beta$  in the training phase, and, the actual normalization using these parameters in the test phase. In short, the computations done in each phase, in a *batch normalization* layer, are the following:

- *Training Phase:* The mean  $\mu_B$  and variance  $\sigma_B^2$  of each batch is computed using Equations 4.2 and 4.3. Afterwards, the mean and variance of the population ( $\mu_{pop}, \sigma_{pop}^2$ ) is updated based on the current  $\mu_B$  and  $\sigma_B^2$  with some decay. Note that, this  $\mu_{pop}$  and  $\sigma_{pop}^2$  will reflect the mean and variance of the whole training set and is crucial for the testing phase. The inputs are, then, normalized by means of Equations 4.4, where  $\mu$  and  $\sigma^2$  are the updated mean and variance of the population at the time. Finally, the normalized input is scaled and shifted using the parameters  $\gamma$  and  $\beta$  (Equation 4.5) that are being constantly updated by means of backpropagation.
- *Testing Phase:* In a testing phase, the layer is only supposed to access one test data point at a time, hence, computing the mean and variance of the batch is unfeasible. Instead, the mean and variance of the whole population, estimated in the training phase, is used to normalize with Equation 4.4. Afterwards, the normalized test input is scaled and shifted with the already learned parameters  $\gamma$  and  $\beta$ , using Equation 4.5. Notice that, according to the law of large numbers, the mean and variance of the training population, used in the inferring, provides a rather good approximation of the real mean and variance of the input.

Note that, each scalar feature is independently normalized with zero mean and unity variance. It is also important to notice that the simple normalization computed by Equation 4.4 may change what the layer can actually represent. This is the reason why the scale  $\gamma$  and shift  $\beta$  parameters are introduced and learned over training data [65].

All the operations in Equations 4.2, 4.3, 4.4, and 4.5 can be expressed in a FHE-friendly way, through multiplications and additions, thus allowing it to be performed on ciphertexts. The only element that will be encrypted in the normalization will be the input, since the mean  $\mu$ , variance  $\sigma^2$ , scale  $\gamma$  and shift  $\beta$  have all been previously computed in a training phase. Note that, not being compliant with divisions, it is necessary to have an intermediate calculation of divisor in Equation 4.4:

$$div = \frac{1}{\sqrt{\sigma^2 + \epsilon}}. \quad (4.6)$$

With this, it is possible to rearrange the norm computation in Equation 4.4 into

$$Enc(\hat{x}_i) = (Enc(x_i) - \mu).div, \quad (4.7)$$

which is more suitable for FHE operations.

Normalizing the inputs of the activation function will not only force the activation function inputs to be within a previously established and acceptable range that guarantees a fair approximation of the ReLU

function, but also enable higher learning rates and, as a result, help reducing the overall training time. Furthermore, it allows for a regularization of the model [65].

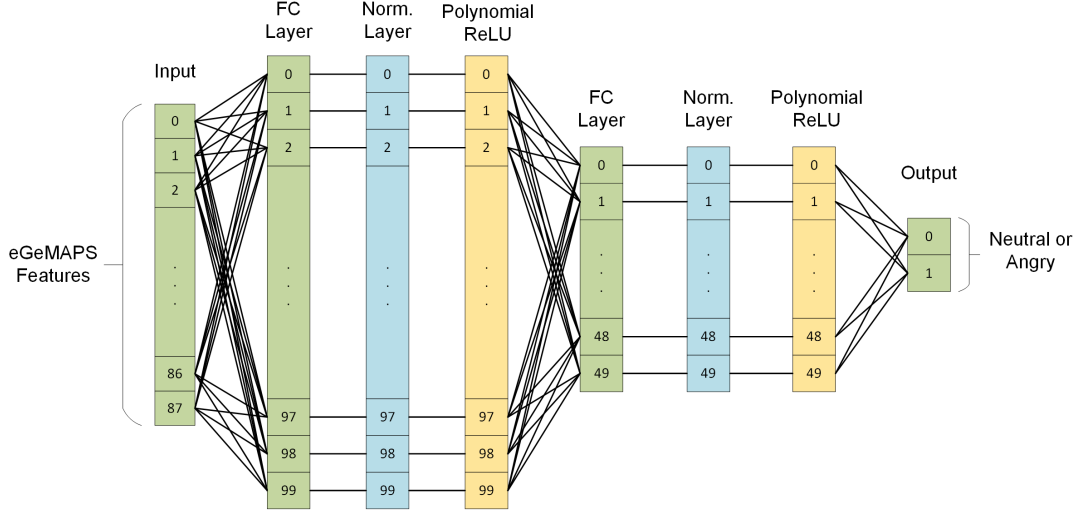
#### 4.3.3.3 Encryption Noise

Noise is a fundamental part of encryption. As soon as a plaintext is encrypted into a ciphertext, it contains a non-zero amount of noise. The noise of a specific ciphertext grows as FHE operations are performed. As a matter of fact, addition and subtraction have a very small impact on noise, while multiplications have a much greater impact [4]. In SEAL, the noise is called *invariant noise*, and it is defined as the quantity that must be "rounded away" correctly for decryption to succeed. With this, the library calculates the *invariant\_noise\_budget* of a ciphertext, which is computed by  $-\log_2(2v)$ , where  $v$  is the amount of invariant noise in the ciphertext, and decreases with every homomorphic operation until it reaches zero. While the budget is greater than zero, the ciphertext may be correctly decrypted, otherwise it becomes undecryptable. [4].

Thus, in this work, one must take into account aspects such as the multiplicative depth of operations and nonessential homomorphic operations, in order not to unnecessarily consume the *invariant noise budget*. There is, however, something to be done to either increase the *invariant\_noise\_budget*, or to decrease the budget cost of certain homomorphic operations, if the need arises. These include, but are not limited to:

- Decreasing *plain\_modulus* parameter in the encryption process. Decreasing this parameter will also decrease the noise budget consumption in homomorphic multiplication. However, this may cause other issues such as the plaintext polynomial wrapping around the plaintext modulus. Nonetheless, this is a risky solution since the encoders used in this thesis are fractional encoders and, it is very difficult to estimate when and if this wrapping happens.
- Increasing homomorphic encryption parameters. Increasing both the *poly\_modulus* and *coeff\_modulus* parameters will effectively increase the amount of noise budget, while also increasing the security of the encryption. Although, this effect will slow down the encryption/decryption process and increase homomorphic operation computational times.

All of these encryption parameters control very important aspects, such as the level of security, the speed of computations, the amount of noise budget, and the number of arithmetic operations, specially multiplications, that can be applied to the ciphertext. There are major trade-offs between these parameters, hence choosing the ideal parameters for a specific task is a rather important step. A more detailed description of the noise, security and SEAL parameters may be found in Appendix C.



**Figure 4.4:** Modified MLP architecture compliant with an FHE scheme.

### 4.3.4 Experimental Setup

The initial step for the implementation of the MLP cryptonet is to choose the ideal encryption parameters so that the *eGeMAPS* features may be encrypted and stored onto a file for later testing. The encryption and encoding parameters chosen for our cryptonet are depicted in Table 4.3. The modified architecture of the baseline network is illustrated in Figure 4.4.

**Table 4.3:** Cryptonets Encoding and Encryption Parameters. The parameter name refers to the name of the parameter as it appears in the *SEAL* library while the value represents the actual numerical value chosen for the experiments. Parameters that are not mentioned, but are still crucial to the experiments, are left as default.

**(a) General Encryption Parameters.**

Parameter Name	Value
<i>poly_modulus</i>	$x^{4096} + 1$
<i>coeff_modulus</i>	4096
<i>plain_modulus</i>	8

**(b) Fractional Encoder Parameters.**

Parameter Name	Value
<i>integer_coeff_count</i>	64
<i>fraction_coeff_count</i>	32
<i>pool</i>	3

The encryption parameters used in the experiments, shown in Table 4.3, have an estimated security level of 119.1 bits [4]. Having all test recordings encrypted onto a static file, the MLP is trained as it would be in the baseline, with unencrypted recordings, and with the same learning parameters.

The difference between unencrypted and encrypted testing revolves, mainly, on two steps, namely a pre-activation normalization and a polynomial approximation of the activation function. The activation function input data is normalized using a batch normalization layer. Furthermore, three polynomial approximations to the ReLU were made with 2, 4 and 6 polynomial degree, all of which were then used for testing. The whole privacy-preserving MLP was implemented using C++, with the *SEAL* library, developed by researchers in the Cryptography Research Group at Microsoft Research [4].

## Implementation Details

The implementation of a fully functional cryptonet is far from trivial. A lot of research was put into devising this classifier, not only in terms of programming and libraries, but also theoretical concepts that were crucial for the implementation.

In a first stage, we implemented the unencrypted model using Python's *TensorFlow* package, since it is the most used open-source library for the implementation of ANNs. Here, we encountered an issue with FHE encryption libraries. Since FHE schemes are still not efficient enough, they are usually implemented using low level programming languages, such as C++, in order to speed up the computations, which generated an incompatibility between Python and C++. It is possible to construct wrappers of Python for C++ libraries, however, this is not straightforward and some of the speed would be lost. To try and keep it simple, and after some research, tried to use HEIDE [66], which is an Integrated Development Environment (IDE) for the HElib [43]. After some issues in the installation, mainly incompatibilities between the IDE and the HElib, the usage of this software proved pointless because HElib does not support operations with floating-point numbers.

In a second stage, we decided to implement the ANN from scratch in C++. This would finally allow us to use FHE libraries, such as HElib [43], YASHE [42] and SEAL [4]. Again, the implementation of these networks in such a low level programming language is far from trivial, specially for someone with little background in programming. We finally opted for the SEAL library since it was the most complete and up to date FHE library, and, most importantly, allowed floating-point operations to be performed on ciphertexts. To have a better sense of the modelled network, we decided to first use it with the MNIST<sup>1</sup> database, which is a hand-written digit database for image recognition, and only then applied it to our emotional database.

Other issues, such as noise growth and encrypted normalization, eventually came up throughout the implementation of this network but were solved after some careful examination of the code and extensive research on the subject.

### 4.3.5 Results

Having adapted the whole network to work with FHE operations, the saved files containing the encryption parameters and the test data were loaded into the model. The network would still be fed 88 features, but now these features are encrypted under a private key. The public key is known to both the owner of the network and the client. Again, this public key is required to perform encrypted evaluations, i.e., homomorphic computations.

The network has been modified to have 2 different feedforward functions. One working with unencrypted data (training phase) and another to work with encrypted data (test phase). Note that the activation function present in the training phase is the actual ReLU function, and not an approxima-

---

<sup>1</sup>MNIST Database available in <http://yann.lecun.com/exdb/mnist/>

tion. The polynomial approximated ReLU function is only used on a testing phase. In order to have a better understanding of what actually degrades the accuracy results, we first perform experiments in the unencrypted test set for different values of  $p$ , with and without normalization. The results for these experiments may be consulted in Table 4.4.

**Table 4.4:** Accuracy results of the tests performed on the unencrypted test set for different values of  $p$ , with and without normalization. The normalization refers to the pre-activation normalization, serving to adjust the inputs to a more viable interval. The  $p$  represents the degree of the polynomial approximation of the ReLU activation function.

Accuracy	$p = 2$	$p = 4$	$p = 6$
Without Normalization	81.0671 %	81.4114 %	81.2392 %
With Normalization	81.5835 %	81.7566 %	81.5835 %

Afterwards, we performed the same experiments with the encrypted test set. The time required to test each encrypted recording is between 65-75 seconds. Keep in mind that every test was made in a MacBook Pro, 2.7Ghz Intel Core i5, and 8Gb RAM DDR3, so, it is safe to say that a computer with more computational power would be able to speed up the testing phase significantly. In any case, the accuracy results of the cryptonet for the test set, for different polynomial degrees of the approximation, with and without pre-activation normalization are present in Table 4.5.

**Table 4.5:** Accuracy results of the tests performed on the encrypted test set for different values of  $p$ , with and without normalization. The normalization refers to the pre-activation normalization, serving to adjust the inputs to a more viable interval. The  $p$  represents the degree of the polynomial approximation of the ReLU activation function.

Accuracy	$p = 2$	$p = 4$	$p = 6$
Without Normalization	59.7246 %	81.5835 %	81.4114 %
With Normalization	79.1738 %	80.5508 %	80.2065 %

### 4.3.6 Discussion

There are many factors in this cryptonet approach that may influence the accuracy of the model. The noise from homomorphic operations can, on the face of uncertainty, shift the prediction from *angry* to *neutral*, or the other way around, specially in the activation function, due to the presence of exponential operations with ciphertexts, which are the homomorphic operations that develop most noise. On the other hand, the approximation of the ReLU function is directly correlated with the accuracy scores, because on a testing phase, the values presented on the approximated activation function need to be as close as possible to the values in the actual activation function of the training phase. Thus, a better approximation leads to better accuracy. This is seen in the experiments performed on unencrypted data, with results shown in Table 4.4, where there is, approximately, 1% degradation, in comparison to the baseline, which means that the introduction of the approximated ReLU function decreases the model accuracy by approximately 1%.

The introduction of the normalization layer slightly increases the accuracy of the model for all polynomial degrees. This is due to a small percentage of the activation input values being outside the reliable approximation interval. As mentioned, the approximated activation function deviates exponentially from the true ReLU outside the reliable interval, and thus, a value that is just slightly outside this interval increases the chance to make a wrong prediction. Another explanation for this accuracy increase is that inside the normalization interval, the approximation is better. For instance, inspecting the Figure 4.3(a) and 4.3(b), we can conclude that, in case the normalization layer mapped the activation function entries to an interval  $[-1, 1]$ , the approximation would be slightly better than if there was no normalization. However, the smaller the normalization interval, the smaller the precision the activation function has.

The results shown in Table 4.4 with and without normalization are worth discussing. With normalization, one obtains a degradation of approximately 2%, when comparing with baseline results. Some of this degradation may be attributed to factors already mentioned such as noise and approximations. Even so, a 2% degradation in accuracy is an acceptable cost to ensure the privacy-preservation of the clients data. As mentioned in the last paragraph, 1% of this degradation is provided by the approximation of the activation function. Thus, the other 1% of degradation is mainly based on the noise provided by homomorphic encryptions.

When the inputs of the activation are not normalized, two observations may be made. For  $p = 2$ , the accuracy significantly decreases (about 20%), whereas for other degrees, it actually increases about 1%. The main reason for the degradation in  $p = 2$  may be attributed to activation inputs falling outside the reliable interval of approximation, as mentioned in previous paragraphs. In an unencrypted domain, this effect is not very noticeable, however, in an encrypted domain, with the presence of noise provided by homomorphic operations, the effect may be enhanced, pushing some values even further from the acceptable range. This deviation increases exponentially, resulting in poor accuracies. Nonetheless, for  $p = 4$  and  $p = 6$ , the accuracy is similar to its unencrypted counterparts, slightly deviating due to noise. Comparing the encrypted results with and without normalization, there is about 1% degradation from a non-normalization approach to a normalization approach. This is, again, due to homomorphic noise and a loss of precision due to normalization. Even though the accuracy generally decreases, the normalization counters the poor accuracy of the  $p = 2$  without normalization, because it successfully maps the activation inputs to a more suitable interval. Moreover, the features used in these experiments were previously pre-processed and scaled to an interval  $[-1, 1]$ . This means that, even without normalization, the encrypted model can still be rather accurate. This may not happen with other unprocessed inputs, which surely would result in incredibly poor accuracy scores due to too many activation inputs falling outside the acceptable approximation interval. Thus, the normalization layer helps not only to increase the accuracy for smaller polynomial degrees, but also to generalize the model to other inputs.

Finally, the fact that this classification task is binary, i.e., the model only classifies either one of two classes, also increases the probability that, on the face of uncertainty, a previously wrong prediction can turn into a correct prediction and vice-versa, by means of noise or approximations. Anyhow, it is very

**Table 4.6:** Default pairs  $(n, q)$  and their estimated security levels [4].

$n$	$q$	Security Estimate (bits)
2048	$2^{60} - 2^{14} + 1$	115.1
4096	$2^{116} - 2^{18} + 1$	119.1
8192	$2^{226} - 2^{26} + 1$	123.1
16384	$2^{435} - 2^{33} + 1$	130.5
32768	$2^{889} - 2^{54} - 2^{53} - 2^{52} + 1$	127.7

difficult to tell if this will be beneficial or prejudicial. However, slightly better or worse accuracy scores might be caused by this effect.

## 4.4 Privacy Analysis

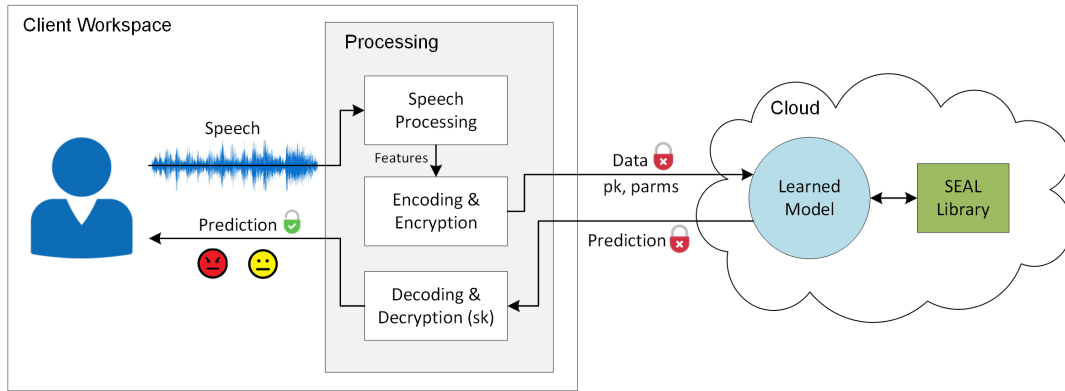
The security of this approach is based on the security of the Fan-Vercauteren (FV) homomorphic encryption scheme [67], which is the scheme used by the SEAL library with some small changes on some operations. The security of the FV encryption scheme is based on the apparent hardness of the Ring Learning with Errors (RLWE) problem [68]. SEAL only allows for polynomial moduli to be of the form  $x^n + 1$ ,  $n$  being a power of 2 because polynomial moduli of this form not only provides security but also increases the performance of the encryption. To break security, an attacker would need to solve the d-sample decision-RLWE problem, which is extremely hard.

Determining concrete hardness of parametrizations of RLWE is an active field of research and no standardized RLWE parameter sets exist. The security estimates for the default parameters in Table 4.6 represent the security guarantees, though these are not definite [4].

In the textbook FV scheme, an evaluation key is required, which is essentially a masking of the secret key raised to the power of 2. Here, to argue security, one needs to assume that the encryption scheme is secure even when the adversary has access to all of the evaluation keys. In SEAL, no relinearization is done by default, thus, if the user chooses not to do relinearization, there is no need to generate evaluation keys which allows SEAL to avoid the previous assumption. Even if a user chooses to perform relinearization, there are no practical attacks using evaluation keys, at the time of this thesis [4].

## 4.5 Case Example

The implementation of the fully homomorphic encryption scheme in neural networks maps a path to Secure Machine Learning as a Service (SMLaaS). MLaaS has been growing in popularity in recent years, and this sudden popularity has raised some major privacy concerns. In the speech scope, one can uncover a great deal of information about someone from their speech, and therefore, performing privacy-preserving machine learning in speech becomes a dire necessity in today's world.



**Figure 4.5:** Case example of a client using a cloud-based application to predict the emotion label of a given recording.

The most useful usage of cryptonets is in MLaaS, specifically cloud-based applications. Let us assume that some user (Client), either individual or corporate, wishes to use the most accurate ANN prediction model in the market, provided as a cloud application (Cloud), to predict the outcomes of certain speech recordings, in our case, emotions. Obviously, the client does not entrust this sensitive speech data to an unknown party, that may or may not be trustworthy. The owners of the cloud-based application, having their income relying on the amount of predictions their model performs, will not profit from those that do not want to share sensitive information. If, the owners of the ANN modify their network to work with FHE-multiplications and FHE-additions, then, the previously unwilling clients would most surely agree on sharing their sensitive data, so long that it was encrypted. A simple cloud-based MLaaS application protocol using cryptonets is illustrated in Figure 4.5 and is described as follows:

1. The client processes a speech recording, extracting paralinguistic features, such as *eGeMAPS* features.
2. The client encrypts his sensitive features, generating a public key (pk) and a secret key (sk).
3. The client then sends his encrypted data to the cloud, along with the public key and encryption parameters (for eventual encodings and evaluations).
4. The network predicts the outcome of each encrypted input.
5. The network sends these encrypted predictions to the client.
6. The client, holding the secret key, decrypts the predictions, which are in this case either angry or neutral labels.

The public key is known by all parties involved and is mostly used to perform homomorphic operations within the network. The secret key is held by the client and it is private to him. It is solely used for encrypting and decrypting data. It is important to note that the predictions provided by the network are encrypted under the secret key held by the client. These encrypted predictions hold no real value to the cloud and cannot be used to infer anything about the actual inputs of the network.

## 4.6 Summary

In this chapter, we showed that it is possible to devise privacy-preserving neural network schemes for speech signals using fully homomorphic encryption. The main approach is to adapt the network to only perform multiplications and additions, so these can be later replaced with FHE-based multiplications and additions. However, the implementation is not trivial, and one needs to care for imminent issues, such as noise budget, poor approximations, security and computational times. Polynomial approximations of the ReLU activation function were made for 2, 4 and 6 polynomial degree, and some optimizations such as normalization and parameter selection were made.

Having made several experiments with *eGeMAPS* features on the LEGO database, we achieved little degradation on the privacy-preserving cryptonet, in the order of 1%, for higher degree polynomials. By applying a normalization on the polynomial ReLU inputs, we were able to effectively restrict the activation inputs to a more viable interval. With this, we managed to obtain, approximately, 2%-3% degradation for all polynomial degrees.

In short, with a scaled *eGeMAPS* feature vector, the cryptonets accuracy results is based mostly on how good the approximation of the activation function is. The normalization of the activation inputs effectively generalizes the cryptonet to other databases and provides better accuracy scores for lower polynomial degrees, reducing the multiplication depth and, therefore, the noise growth and complexity of the network. Generally, the results obtained for our privacy-preserving scheme using FHE offer good accuracy results at the cost of some computational time.

# 5

## Conclusions and Future Work

### Contents

5.1	Conclusions	69
5.2	Contributions	70
5.3	Future Work	71



## 5.1 Conclusions

In recent times, more and more accurate machine learning and data mining techniques are being developed and researched at an incredible pace. To keep up with this trend, new privacy-preserving schemes must be developed to work alongside the machine learning or data mining algorithms in order to perform tasks, such as classification, regression, clustering, among others, whilst preserving the sensitive data of a client. Privacy-preserving machine learning schemes lead the way to a very desirable service: *Secure Machine Learning as a Service* (SMLaaS).

In Chapter 3, distance-preserving hashing techniques were used to develop privacy-preserving emotion classifiers from speech, namely SBE and SMH. Both are based on the idea of leaking information about the true distance of features when the hamming distance in between the fabricated hashes is small enough. It effectively provides information-theoretic security, allowing the scheme to compute machine learning algorithms based on the euclidean distance between of features, such as SVMs. By performing various experiments on a speech emotional dataset, we showed that it is possible to effectively perform privacy-preserving emotion recognition tasks in speech signals with a small accuracy degradation (0.17% on SBE and 2.23% on SMH) and some increased computational time on the client's workspace, since the fabrication of the hashes is his responsibility. The client decides on the precision of these hashes and how much information he is willing to leak by controlling the hashing parameters. We also show that it is possible to apply this scheme to a real world scenario, providing complete privacy to each party involved, by having all data providers to agree on specific hashing keys and sending their hashed data to be trained or tested by an untrustworthy server. Here, the untrustworthy party will have access to the hashes but not the keys, reducing the chance of malicious attempts to decrypt the data.

In chapter 4, a Fully Homomorphic Encryption (FHE) scheme is used to keep up with the newest trend in machine learning, Artificial Neural Networks (ANNs). This is based on the idea of exchanging regular multiplications and additions with homomorphic multiplications and additions, which preserve the calculations made in ciphertexts onto their corresponding plaintexts. We showed that, by means of slight modifications and approximations to the baseline network, it is feasible to effectively construct a privacy-preserving neural network, or simply cryptonet, for speech signals, at a cost of some accuracy degradation and computational time. In fact, the most significant degradation obtained in our experiments is approximately 3.5% for a low polynomial approximation of the activation function (degree 2). For better approximations, the degradation decreases to around 2%. Although this loss of accuracy is not insignificant, it is the price to pay for the security this scheme provides. Moreover, we conclude that training a neural network with such a small training set reduces the accuracy of the baseline and may cause slight overfitting. Furthermore, the security of this scheme surpasses the security of the techniques used in Chapter 3, since there is no leaking of information and all operations performed inside the model are done in the encrypted domain, thus, keeping privacy at all times. Finally, we showed that it is possible to apply this model as a service (SMLaaS) by having the client encrypt speech data, sending

it into an untrustworthy cloud, which is holding a trained model, and recovering encrypted predictions which he may then decrypt, allowing for a complete privacy of the data and its predictions.

Finally, we conclude that the privacy-preserving techniques developed over the course of this thesis all fulfill previously set objectives from Section 1.3. All these schemes offer security of the client's data while effectively performing training or predictions over it, with minimal degradation of accuracy. This will preserve the utility of the model in an encrypted domain.

## 5.2 Contributions

The main contributions of this thesis are the implementation of privacy-preserving schemes for speech processing and machine learning tasks. Most of the protocols implemented in this thesis, to the best of our knowledge, have been previously implemented to work with other kinds of processed data, such as images, text, music and other types of audio data. Some speech tasks have also been performed using SBE [13], however, very little work has been done with preserving-privacy in the field of paralinguistics. Therefore, this thesis contributes to the expansion of privacy-preserving speech mining for paralinguistic tasks and is as an extension of J. Portêlo's work [13], where privacy-preserving frameworks were designed and implemented for a variety of tasks, such as music matching, query-by-example speech search and speaker verification.

Besides these main contributions, this thesis opens a gateway to Secure Machine Learning as a Service (SMLaaS). With the rise of cloud computations, machine learning and data mining algorithms, a great deal of protocols have been devised to work as service applications. The work developed along this thesis allows data to be effectively hidden from malicious parties, while computations may still be performed, thus soothing eventual privacy concerns.

Moreover, using either SBE/SMH algorithms, we showed the possibility of joint learning through data sharing of different untrustworthy parties to improve the training model of a specific task. We also showed that the encrypted testing on a fully encrypted neural network is not only possible, but also quite effective using cryptographic schemes such as FHE. With deep neural networks being the current state-of-the-art for most machine learning tasks with vast training sets, this implementation productively contributes to major advancements in speech mining as a service.

In short, the contributions of this thesis can be summarized in the following list:

- Implementation of privacy-preserving schemes on more challenging speech tasks (paralinguistics).
- Expansion of the literature in privacy-preserving speech mining protocols.
- Advancements in the field of Secure Machine Learning as a Service (SMLaaS).
- Effectively performing predictions on secure and private data with little accuracy degradation.

- Secure Joint Learning (SJL).
- Encrypted testing in neural networks.

## 5.3 Future Work

Although the methods implemented in this thesis provide fair results, it is worth mentioning some optimizations and modifications that may be applied to schemes developed in this thesis, and, also, other new paths of research, regarding the approaches discussed. These may direct researchers to optimize and improve the schemes in security, efficiency, utility or even in computation timing.

Some future work based on the SBE and SMH schemes falls within the following concepts:

- *Proof of non-invertibility*: The main problem with these schemes is still the proof non-invertibility of the matrix  $A$ , even though the scheme offers information-theoretic security. Nonetheless, even if  $A$  was invertible, the computation of the inverse would still be too computationally demanding, unless the attacker has some prior knowledge about the true feature vector. Anyhow, proving the non-invertibility of  $A$  would certainly solve the main security issue of the SBE and allow for the keys  $A$  and  $w$  to be released into the public, opening doors for simpler and more useful real world scenarios, such as, multiparty computations.
- *Role of modulo  $k$* : Another subject for future research is to understand the role of the  $k$  modulo, from Equation 3.5 of the SMH approach, in certain tasks. For now, the  $k$  serves simply as another parameter to help control the leakage of the scheme and it would be useful for it to serve other purposes in certain scenarios.
- *End-to-end approach*: The creation of the hashes in both approaches relies on the pre-processing of the speech signals to recover paralinguistic and other features. These features are then turned into hashes and fed into the model. It might be useful to ignore the pre-processing of the signal and to develop an end-to-end model, decreasing the user's need to extract features. This end-to-end model may even increase accuracy results, since features are learned for the specific task in the training phase.

Regarding cryptonets, there are some interesting improvements and other related implementations to be made. Our experiments serve mainly as a proof of concept of the usage of FHE with speech neural networks, and it draws a path to much more complex cryptonets. Some eventual future work and research may fall within the following subjects:

- *Deep Cryptonet*: Our cryptonet experiments are based on a light ANN, which means that the network has very few layers, and both the noise and the deviations on the activation function entries will not be too significant. This is not the case in deeper ANNs, also known as DNNs. In

this case, the noise growth and other deviations will undergo a significant increase and are due for new modifications and improvements to the network, such as controlling the batch normalization layer in a way that restrains the activation inputs to the reliable approximation interval. Also, DNNs currently hold state-of-the-art accuracy results and some privacy-preserving research is due in this field.

- *End-to-end Cryptonet*: End-to-end deep convolutional neural networks , such as the one in Appendix B, are the state-of-the-art machine learning schemes in the scope of paralinguistics, and devising a privacy-preserving protocol for this scheme does not only, end the need of speech pre-processing, such as feature extraction, but also provide state-of-the-art results in privacy-preserving speech mining. Not to mention that more practical real world scenarios become available with this implementation.
- *Encrypted Learning in ANN*: It would be rather interesting to have the training phase also performed on an encrypted domain. This encrypted learning would broaden the scope in which these privacy-preserving schemes could be applied in the real world. One example is the construction of an application that has a training model and receives encrypted data for prediction. These predictions are then returned to the client and the server may ask him or her if the prediction is correct or not, to the best of their knowledge. The application would then use it as encrypted training data to improve the training model.
- *Deep learning with Differential Privacy*: Another privacy-preserving scheme we came across during research but did not have the chance to implement is deep learning with differential privacy [58]. It would be rather interesting to apply this privacy-preserving framework on speech signals, since it also allows encrypted learning under modest privacy budgets.

# Bibliography

- [1] P. Boufounos and S. Rane, "Secure binary embeddings for privacy preserving nearest neighbors," in *Information Forensics and Security (WIFS), 2011 IEEE International Workshop on*. IEEE, 2011, pp. 1–6.
- [2] A. Jiménez, B. Raj, J. Portêlo, and I. Trancoso, "Secure modular hashing," in *Information Forensics and Security (WIFS), 2015 IEEE International Workshop on*. IEEE, 2015, pp. 1–6.
- [3] H. Chabanne, A. de Wargny, J. Milgram, C. Morel, and E. Prouff, "Privacy-preserving classification on deep neural network." *IACR Cryptology ePrint Archive*, vol. 2017, p. 35, 2017.
- [4] K. Laine, R. Player, and H. Chen, "Simple encrypted arithmetic library - seal v2.2," Technical report, September, Tech. Rep., 2017.
- [5] S. Ultes, M. J. P. Sánchez, A. Schmitt, and W. Minker, "Analysis of an extended interaction quality corpus," in *Natural Language Dialog Systems and Intelligent Assistants*. Springer, 2015, pp. 41–52.
- [6] K. Hirose and N. Minematsu, "Use of prosodic features for speech recognition," in *Proc. ICSLP, Jeju*, 2004, pp. 1445–1448.
- [7] G. D. Sree, P. Chandrasekhar, and B. Venkateshulu, "Svm based speech emotion recognition compared with gmm-ubm and nn," *International Journal of Engineering Science*, vol. 3293, 2016.
- [8] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep Neural Networks for Acoustic Modeling in Speech Recognition," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.
- [9] A. K. Jain, K. Nandakumar, and A. Nagar, "Biometric template security," *EURASIP Journal on advances in signal processing*, vol. 2008, p. 113, 2008.
- [10] L. Greenemeier, "What Could Criminals Do with 5.6 Million Fingerprint Files?" *Scientific American*, September 30, 2015. [Online]. Available: <https://blogs.scientificamerican.com/observations/what-could-criminals-do-with-5-6-million-fingerprint-files/>

- [11] A. Davletcharova, S. Sugathan, B. Abraham, and A. P. James, "Detection and Analysis of Emotion from Speech Signals," *Procedia Computer Science*, vol. 58, pp. 91–96, 2015. [Online]. Available: <http://dx.doi.org/10.1016/j.procs.2015.08.032>
- [12] I. Shafran and M. Mohri, "A comparison of classifiers for detecting emotion from speech," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 1, 2005.
- [13] J. M. L. Portêlo, "Privacy-preserving frameworks for speech mining," Ph.D. dissertation, Instituto Superior Técnico, 2015.
- [14] G. Trigeorgis, F. Ringeval, R. Brueckner, E. Marchi, M. A. Nicolaou, B. Schuller, and S. Zafeiriou, "Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 5200–5204.
- [15] J. Bhaskar, K. Sruthi, and P. Nedungadi, "Hybrid approach for emotion classification of audio conversation based on text and speech mining," *Procedia Computer Science*, vol. 46, no. Icict 2014, pp. 635–643, 2015. [Online]. Available: <http://dx.doi.org/10.1016/j.procs.2015.02.112>
- [16] M. Asgari and I. Shafran, "Predicting Severity of Parkinson ' s Disease from Speech," *Engineering In Medicine And Biology*, pp. 5201–5204, 2010.
- [17] U. Shrawankar and V. Thakare, "Techniques for Feature Extraction in Speech Recognition System : a Comparative Study," *International Journal Of Computer Applications In Engineering, Technology and Sciences (IJCAETS)*, pp. 412–418, 2013.
- [18] S. Chu, S. Member, S. Narayanan, and C. J. Kuo, "Time – Frequency Audio Features," *Language*, vol. 17, no. 6, pp. 1142–1158, 2009. [Online]. Available: [http://ieeexplore.ieee.org/xpls/abs/\\_all.jsp?arnumber=5109766](http://ieeexplore.ieee.org/xpls/abs/_all.jsp?arnumber=5109766)
- [19] P. Shen, Z. Changjun, and X. Chen, "Automatic Speech Emotion Recognition using Support Vector Machine," *Electronic and Mechanical Engineering and Information Technology (EMEIT), 2011 International Conference*, vol. 2, pp. 621–625, 2011.
- [20] D. R. Tobergte and S. Curtis, "Accent Identification," *Journal of Chemical Information and Modeling*, vol. 53, no. 9, pp. 1689–1699, 2013.
- [21] N. Singh and R. A. Khan, "MFCC and Prosodic Feature Extraction Techniques : A Comparative Study," *International Journal of Computer Applications*, vol. 54, no. 1, pp. 9–13, 2012.
- [22] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4580–4584.

- [23] Q. Mao, M. Dong, Z. Huang, and Y. Zhan, "Learning salient features for speech emotion recognition using convolutional neural networks," *IEEE Transactions on Multimedia*, vol. 16, no. 8, pp. 2203–2213, 2014.
- [24] B. Milde and C. Biemann, "Using representation learning and out-of-domain data for a paralinguistic speech task," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.
- [25] T. Falk, "Nonintrusive speech quality estimation using Gaussian mixture models," *Signal Processing Letters, IEEE*, vol. 13, no. 2, pp. 108–111, 2006. [Online]. Available: <http://ieeexplore.ieee.org/xpls/abs{.}all.jsp?arnumber=1576792>
- [26] V. Chauhan, S. Dwivedi, P. Karale, and P. S. M. Potdar, "Speech To Text Converter Using Gaussian Mixture Model ( Gmm )," pp. 160–164, 2016.
- [27] M. Shahin, B. Ahmed, J. McKechnie, K. Ballard, and R. Gutierrez-Osuna, "Comparison of GMM-HMM and DNN-HMM based pronunciation verification techniques for use in the assessment of childhood apraxia of speech," *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, no. September, pp. 1583–1587, 2014.
- [28] O. Chia Ai, M. Hariharan, S. Yaacob, and L. Sin Chee, "Classification of speech dysfluencies with MFCC and LPCC features," *Expert Systems with Applications*, vol. 39, no. 2, pp. 2157–2165, 2012. [Online]. Available: <http://dx.doi.org/10.1016/j.eswa.2011.07.065>
- [29] T. Arias-Vergara, J. C. Vasquez-Correa, J. R. Orozco-Arroyave, J. F. Vargas-Bonilla, T. Haderlein, and E. Noeth, "Gender-dependent gmm-ubm for tracking parkinson's disease progression from speech," in *Speech Communication; 12. ITG Symposium; Proceedings of. VDE*, 2016, pp. 1–5.
- [30] T. Arias-Vergara, J. Vasquez-Correa, J. R. Orozco-Arroyave, J. Vargas-Bonilla, and E. Nöth, "Parkinson's disease progression assessment from speech using gmm-ubm," *Interspeech 2016*, pp. 1933–1937, 2016.
- [31] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [32] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [33] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on. IEEE*, 2013, pp. 6645–6649.
- [34] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014, pp. 1764–1772.

- [35] Y. A. A. S. S. Aldeen, M. Salleh, and M. Abdur Razzaque, "A comprehensive review on privacy preserving data mining," *SpringerPlus*, vol. 4, no. 694, pp. 1–36, 2015. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6510302>
- [36] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data." in *NDSS*, 2015.
- [37] M. A. Pathak, *Privacy-preserving machine learning for speech processing*. Springer Science & Business Media, 2012.
- [38] C. C. Aggarwal and S. Y. Philip, "A general survey of privacy-preserving data mining models and algorithms," in *Privacy-preserving data mining*. Springer, 2008, pp. 11–52.
- [39] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Foundations of secure computation*, vol. 4, no. 11, pp. 169–180, 1978.
- [40] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *International Conference on Machine Learning*, 2016, pp. 201–210.
- [41] C. Gentry *et al.*, "Fully homomorphic encryption using ideal lattices." in *STOC*, vol. 9, no. 2009, 2009, pp. 169–178.
- [42] J. W. Bos, K. E. Lauter, J. Loftus, and M. Naehrig, "Improved security for a ring-based fully homomorphic encryption scheme." in *IMA Int. Conf.* Springer, 2013, pp. 45–64.
- [43] S. Halevi and V. Shoup, "Helib-an implementation of homomorphic encryption," 2014.
- [44] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.
- [45] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 1999, pp. 223–238.
- [46] T. Graepel, K. Lauter, and M. Naehrig, "MI confidential: Machine learning on encrypted data," in *International Conference on Information Security and Cryptology*. Springer, 2012, pp. 1–21.
- [47] L. J. Aslett, P. M. Esperança, and C. C. Holmes, "Encrypted statistical machine learning: new privacy preserving methods," *arXiv preprint arXiv:1508.06845*, 2015.
- [48] L. Aslett, P. M. Esperança, and C. C. Holmes, "A review of homomorphic encryption and software tools for encrypted statistical machine learning," *arXiv preprint arXiv:1508.06574*, 2015.
- [49] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM Journal on computing*, vol. 18, no. 1, pp. 186–208, 1989.

- [50] J.-J. Quisquater, L. C. Guillou, and T. A. Berson, "How to explain zero-knowledge protocols to your children," in *Advances in Cryptology—CRYPTO'89 Proceedings*. Springer, 1990, pp. 628–631.
- [51] S. Even, O. Goldreich, and A. Lempel, "A randomized protocol for signing contracts," *Communications of the ACM*, vol. 28, no. 6, pp. 637–647, 1985.
- [52] Y. Huang, D. Evans, J. Katz, and L. Malka, "Faster secure two-party computation using garbled circuits," in *USENIX Security Symposium*, vol. 201, no. 1, 2011.
- [53] D. Malkhi, N. Nisan, B. Pinkas, Y. Sella *et al.*, "Fairplay-secure two-party computation system," in *USENIX Security Symposium*, vol. 4. San Diego, CA, USA, 2004.
- [54] V. Kolesnikov and T. Schneider, "Improved garbled circuit: Free xor gates and applications," *Automata, Languages and Programming*, pp. 486–498, 2008.
- [55] B. Pinkas, T. Schneider, N. P. Smart, and S. C. Williams, "Secure two-party computation is practical," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2009, pp. 250–267.
- [56] S. Zahur, M. Rosulek, and D. Evans, "Two halves make a whole," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2015, pp. 220–250.
- [57] M. Bellare, V. T. Hoang, S. Keelveedhi, and P. Rogaway, "Efficient garbling from a fixed-key blockcipher," in *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE, 2013, pp. 478–492.
- [58] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 308–318.
- [59] S. Rane and P. T. Boufounos, "Privacy-preserving nearest neighbor methods: Comparing signals without revealing them," *IEEE Signal Processing Magazine*, vol. 30, no. 2, pp. 18–28, 2013.
- [60] P. T. Boufounos, "Universal rate-efficient scalar quantization," *IEEE transactions on information theory*, vol. 58, no. 3, pp. 1861–1872, 2012.
- [61] F. Eyben, M. Wöllmer, and B. Schuller, "Opensmile: the munich versatile and fast open-source audio feature extractor," in *Proceedings of the 18th ACM international conference on Multimedia*. ACM, 2010, pp. 1459–1462.
- [62] F. Eyben, K. R. Scherer, B. W. Schuller, J. Sundberg, E. André, C. Busso, L. Y. Devillers, J. Epps, P. Laukka, S. S. Narayanan *et al.*, "The geneva minimalistic acoustic parameter set (gemaps) for voice research and affective computing," *IEEE Transactions on Affective Computing*, vol. 7, no. 2, pp. 190–202, 2016.

- [63] A. Raux, B. Langner, D. Bohus, A. W. Black, and M. Eskenazi, "Let's go public! taking a spoken dialog system to the real world," in *in Proc. of Interspeech 2005*. Citeseer, 2005.
- [64] A. Schmitt, S. Ultes, and W. Minker, "A parameterized and annotated spoken dialog corpus of the cmu let's go bus information system." in *LREC*, 2012, pp. 3369–3373.
- [65] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.
- [66] G. Taylor Frame, "Heide: An ide for the homomorphic encryption library helib," Master's thesis, California Polytechnic State University, San Luis Obispo, June 2015.
- [67] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption." *IACR Cryptology ePrint Archive*, vol. 2012, p. 144, 2012.
- [68] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2010, pp. 1–23.
- [69] T. N. Sainath, R. J. Weiss, A. Senior, K. W. Wilson, and O. Vinyals, "Learning the speech front-end with raw waveform cldnns," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.



## Corpora

This appendix presents a more detailed description of the dataset used in this thesis. Information about the interaction quality, the emotion state annotations, the raters, rating guidelines and other useful information is also available in this appendix.

### A.1 Let's Go / LEGO

The "Let's Go" public dialogue system [63] was developed and is maintained by the Dialogue Research Center at Carnegie Mellon University. It provides the bus schedule information for Pittsburgh's Port Authority buses during off-peak hours. The system ran live from 2005 until 2014, with an average of 40 received calls per day on weekdays and 90 received calls on the weekends. The *LEGO* [64] is an updated version of the "Let's Go" and is comprised of a total of 347 calls and 9,083 system-user exchanges, in which 200 of those calls and 4,885 exchanges have Interaction Quality (IQ) and emotion state annotations. Regarding the *Task Success*, each call has been annotated with a Task Success label, which is either *Completed* (187), *failed due to system behaviour* (15), *found out there is no solution* (52), *not completed* (71) or *partially completed* (3). These labels were ultimately derived from an heuristic scheme, where the number of REQUEST, CONFIRMATION, and ERROR actions and the number of NO-MATCHES has been used [64].

The corpus used for the tests performed in this thesis is an extended version of the *LEGO* corpus [64] which is comprised of an additional 201 calls and 4753 exchanges with IQ and emotional state annotations from three different expert raters, constituting the *LEGOext* corpus [5]. These raters were asked to annotate each system-user exchange with (1) Extremely Unsatisfied, (2) Strongly Unsatisfied, (3) Unsatisfied, (4) Slightly Unsatisfied or (5) Satisfied, according to the following guidelines [64]:

1. The rater should try to mirror the users point of view on the interaction as objectively as possible.
2. An exchange consists of the system prompt and the user response. Due to system design, the latter is not always present.
3. The IQ score is defined on a 5-point scale with (1) Bad, (2) Poor, (3) Fair, (4) Good or (5) Excellent.
4. The Interaction Quality is to be rated for each exchange in the dialogue. The dialogue's specific history should be minded. For example, a dialogue that has proceeded fairly poor for a long time, should require some time to recover.
5. A dialogue always starts with an IQ score 5.
6. The first user input should also be rated with 5, since until this moment, no rateable interaction has taken place.
7. A request for help does not invariably cause a lower IQ, but can result in it.
8. In general, the score from one exchange to the following exchange is increased by one point at the most.
9. Exceptions, where the score can be decreased by two points, are for example, hot anger or sudden frustration. The rater's perception is decisive here.
10. If the dialogue obviously collapses due to system or user behaviour, the score can be set to 1 immediately. An example is a reasonable frustrated sudden hang-up.
11. Anger does not need to influence the score, but can. The rater should try to figure out whether it might be caused by the dialogue behaviour or not.
12. In case a user realizes that he should adapt his dialogue strategy to obtain the desired result and actually succeeded that way, the IQ score can be raised up to two points per turn. In a manner of speaking, the user realizes that he caused the poor IQ by himself.
13. If the system does not reply with a bus schedule to a specific user query and prompts that the request is out of scope, this can be considered as completed task and therefore does not need to affect the IQ.

**Table A.1:** Statistics of the two corpora LEGO and LEGOext and of the combined corpus LEGOV2 and the actual used corpus in this thesis (LEGOv2.1). Shown are the reference, the total number of calls, the number of calls with emotional labels, the total number of exchanges, the number of exchanges with emotional labels, the average dialogue length in number of exchanges and the inter-rater agreement (Adapted [5]).

Corpus	Ref.	# TotCalls	# emoCalls	# TotExch	# emoExch	Avg. Length	$k$
LEGO	[64]	347	200	9,083	4,885	25.4	0.54
LEGOext	[5]	201	201	4,753	4,753	22.6	0.50
LEGOv2	[5]	547	401	13,836	9,638	24.0	0.52
LEGOv2.1	—	547	302	13,836	4,243	—	—

14. If a dialogue consists of several independent queries, each query's quality is to be rated independently. The dialogue's history shouldn't be minded when a new query begins. However, the score provided for the first exchange should be equal to the last label of the previous query.
15. If a dialogue proceeds fairly poor for a long time, the rater should consider to increase the score more slowly if it's getting better. Also, in general, he or she should observe the remaining dialogue more critically.
16. If a constantly low-quality dialogue finishes with a reasonable result, the IQ should be increased.

Following the same rating guidelines, the rating of the original *LEGO* corpus was also performed. The three raters achieved an inter-rater agreement  $k = 0.54^2$  for the original *LEGO* and  $k = 0.50^2$  for the *LEGOext*. The Cohen's Kappa ( $k$ ) is used as the measurement for the relative agreement between two corresponding sets of ratings, the number of label agreements corrected by the chance level of agreement divided by the maximum proportion of times the labelers could agree is computed. General statistics for both corpora as well as the combined corpus *LEGOv2* are depicted in Table A.1. This combined corpus consists of 547 total number of calls and 13,836 total number of system-user exchanges.

Besides the IQ annotations, one of the raters classified each exchange, from 401 calls and 9,638 exchanges, with a negative emotional state of the user. Adapted from the original *LEGOv2*, the rater labeled each exchange, from a total of 302 calls and 4,243 exchanges, as *friendly*, *neutral*, *slightly angry* or *very angry*. Moreover, a binary emotion class labeling is also provided, where each exchange is classified either with *neutral* or *angry*. This updated version of the *LEGOv2*, also known as *LEGOv2.1*, contains less calls and exchanges with emotional state labels than the actual *LEGOv2* (See Table A.1).



# B

## End-to-end Emotion Classifier

This appendix contains information about the implementation and testing of a deep convolutional recurrent network, which is, to the best of our knowledge, the state-of-the-art in emotion recognition. Section B.1 introduces the topic of end-to-end emotion recognition using speech and, Section B.2 contains the actual implementation, results and discussion about the end-to-end emotion recognizer.

### B.1 Introduction

End-to-end approaches are an increasing trend in machine learning tasks, such as classification, regression and clustering, urging the scientific community to develop new and effective frameworks for machine learning and data mining tasks using powerful ANNs and deep learning. Even though hand-engineered features have been shown to be quite effective a wide number of tasks, the idea of having no pre-processing of the signal, in this case, speech, allows for the development of training models with little to none apriori knowledge about the signal, which results in a more raw input oriented classifier and, thus, better accuracy in a classification.

There is always an interest for individual untrustworthy parties to share data and perform joint machine learning tasks for their own benefit. Thus, as this new end-to-end trend grows, so does the need for private data retrieval and mining frameworks. As it was discussed in Chapter 3, each individual

hand-engineered feature is encrypted or hashed for later processing and classification using the SMH technique. Since, in an end-to-end approach there are no features and no relation to euclidean distance, a privacy-preserving scheme implementation becomes more complex and tricky.

## B.2 Deep Convolutional Recurrent Network

The use of a DCRN for emotion classification directly from raw speech signals was first proposed by Trigeorgis et. al. [14] in 2016, and showed better results in valence and arousal, when comparing against the results from state of the art techniques for emotion recognition from speech at the time.

The design of the DCRN consists of 1-d convolutions that operate in the discrete-time waveform  $h(k)$  and is represented as

$$(f \circledast h)(t) = \sum_{k=-T}^T f(t) \cdot h(t - k), \quad (\text{B.1})$$

where  $f(x)$  is a kernel function whose parameters are learnt from the data of the task at hand. A recurrent network with LSTM cells is used to model the temporal structure of the speech. It has been shown that a time convolution layer reduces temporal variation, a frequency convolution layer allows for a preservation of locality and reduction in frequency variation and LSTM layers serve for contextual modelling of the speech signal [69]. Moreover, LSTMs are very simple, effective and provide a fair comparison against existing hand-engineered features approaches [14]. The final model is obtained by jointly training both the convolutional model and the recurrent model with LSTM cells with backpropagation using the same objective function, defined as

$$L_c = 1 - \rho_c = \frac{2\sigma_{xy}^2}{\sigma_x^2 + \sigma_y^2 + (\mu_x - \mu_y)^2}. \quad (\text{B.2})$$

This proposed DCRN architecture for emotion recognition outperformed hand-engineered features, such as, eGeMAPS and ComParE, both in arousal and valence [14].

### B.2.1 Experimental Setup

The corpus considered for all experiments is the Let's Go corpus. As it was mentioned in Chapter ??, "Let's Go" [63] is a public dialogue system that provides the bus schedule information for Pittsburgh's Port Authority buses during off-peak hours. The corpus used on the baseline of the emotional classifier was, again, the *LEGOv2.1* corpus, which is an updated version of the original "Let's Go" Corpus [5, 63, 64] with emotional labels provided by an expert rater. It contains a total of 547 calls to the callcenter, where 302 of those have emotional annotations, and a total of 13,836 system-user exchanges, where 4,243 of those have emotional annotations. The emotional labels are provided as a binary class representation, where each exchange is either classified as *neutral* or *angry*, and a multiclass representation, where

each exchange is classified with *friendly*, *neutral*, *slightlyAngry* or *veryAngry*. More information about this corpus is present in the Appendix A.

The same labelled sets used in Chapters 3 and 4 were also used for these experiments. All experiments for this baseline were performed using Python and the deep neural network toolkit, Tensorflow. The training, validation and test sets were converted into a more suitable format using TFrecords. The training model is devised of two submodels, an audio model, composed by a CNN and, a recurrent model implemented on top of the previous.

### Audio Model

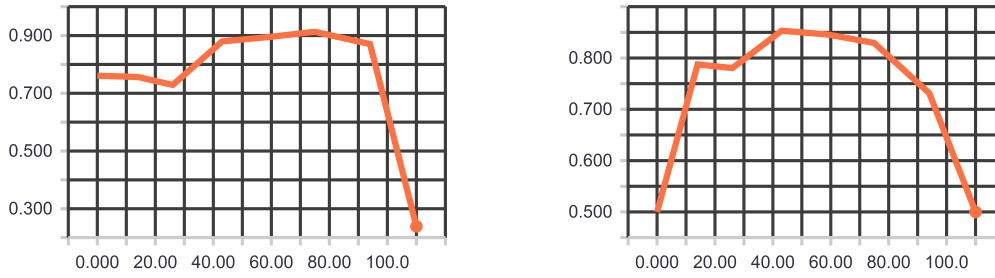
The audio model, also known as the convolutional model, consists of several layers of convolution filters being applied to a signal, which allows for a better representation of this raw input signal and, therefore, a model more fitting to the task at hand. This convolutional model is summarized with the following major steps:

- *Inputs*: Raw recordings from the LEGO dataset, sampled at 8Khz and divided into chunks of 40ms.
- *Average Pooling*: The raw inputs pass through pooling function with a pool size = 2, allowing a subsampling of the input signals.
- *1-d Convolutions*: The convolution process consists in 8 layers, each composed of 32 convolution filters, that are convoluted with the input signals, after the average pooling operation. This convolution layer may also be designated as feature extraction layers of the model.
- *Max Pooling*: The max pooling, with a pool size = 2, is performed subsequently to the convolution process, for each of the convolution layers. This max pooling allows for a more selective feature learning. In short, it chooses the maximum and most adequate feature, from all the other adjacent ones.

This convolution process is the most crucial component of the model. It, not only, allows for a removal of background noise, but also, enhances certain relevant parts of the input signal, leading to a more accurate feature extraction for a certain task that is to be performed [14].

### Recurrent Model

The second submodel is based on the LSTM architecture, which is a specific architecture of RNNs. Besides having the same capabilities of a standard RNN, the LSTM also solves the long-term dependencies of the RNN that are quite common in more complex problems. For this model, LSTMs are used for simplicity [14] and due to the fact that LSTMs allow for inputs to be variable in length, as it was explained in Section 2.1. This fact is very advantageous, since the inputs do not need to be all the same size, and thus, the model becomes more generalized. The recurrent model implemented used two bidirectional LSTM modules, each with 64 hidden layers.



(a) Accuracy evolution against the number of training steps.

(b) Unweighted average recall against the number of training steps.

**Figure B.1:** Evolution of the evaluation results on the validation set against the number of training steps.

Having both submodels implemented in Python, these are then jointly trained by backpropagation using the cost function in Equation B.2 and subsequently optimized with the standard Adam Optimizer.

While the DCRN model is being trained, both the validation and test set are evaluated every 60 seconds to check the progress of the training. From this iterated evaluation, information about the accuracy and the unweighted average recall is obtained. Recall is defined as the amount of recordings classified as a specific class divided by the total amount of recordings labelled as that class. And so, the unweighted average recall is the average between all the recalls and should as close to 1 as possible. The results for the validation set can be inspected in Figure B.1.

## B.2.2 Results and Discussion

As it can be depicted from Figure B.1(a), as the training carries on, the accuracy of the DCRN emotion classifier for the validation set tends to grow until it reaches a point where it classifies all recordings as either one of the classes. This effect might be due to the fact that after a big number of steps, the learning parameters become unsuitable to train a convolutional recurrent model which results in all recording being classified with the same class. From Figure B.1(b), it is possible to observe the growing average recall until it reaches a peak, at which point decreases. This average recall can be seen as the "completeness" of the evaluation, i.e., it is the average between the amount of recording classified as one class divided by all labelled recordings from that class.

Having carefully analyzed the behaviour of the classifier on the validation set, it is possible to see that the model created in step 75 resulted in maximum accuracy for the validation set of 91.32 %. Afterwards, this specific model was used to evaluate the test set, which evaluated the test set with accuracy levels of **81.41 %**. It is important to note that convolutional neural networks require an enormous amount of training data in order to be reliable and effective and thus, a training set composed of 3,005 recording is considered a small training set and might not be the best suitable training set. Still, for such a small training set, the classification results are very similar to the classification results obtained using hand-

engineered features, not to mention the fact that, there is no pre-processing of the speech signal nor the need to carefully depict prosodic cues and features, all of which may decrease the performance of the classifier if some relevant information is discarded. On the other hand, the existence of end-to-end techniques, such as DCRN, allows for a more accurate representation of the raw signal, since no valuable paralinguistic information will be lost. Moreover, the end-to-end training is more specific and will better suit the task which it was meant to train, leading to better accuracy results.





# SEAL Library

This appendix contains complimentary information about the encryption library used in Chapter 4, SEAL, which is a fully homomorphic encryption library based on the FV security scheme. This appendix is based on the SEAL manual [4]. Section C.1 introduces the topic of FHE and library. Section C.2 presents the encryption parameters and their uses. Section C.3 describes the different encoders in SEAL. Section C.4 explains what is relinearization and why it is useful. Finally, Section C.5 describes the noise present in homomorphic operations.

## C.1 Introduction

Homomorphic encryption has become one of the bases of encrypted computations. Several partial homomorphic schemes have already been intensively researched, such as the Pailler [45], an additive homomorphic encryption scheme, and RSA [44], a multiplicative homomorphic scheme. The first real effective by Craig Gentry in 2009 [41], the research done to improve FHE schemes has been substantial. In 2015, the first version of the *Simple Encrypted Arithmetic Library* SEAL was released, with the specific goal of providing a well-engineered and documented homomorphic encryption library, with no external dependencies, that would be easy to use both by experts and by non-experts with little or no cryptographic background [4].

This document, therefore, describes the main features of SEAL, as well as security guarantees, noise growth, encryption scheme, among others.

## C.2 Encryption Parameters

The encryption parameters of seal are the following.

- *poly\_modulus*: a polynomial  $x^n + 1$ , being  $n$  a power of 2.
- *coeff\_modulus*: an integer modulus  $q$ .
- *plain\_modulus*: an integer modulus  $t$ .
- *noise\_standard\_deviation*: a standard deviation  $\sigma$ .
- *noise\_max\_deviation*: a bound for the error distribution  $B$ .
- *decomposition\_bit\_count*: the logarithm  $\log(w)$  of  $w$ .
- *random\_generator*: a source of randomness.

The selection of the *poly\_modulus*, *coeff\_modulus* and *plain\_modulus* are crucial to the encryption. This selection significantly affects the performance, capabilities and security of the encryption scheme [4]. The rest of the parameters usually offer good default values and, therefore, are not necessary for the user to set, in typical situations. Table C.1 presents a list of the default values of the  $n$  and  $q$ , and their corresponding security. The security level estimates use the LWE estimator, which takes into account the recent attacks.

**Table C.1:** Default pairs  $(n, q)$  and their estimated security levels [4].

$n$	$q$	Security Estimate (bits)
2048	$2^{60} - 2^{14} + 1$	115.1
4096	$2^{116} - 2^{18} + 1$	119.1
8192	$2^{226} - 2^{26} + 1$	123.1
16384	$2^{435} - 2^{33} + 1$	130.5
32768	$2^{889} - 2^{54} - 2^{53} - 2^{52} + 1$	127.7

SEAL has a feature that helps users to choose parameters for a specific computation. It consists in two parts. First, a *simulator* component simulates the noise growth in homomorphic encryption operations using the estimates in Table C.2. Afterwards, a *chooser* component estimates the growth of the coefficients in the underlying plaintext polynomials [4].

## C.3 Encoding/Decoding

One of the most important aspects in making homomorphic encryption practical and useful is in using an appropriate encoder for the task at hand. SEAL allows the use of four encoders: *Scalar Encoder*, *Integer Encoder*, *Fractional Encoder*, and a *CRT Batching Encoder*.

### Scalar Encoder

The *scalar encoder* is the simplest encoder possible. Given an integer  $a$ , the process is to simply encode it as a constant polynomial. Decoding the constant polynomial is basically reading the constant coefficient and interpreting it as an integer. This, however raises an issue. As soon as the underlying plaintext polynomial wraps around modulo  $t$  at any point during the computation, we are no longer doing integer arithmetic, but rather modulo  $t$  arithmetic, and decoding might yield an unexpected result. To solve this, the integer needs to be encrypted twice using different prime plaintext moduli [4]. Still, the scalar encoder is not a wise choice for most practical situations.

### Integer Encoder

The *integer encoder* is used to encode integers in a much more efficient manner. The idea is to encode an integer  $-(2^n - 1) \leq a \leq 2^n - 1$  as follows. First, form the (up to  $n$ -bit) binary expansion of  $|a|$ . Then, the binary encoding of  $a$  is

$$\text{IntegerEncode}(a, B = 2) = \text{sign}(a) \cdot (a_{n-1}x^{n-1} + \dots + a_1x + a_0).$$

To decode, SEAL evaluates the plaintext polynomial at  $x = 2$ .

This integer encoder is significantly better than the scalar encoder, as the coefficients in the beginning are much smaller than in the plaintexts encoded with the scalar encoder, leaving more room for homomorphic operations before problems with reduction modulo  $t$  are encountered [4].

### Fractional Encoder

To encode rational numbers, SEAL scales all rational numbers into integers, encoding them using an integer encoder and modify the computations.

Consider the rational number 5.8125. It has a finite binary expansion of  $5.875 = 2^2 + 2^0 + 2^{-1} + 2^{-2} + 2^{-4}$ . First, the integer part is encoded with an integer encoder. Afterwards, to the fractional part is added  $n$  to each exponent, and converted into a polynomial by changing the base 2 into variable  $x$  and flip the signs of each term, resulting in  $-x^{n-1} - x^{n-2} - x^{n-4}$ . Then the encoding is set as

$$\text{FracEncoder}(5.8125, B = 2) = -x^{n-1} - x^{n-2} - x^{n-4} + x^2 + 1.$$

To decode, is essentially to run the protocol in reverse [4].

## CRT Batching

This encoder is extremely powerful and is based on the *Chinese Remainder Theorem* (CRT). It allows the user to pack  $n$  integer modulo  $t$  into one plaintext polynomial, and to operate on those integer in a SIMD (Single Instruction, Multiple Data) manner. However, this only works when the plaintext modulus  $t$  is chosen to be a prime number and congruent to  $1 \bmod 2n$ , which is assumed to be the case [4].

When used correctly, batching can provide an enormous performance improvement over the other encoders.

## C.4 Relinearization

The goal of relinearization is to decrease the size of the ciphertext back to, at least, 2 after it has been increased by multiplications. This conversion will require an evaluation key, at least, to be given to the evaluator. It is important to notice that, to generate evaluation keys, one needs access to the secret key, which means that only the user has the power to generate evaluation keys. This repeated reduction of the size of a ciphertext allows for better performances and quicker computations [4].

## C.5 Noise

In homomorphic encryption, every ciphertext has a property named noise, which grows as homomorphic operations are performed. In SEAL, a different definition for the noise is used, *invariant\_noise*. The *noise\_budget* of a ciphertext is defined to be  $-\log_2(2v)$ , where  $v$  is the size of the invariant noise in a ciphertext. With this, a ciphertext is only decryptable if this noise budget is greater than zero. If the noise budget reaches zero, the ciphertext becomes undecryptable [4]. Table C.2 presents the noise growth estimates implemented in SEAL.

**Table C.2:** Noise estimates for homomorphic operations in SEAL [4].

Operation	Input Description	Noise Bound of Output
Encrypt	Plaintext $m \in R_t$	$2B\sqrt{2n/3}$
Negate	Ciphertext $ct$	$\ v\ $
Add/Sub	Ciphertext $ct_1$ and $ct_2$	$\ v_1\  + \ v_2\  + r_t(q)$
AddPlain/SubPlain	Ciphertext $ct$ and plaintext $m$	$\ v\  + r_t(q)$
MultiplyPlain	Ciphertext $ct$ and plaintext $m$ with $N$ non-zero coefficients	$N\ m\ (\ v\  + r_t(q)/2)$
Multiply	Ciphertext $ct_1$ and $ct_2$ of sizes $j_1 + 1$ and $j_2 + 1$	$t(\ v_1\  + \ v_2\  + r_t(q)) \times \lceil \sqrt{2n/3} \rceil^{j_1+j_2-1} 2^{j_1+j_2}$
Square	Ciphertext $ct$ of size $j$	Same as Multiply(ct,ct) but faster
Relinearize	Ciphertext $ct$ of size $K$ and target size $L$ such that $2 \leq L < K$	$\ v\  + \  + (K - L)\sqrt{n}B(l + 1)w$
AddMany	Ciphertexts $ct_1, \dots, ct_k$	$\sum_i \ v_i\  + \  + (k - 1)r_t(q)$
MultiplyMany	Ciphertexts $ct_1, \dots, ct_k$	Apply Multiply in a tree-like manner, and Relinearize down to size 2 after every multiplication
Exponentiate	Ciphertext $ct$ and exponent $k$	Apply MultiplyMany to $k$ copies of ct