Privacy in Paralinguistic Tasks

Francisco Saraiva Sepúlveda Teixeira, Instituto Superior Técnico, Universidade de Lisboa

***Abstract*—The widespread use of devices with internet access, together with the emerging market for data mining applications has raised concerns over the level of privacy currently given to users. Taking advantage of increasingly accurate Machine Learning algorithms, many services use sensitive data to extract information and make predictions about the characteristics of their users. Among other data types, speech stands out for the amount of information it holds. Aside from the linguistic content, from speech one can obtain other information, such as the speakers age, gender, health and personality traits. However, the reasons that make speech useful also make it a target for malicious third parties intending to obtain sensitive information about unsuspecting users. This is especially true for health-related applications where a system may try to uncover whether someone presents symptoms of a medical condition, as this information is deeply sensitive.**

In this thesis we show how Homomorphic Encryption can be used to build speech-based privacy-preserving Neural Networks, with focus on three speech affecting conditions: the common Cold, Depression and Parkinsons Disease. To this end, in a first experiment we apply an Encrypted Neural Network, whose operations have been replaced with their encrypted counterparts, to the three aforementioned conditions. On a second approach, we discuss and experiment on the feasibility of building an end-to-end network for the same purpose. Finally, as a last experiment we show how a Neural Network, and its input features, can be discretized in order to allow the use of a Homomorphic Encryption batching technique.

***Index Terms*—Privacy; Machine Learning; Homomorphic Encryption; Speech; Health**

I. Introduction

Speech is one of the primary means of communication for humans. It can be viewed as a carrier for information on several levels as it conveys not only the meaning and intention predetermined by the speaker, the *Linguistic* information, but also information about the age, gender, personality, emotional state, and health of the speaker, among many other *Paralinguistic* and *Extralinguistic* characteristics. The amount of information conveyed in speech makes it the scope of a large number of computational applications that aim to automate tasks such as speech recognition, speech mining, and text-to-speech synthesis. This makes speech useful for technological applications, but it also makes it an uniquely sensitive biometric that should be kept private. As such, it is important to secure not only the user's voice, but also the information obtained from applying pattern recognition systems. This is specially true for applications in which speech is utilized to determine whether or not a person presents symptoms of a medical condition, as this information is deeply sensitive and personal.

Much effort has been put into the development of applications to extract *paralinguistic* information from speech [1]. However, the topic of privacy in speech-based, *paralinguistic* Machine Learning (ML) frameworks still remains largely unexplored, notwithstanding the important role these applications may play in detecting and monitoring certain health conditions.

As the number and reach of internet based application grows, so does the concern on user privacy, as the recently approved European Union General Data Protection Regulation has come to show. Machine Learning as a Service (MLaaS) applications have a particular and privileged access to the user's private information. Nevertheless, in most cases, no privacy guarantee is given for either the user's data or the information obtained from it.

This absence of privacy-preserving solutions for MLaaS applications has resulted in an increasingly active field of research that aims to use Secure Multiparty Computation (SMPC) techniques to develop Privacy-preserving Machine Learning (PPML) solutions for Secure Machine Learning as a Service (SMLaaS) applications [2]. However, there is still a lack of such solutions for speech, and in particular health-related speech *paralinguistic* tasks. The aim of this work is thus to build on existing PPML frameworks, and to apply them to this type of task.

More specifically we will build on the work of [3] and [4], and provide a proof-of-concept on how a Neural Network (NN) can be combined with Homomorphic Encryption (HE) to provide an accurate and secure framework for speech-based medical applications. We also intent to explore end-to-end approaches that are compatible with HE, in order to minimize the user's role in the computation, and avoid a feature extraction stage. Finally, we will try to find ways to improve the framework's computational overhead, caused by HE, through the use of SIMD (Single Instruction Multiple Data), HE techniques.

This paper is organized as follows: In Section II we provide a brief overview of the related works. Section III describes the techniques necessary to understand this work. Sections IV, V and VI provide a description, and the results obtained for the experiments described above. Finally in Sections VII and VIII we provide our conclusions regarding this work, and necessary future work, respectively.

II. Related Work

Privacy-preserving in speech is a fairly recent topic of research. One of the first strides in this direction was made by Pathak et al. [5], who adapted a Gaussian Mixture Model (GMM) to work with HE, to perform both speaker verification and identification. In a different approach, Portłlo et al. [6], and later Jimnez et al. [7], applied Secure Binary Embeddings (SBE) and Secure Modular Hashing (SMH) to speaker verification. The most recent effort in this topic was provided by Dias et al. [8], where both SMH and HE were applied to an emotion recognition task.

Outside speech, the literature is extensive, and a very large number of different techniques has been developed for PPML, including frameworks for linear regression, K-Means Clustering, Support Vector Machines (SMV) and NN. An important contribution to PPML was recently proposed by the authors of Cryptonets [9]. In their paper, the authors perform the prediction stage of NN with Leveled Homomorphic Encryption (LHE), replacing normal operations with their homomorphic counterparts. This work was further improved by Chabanne et al. [10], Hesamifard et al. [11] and more recently by Sanyal et al. [12] and Bourse et al. [13]. Other approaches such as [14] [15] [16] [17] have avoided the computational overhead of HE cryptosystems using Oblivious Transfer (OT) and Garbled Circuits (GC), obtaining very good performance results, with very low communication and computational costs, at the cost of model privacy. In addition, due to the accuracy and computational losses imposed by secure frameworks, very few works perform the training stage in a privacy setting. Two notable examples are [15] and [18], which are based on OT, GC and Differential Privacy (DP).

III. Background

A. Artificial Neural Networks

Artificial Neural Networks (ANN), or simply Neural Networks (NN), are currently one of the most widely used ML techniques. An NN is a set of interconnected nodes, usually following a feed-forward structure. Based on the human brain's cells, the nodes in an NN try to emulate the functioning principle of neurons. For this reason, nodes are also referred to as neurons.

Neurons are usually organized into layers, with the most common layer being the Fully Connected (FC) layer, defined as:

$$\mathbf{y}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}, \tag{1}$$

where $\mathbf{A}$ and $\mathbf{b}$ are the weight matrix and bias vector, respectively, and $\mathbf{x}$ is the input vector. Additionally, activation functions are usually included after each FC layer, such as the Rectified Linear Unit (ReLU), the Sigmoid, the Tanh and the Softmax layers. These layers introduce non-linearity to the network, allowing it to learn more complex functions. An NN can be trained to perform classification or regression tasks, using *backpropagation* algorithms, which take the error of each prediction done over a training set, and re-adjust the networks's weights in order to minimize the error in accordance with a *loss* function.

B. Convolutional Neural Networks

Convolutional Neural Networks (CNN) are a specific type of NN, that is usually composed of a convolutional stage, where the network alternates between convolutional, activation and pooling layers, and a second stage, which follows the same structure of a regular NN, alternating between FC and activation layers.

In a convolutional layer, a set of learnable filters (or filter maps), are convolved with the layer's inputs. Filter maps work similarly to neurons, in the sense that both are weighted sums that have adjustable parameters. Each filter map is applied to small regions of the input, and strided along the full signal, highlighting and identifying the most relevant areas for the task at hand.

Pooling layers work similarly to convolutional layers, as a filter is strided along the inputs of the layer. Pooling layers work simply by selecting an area of the input signal and applying a certain operation to reduce its dimensionality, having no trainable parameters. The Max Pooling layer and the Average Pooling layer are examples of commonly used pooling layers.

C. Homomorphic Encryption

Homomorphic Encryption is a type of cryptosystem in which certain operations performed on *ciphertexts* (i.e. encrypted values) are *homomorphic* with regard to the *plaintexts* (i.e. unencrypted values). In other words, considering the encryption of a value x, $E(x)$ and of a value y, $E(y)$, if a homomorphic operation is performed on the two ciphertexts, the result of this operation will correspond to the equivalent unencrypted operation of the two values, as follows:

$$\begin{aligned} E(x) \otimes E(y) &= E(x \times y), \\ E(x) \oplus E(y) &= E(x + y), \end{aligned} \tag{2}$$

with $\otimes$ and $\oplus$ corresponding to homomorphic multiplication and addition, respectively.

While other HE cryptosystems already existed [19], a Fully Homomorphic Encryption (FHE) system was only achieved in 2009 by Craig Gentry [20]. This scheme was constructed using a lattice-based cryptosystem, in which ciphertexts have an associated noise term, which grows with each homomorphic operation, and after a threshold number of operations, the ciphertext can no longer be decrypted correctly. To make it fully homomorphic, Gentry introduced a key innovation, *bootstrapping*, which consists on homomorphically re-encrypting a ciphertext, reseting its noise level, and allowing for an unlimited amount of operations to be performed over it [21].

However, bootstrapping presents a large computational overhead. As such, taking advantage of the fact that in most applications the user knows beforehand the amount of operations that he intends to perform, and to avoid the computational cost of *bootstrapping*, Leveled Homomorphic Encryption schemes rose as an alternative to FHE. LHE allows the user to select the encryption parameters in such a way that it is possible to determine the maximum noise the ciphertext can have, while still being possible to decrypt them correctly. Conversely, this maximum noise threshold can be defined as a *noise budget*, or the amount of noise that one can "spend". To obtain a larger *noise budget* one has to increase the size of the encryption parameters, which means increasing the computational complexity of each operation in the scheme. To compensate for the computational limitations of LHE, Brakerski et al. [22] proposed an important optimization, called *batching*. This technique allows several messages to be encrypted in the same ciphertext and thus, to be operated on at the same time.

Throughout this work we will use the Simple Encryption Arithmetic Library's (SEAL) implementation of the LHE Fan and Vercauteren (FV) scheme [23]. Additional details

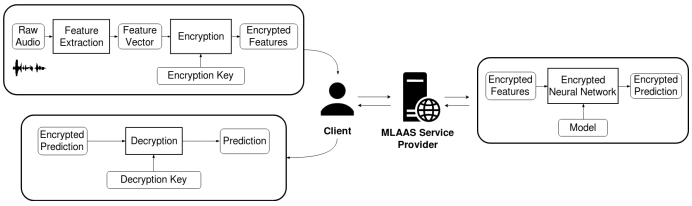

Fig. 1: MLaaS framework using an Encrypted Neural Network.

regarding this implementation, including a security review of the scheme, can be found in its manual [24].

IV. Privacy-preserving Neural Networks

As it has been shown by [9] [10] [11], NNs are perfect candidates for adaptation to a privacy-preserving setting, using Homomorphic Encryption. In these approaches, all mathematical operations in the prediction stage of the NN are performed over encrypted data, taking advantage of the properties of HE.

In Figure 1, we present a scheme of how a MLaaS framework can be applied to speech, using an Encrypted Neural Network (ENN). In this framework, a client extracts a set of features from his/her speech signal, and encrypts it. This vector is then sent to the server who performs an encrypted computation that results in an encrypted prediction, which the server sends to the client, who can then decrypt it. In this way, the client's information is always kept secure, as it is never seen in Plaintext form by anyone other than him/herself. In addition, the encrypted prediction can only be decrypted by the client. Both these conditions prevent the service provider and malicious third parties from learning anything about the user from either the client's data or the prediction of the ENN. Furthermore, since computations are only performed by the server, the model remains private, as opposed to GC-based approaches such as [14] [15] and [17].

A. Encrypted Neural Network

In order to adapt a NN to HE, it is necessary to replace every operation by its HE counterpart. For FC layers, this is simply a matter of replacing additions and multiplications with their HE equivalent. However, nonlinear functions in the network (e.g. activation layers) cannot be computed, and need to be replaced with polynomials.

In Cryptonets, to avoid nonlinear functions, the authors replace each activation function with the square function, although this has some disadvantages during the training stage [9]. To avert these, the authors of [10] proposed instead to train the network with regular Activation functions (such as the ReLU), and replace them with polynomial approximations during inference. However, since the approximations were done using Least Squares, they only have a small convergence interval around the origin and diverge quickly outside of it. As such, the authors proposed the introduction of a Batch Normalization (BN) layer before each Activation Layer, ensuring its inputs will fall within the convergence interval of the polynomial. This approach was also used by Dias et al. [8], where the authors applied the scheme to a speech emotion recognition task. Hesamifard et al. [11] expanded this idea by keeping the BN layer, but approximating the ReLU through its derivative, the Step function, integrating the resulting polynomial, and using it to train the network, as was previously done by Cryptonets. Furthermore, to ensure a more accurate approximation, since the Step function is non-differentiable at zero, the authors decided to approximate a continuous function with similar shape, the Sigmoid, instead of the derivative itself. This was done using Chebyshev polynomials, which allow a trade-off between the size of the convergence interval and the precision of the approximation around zero.

In this work we applied the method proposed by Hesamifard et al. [11], as it obtained the best results of the three above mentioned approaches in the MNIST dataset [25]. However, when approximating the activation functions, the authors of [11] are omissive with regard to the constant term of the polynomial resulting from the integration step, which prompted us to develop our own method to find it. Considering we want the polynomial to have a behavior as similar as possible to the ReLU, we not only want the polynomial to have the smallest error between itself and the ReLU, but also to have a value as close as possible to zero, when $x < 0$. To achieve this, we minimized the MSE between the approximation and the ReLU with an added regularizer that penalizes negative values in the interval in which the function is being approximated, such as the one in equation 3, where $p(n)$ is the polynomial, and c is the value being optimized.

$$R = \sum_{n} max(-p(n) + c, 0)^2 \tag{3}$$

Using this method, the derivative was approximated using Python's *numpy* package, with a high degree Chebyshev polynomial, in the interval $[-120, 120]$. The resulting polynomial was integrated, and all coefficients of degree higher than two were dropped to keep the multiplicative depth of the network to a minimum, for the reasons stated above. To determine the constant coefficient Python's *scipy.optimize* package was used, with a sample of 10000 data points, in the interval $[-50, 50]$. This resulted in the final polynomial in Equation 4.

$$p(x) = 0.03664x^2 + 0.5x + 1.7056 \tag{4}$$

For classification tasks, NNs usually have an output activation function, with the Sigmoid being one of the most common. In view of this, using the same process and parameters as for the ReLU, we approximated the Sigmoid function with a 1^{st} degree polynomial, to keep the degree as low as possible, resulting in Equation 5.

$$y = 0.004997x + 0.5 \tag{5}$$

B. Experimental Setup

To compare the performance of networks with regular activation functions (referred to as NNs), and their encrypted counterparts using polynomial approximations (referred to as ENNs), experiments were conducted for three different speech affecting conditions, Cold, Depression and Parkinson's Disease. For Cold we used the Upper Respiratory Tract Infection Corpus (URTIC) [26] to perform a classification task. This corpus was used in Interspeech's 2017 ComParE Challenge [27]. The experiments concerning Depression were performed

using the Distress Analysis Interview Corpus - Wizard of Oz (DAIC-WOZ) [28], for both regression and classification, as in 2016's AVEC Challenge [29]. Finally for Parkinson's Disease (PD), the Parkinson's Disease Spanish Corpus was used for a regression task [30], which was also the target of the 2015's Interspeech ComParE Challenge [31].

Due to the fact that the datasets used were the same as the ones provided for several speech paralinguistic challenges, labels were only available for training and development sets. While these datasets could have been split to create new test sets, this was not done, so that our results could be compared with the baseline results of each challenge. A detailed description of each of the three datasets is provided in [26], [28], and [30].

C. Features

Two different feature sets were used in our experiments. For the experiments performed on the Depression and Cold datasets, the eGeMAPS feature set was used [32], a feature set developed in an effort to standardize baseline results in paralinguistic tasks. It includes 88 features, containing information on fundamental frequency, energy, spectral and cepstral characteristics. For the PD experiment, eGeMAPS did not yield significant results. Consequently, a Parkinson's Disease specific feature set was used instead. Developed by Pompili et al. [33], it contains 36 GeMAPS based features, along with 78 MFCC based features, resulting in a 114 dimensional feature vector. Both feature sets were extracted from audio files using openSMILE configuration files [34]. All features were zero-centered and normalized with unit variance using the mean and standard deviation computed from each training set.

D. Network Architecture and Training

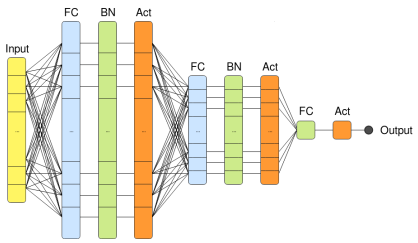

Fig. 2: Neural Network Architecture (adapted from [8])

The network used in our experiments is similar to the one used by Dias et al. [8], following the architecture represented in Figure 2, with the layers labeled FC being Fully Connected layers, BN, Batch Normalization layers and Act, Activation Layers. For classification tasks an output Activation layer was included after the third FC layer. During training, a dropout layer was also inserted, before the second and third FC layers. This serves as a regularizer, to help prevent the network from overfitting [35]. The BN layer already present in the architecture, also has a regularization effect [36], apart from assuring that the inputs of the activation layer are normally distributed.

The size of our network was limited to three sets of layers for two reasons: on one hand, the datasets used for our experiments are relatively small, and using larger networks did not yield any performance gains; on the other hand, adding another layer set (FC + BN + Activation) would require an increase in the encryption parameters, resulting in a higher computational toll.

In all the experiments, the first and second FC had 120 and 50 neurons, respectively, while the last FC layer only had 1 neuron. The values used for dropout probability were found through random search: 0.3746 and 0.5838 for the Cold; 0.092 and 0.209 for Depression; 0.877 and 0.246 for Parkinson's Disease.

While training for classification tasks, it was noted that the Cold and Depression training partitions were largely unbalanced. For this reason, weights were attributed to each class. For Depression a weight of 0.8 was attributed to positive samples, and 0.2 was given to negative samples. In Cold, the difference was larger, thus we gave weights of 0.9 and 0.1 to the positive and negative samples, respectively.

All networks were implemented and trained using Keras [37], using Tensorflow as backend. The predictions for the ENN were computed using the same networks trained on Keras with polynomial approximations, implemented in C++ using the SEAL library [24].

The models were trained with a learning rate of 0.02, 100 epochs with early stopping, and a weight decay of 0.005, using RMSProp. As loss functions, we used Binary Cross-Entropy (BCE) for classification and Mean Square Error (MSE) for regression. To keep track of the performance of each system during training, in regression tasks, the Mean Absolute Error (MAE) was computed. For classification tasks, due to class unbalance, the Unweighted Average Recall (UAR) function was implemented in Keras, and also computed during training.

E. Encryption Parameters

In our tests with SEAL, we used a polynomial modulus of 8192, and a plaintext modulus of 2^{30}. As for the coefficient modulus, we used SEAL's default value for a security level of 128 bits, and the polynomial modulus referenced above. To encode real numbers we used SEAL's Fractional Encoder, with a polynomial expansion base of 3, selecting 64 coefficients for the integer part of the value and 32 coefficients for the fractional part. With these encryption parameters our implementation takes on average 750 ms to encrypt the feature vector, 4.5 seconds to compute a prediction, and less than 5 ms to decrypt the result, using 24 Intel(R) Xeon(R) CPU E5-2630 v2 @ 2.60GHz processors, and SEAL's multithreading capabilities.

F. Results

Method	UAR(%)	F1 Score	Precision	Recall
Challenge Baseline	66.1	-	-	-
NN	66.9	.279 (.687)	.169 (.959)	.803 (.535)
ENN	66.7	.278 (.687)	.168 (.958)	.799 (.535)

TABLE I: Baseline and results obtained for Cold classification.

Method	UAR(%)	F1 Score	Precision	Recall
NN	60.6	.586 (.515)	.454 (.782)	.827 (.384)
ENN	60.2	.541 (.642)	.480 (.713)	.621 (.584)

TABLE II: Results obtained for Depression classification at the Segment level.

Method	RMSE	MAE
NN	7.43	5.80
ENN	6.77	5.64

TABLE III: Results obtained for Depression severity at the Segment Level.

In this section, we present the results relative to our experiments. Each table contains the results obtained for both the regular NN and the ENN, along with the baseline results of the challenge corresponding to each task, with the exception of the Depression task, where we do not include it, since it corresponds to results at the interview level, while our results in all tasks are relative to the segment level. Additionally, all results are relative to the Development set of each corpus.

For classification tasks we include the F1 Score, Precision and Recall for each class, in addition to the UAR, as they allow for a better understanding of how the model is behaving. When referring to these metrics, the values outside of the brackets are relative to the positive class (presence of the condition), while the ones corresponding to the negative class are inside brackets. The metrics for AVEC's regression baseline are the Root Mean Square Error (RMSE) and the MAE. For Interspeech's 2015 ComParE Parkinson's challenge the metric is Spearman's Correlation Coefficient (ρ), but we included the RMSE and MAE as well, for comparison purposes.

From Tables I, II, , III and IV we can see that the results are overall close to those of the baseline. In addition, there is only a slight performance degradation from the NNs to the ENNs. This is likely due to the polynomial activation functions, which have unbounded derivatives, as well as the output activation layer, which in the case of the NN is a Sigmoid while in the ENN it is a linear polynomial.

V. Privacy-preserving End-to-End Convolutional Neural Networks

Recent years have seen the development of end-to-end (E2E) NN architectures for speech applications [38]. Similar to what is accomplished in Image Processing, where raw images are given as inputs to a Convolutional Neural Network, the idea behind E2E schemes is to avoid using specialized feature sets and instead use raw audio, or some other low-level representation of the audio signal (i.e. Spectrograms and Filterbank Energies), so that the system can learn the most relevant features for the task at hand on its own.

The framework described in Figure 1, although useful, has several shortcomings. In particular, it requires the client to perform the feature extraction stage before sending his encrypted data to the service provider. This can be a setback for the client due to the computational expense, and to the service provider, as it discloses information about the model. As such, in this section we aim to determine whether E2E architectures can be applied in the context of HE, in order to avoid this problem.

Method	RMSE	MAE	ρ
Baseline	-	-	0.492
NN	16.6	12.6	0.507
ENN	16.0	12.5	0.450

TABLE IV: Baseline and results obtained for Parkinson's Disease.

A. End-to-end Convolutional Neural Network

Although many of the recent works in E2E speech processing suggest the use of Convolutional Long-Short Term Memory Deep Neural Networks (CLDNN) [38], this approach is not suited for HE. Recurrent layers encompass a large amount of operations and activation functions, which makes this approach unfeasible to compute them with HE. In addition, many speech end-to-end approaches use Raw audio as the input to their networks. This would be ideal to our objectives, however, the dimensionality of raw audio makes it an unsuitable format for HE. For these reasons, we decided to base our approach on the work of Milde et. al. [39], who employ a CNN with input log-Mel Filterbank Energies (FBEs). In this way, we can trade-off between time-frequency resolution and the dimensionality of these features. Moreover, since each file was split into several FBE frames, Milde et. al. [39] proposed the use of a weighted voting strategy, based on Ridge Regression, to combine the predictions for each frame. This type of regression is only a linear transformation of the original features, and as such, it is straightforward to compute it in a HE context, and it can be done in paralel with the CNN. Even though this approach is not entirely end-to-end, it still fulfills the objectives defined in the beginning of the section.

B. Encrypted Convolutional Neural Networks

1) Adaptation to Homomorphic Encryption: For Encrypted Convolutional Neural Networks (ECNNs), we can apply the same reasoning and methods presented in Section IV, for ENNs. However, some further adaptations are required. Convolutional Layers can be considered weighted sums, requiring additions and multiplications to be performed and, therefore, it is straightforward to adapt them to HE. Nevertheless, this is not the case for pooling layers. Even though it is one of the most commonly used pooling layers, the Max Pooling Layer is a nonlinear operation which is unsuited for HE. As an alternative, we can use a more HE-friendly layer, the Average Pooling Layer. Since this layer is essentially a weighted sum, it is simple to adapt to HE [9] [10] [11].

2) Weighted Prediction Strategies: A simple way to combine frame-level predictions is to compute the average of each prediction. Alternatively, for frames with different lengths, we can use the frame's length as a weight, and perform a weighted average. Another similarly straightforward method would be to use a majority vote. However, these methods do not take the frame's importance to the final prediction into account, as opposed to the method suggested by Milde et. al. [39], who train a Ridge Regression model for each class, using the

CNN's input features of the corresponding class, with labels defined as 0 if the prediction is incorrect, and as 1 if the prediction is correct. This way, the regressor will output a prediction closer to 0 if the frame is relevant, and close to 1 if it is not. This output can then be used to weigh the prediction of each class, to compute a Weighted Majority Vote.

C. Experimental Setup

The models used in our experiments were composed of two Convolutional layers (Conv 1 and Conv 2), each followed by a BN layer, an Activation layer and an AP layer. The output of the last Convolutional layer is turned into a vector, to enter the DNN stage of the network. This stage was composed by two FC layers (FC 1 and FC 2), each followed by a BN and an Activation layer, which in turn are followed by an output FC layer (FC 3). For classification tasks, a BN and an activation layer were added after the last fully connected layer. All experiments were implemented in Keras with Tensorflow backend and RMSprop with default parameters as the backpropagation algorithm, and were performed on the URTIC corpus [26].

1) Raw Audio: To establish a baseline for the CNN described in Section V-A, a first experiment was conducted using raw audio feature vectors. Each file was split into 3-second long segments, with the remaining split segments being zero-padded. Additionally, each segment was zero-centered and normalized to unit variance using its own mean and variance. To aggregate predictions for each segment, a weighted average was employed, using the original size of each segment as weight. The network used in this experiment was trained with Binary Cross-Entropy, a learning rate of 0.001 and 500 epochs, together with early stopping. The remaining network parameters used were as follows: Conv. 1 included 20 one-dimensional filters of shape (1,80). AP 1 is a 2-dimensional pooling layer, having a pooling filter and a stride with shape (2,10). Conv. 2 is composed of 40 2-dimensional filters with shape (160,2). AP 2 is a 2D pooling layer with a pooling filter and strides of shape (20,10). FC 1 and FC 2 consist of 200 units, while FC 3 is composed of only one unit. Before each FC layer, a Dropout layer was inserted during training, with a probability of dropout of 0.5.

2) Log-Mel Filterbank Energies: For the reasons presented in Section V-A, we chose FBEs as inputs to our ECNN, following the approach of [39]. As such, vectors of 40 log-Mel Filterbank Energies were computed from each file with 30 ms windows and 10 ms shift, using OpenSMILE [34]. From each of the resulting vectors, 11 x 40 feature maps were created with the ± 5 feature vectors around it. All data was zero-centered and normalized to unit variance, using the mean and variance of the training set. In order to combine predictions for frames of the same original file, the two majority voting techniques described in V-B2 were used, plus a simple average.

This model was trained with Categorical Cross-Entropy as the loss function, 0.01 learning rate, 500 epochs, a weight decay of 0.001 and early stopping. The remaining parameters were as follows: Conv 1 and Conv 2 layers were composed of 32 and 64 2-dimensional filters with shape (3,3) and (2,2), respectively. Furthermore each AP layer had a pool-size and strides with shape (2,2). FC 1 and FC 2 were composed of 500 units, while FC 3 had two output units. Before each FC layer, a Dropout layer was inserted during training, with 0.5 dropout probability. It is important to note that here the network has two outputs, to allow the use of Majority Voting with Ridge Regression.

3) Encryption Parameters and Performance: The networks of the experiments performed using encrypted features were implemented using SEAL, using the library's Fractional Encoding scheme, with an expansion base of 3 and 16 coefficients for both the integer and the fractional part. The values assigned to the encryption parameters were as follows: 16,384 for the polynomial modulus, 9,147,936,743,096,321 for the plaintext modulus and finally, the library's default value for the aforementioned polynomial modulus and 128-bit security was used for the coefficient modulus. Predictions were computed using multithreading with 24 Intel(R) Xeon(R) CPU E5-2630 v2 @ 2.60GHz processors, in a machine with 250 GB of RAM.

The implementation of the network described in this section takes 36 minutes to compute a single prediction. Considering this and that the development partition of the URTIC corpus is composed of a total of 2,624,391 frames, it would be unfeasible to compute the results with encrypted features. For this reason, all the results we present in Section V-D were computed with unencrypted data.

D. Results

In this section, we present our results for the three experiments described in Section V-C. For all experiments we give the average result of each metric for both classes, in percentage. This is done to favour the intelligibility of the results, and to facilitate the comparison of different systems.

Table V presents our results for the experiments conducted with Raw Audio, for a CNN and an ECNN (whose Activation functions were replaced by polynomials), together with the baseline result for a NN obtained in Section IV. All results correspond to the utterance level, and were obtained by applying the Weighted Average strategy described in Section V-C1, to frame-level predictions.

In Table VI and VII, we present the results for the NN baseline, the CNN, and the ECNN, that receive FBEs as inputs. Additionally, we display the results for each of the prediction combination strategies described in Section V-B2. Regarding nomenclature, SA corresponds to Simple Average, MV to Majority Vote, and WMV to Weighted Majority Vote, using the weights computed with Ridge Regression.

Although the UAR is in overall lower than the baseline, the values obtained for the F1 Score and Precision are higher, and show that the individual Recall and Precision for each class are much more balanced than those of the baseline. Comparing the prediction aggregation methods, there is no significant difference from the MV to the SA. However, for the NN using the WMV increases both the UAR and the F1 Score, when compared to the other strategies. On the other hand, for the ENN the WMV method produces the lowest results. In addition, the results obtained with FBEs are never as high as the ones obtained with raw audio. This was to be expected,

Method	UAR(%)	F1 Score (%)	Precision (%)
NN	66.9	48.3	56.4
CNN	66.1	56.3	70.7
ECNN	65.0	55.9	66.7

TABLE V: Results obtained using Raw Audio

Method	UAR(%)	F1 Score(%)	Precision(%)
NN Baseline	66.9	48.3	56.4
CNN + SA	60.7	61.7	63.0
CNN + MV	60.8	61.7	63.0
CNN + WMV	62.6	62.2	61.8

TABLE VI: Results obtained using FBEs.

since FBEs contain information for a small fraction of the utterance. In addition, it is important to note that the values obtained with encrypted predictions were mostly wrong. Since each layer was tested individually, with correct results, this was most likely caused by the Fractional Encoding strategy, due to overlapping terms of each part, caused by a large amount of multiplications [24].

VI. Privacy-preserving Discretized Neural Networks

Many speech processing tasks require single utterances to be split into several frames, due to their variable (and usually high) dimensionality, as well as to trade-off between precision and the time-span covered by the features computed from them. In addition, some HE schemes allow several values to be batched into a single ciphertext, which can be operated on as *SIMD (Single Instruction, Multiple Data)*, with a technique called *batching*. Since most operations on NNs are vectorizable, *batching* would allow several encrypted predictions to be made at the same time. Nevertheless, this technique is incompatible with other encoding strategies, and requires that all values be integers. Moreover, in NNs both the weights and inputs are real numbers. The goal of the experiments in this section is thus to determine how a network and its inputs can be discretized, in order to make them compatible with *batching*.

A. Discretized Neural Networks

Works such as Cryptonets [9] and Hesamifard et al. [11] make use of batching techniques, and discretize the weights of their networks by scaling them up to a fixed precision. In a different approach, encrypted Binarized Neural Networks have been implemented by [12], while [13] implemented an ENN with real valued weights directly converted to integers. However, this approach gives origin to a large accuracy drop, as is stated by the authors [13]. In addition, both these works use a FHE which does not allow batching.

Having the above in mind, in this experiment we simply scale each weight of the network by a factor. However, this has to be done with care, as the activation layers being used in our encrypted networks are 2^{nd} degree polynomials. If a weight is multiplied by a scaling factor, when the weight passes through this layer, the factor will be squared. As such, propagating scaled values through the network will result in a very rapid growth of the scaling factor. In an encrypted context, if the values grow too large, they may be reduced modulo the plaintext modulus. Thus the scaling factor has to be selected considering the largest absolute value allowed by the plaintext modulus. To avoid having this problem with features, we chose to quantize them instead, allowing them to be directly employed with the ENN.

Method	UAR(%)	F1 Score(%)	Precision(%)
ENN Baseline	66.7	48.3	56.3
ECNN + SA	59.1	64.0	60.4
ECNN + MV	59.2	64.4	60.5
ECNN + WMV	57.4	63.3	58.2

TABLE VII: Results obtained using FBEs with Polynomial Activation functions.

B. Experimental Setup

To determine the ideal number of quantization channels 2^n (or quantization bits) and the scaling factor s we performed several experiments. As a first step, we trained a baseline NN for different feature quantization values. From this, we selected the model with the best results, and tested different scaling factors, to determine which achieved the best trade-off between model accuracy and largest absolute value in the network. Finally, using these values, the networks described in Section IV were retrained and re-implemented in SEAL using batching.

1) Implementation: For the reasons stated in Section IV, between each pair of FC and Activation layers, our encrypted networks include a Batch Normalization Layer. Since these layers are both linear transformations, they can be easily collapsed:

$$y = \gamma \frac{(A \cdot x + b) - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta = A'x + b' \tag{6}$$

In this way, the initial input of the polynomial activation layer, only needs to be scaled by a factor s. However, the activation layer, being a polynomial, includes multiplicative coefficients, which also need to be scaled. Therefore, to avoid increasing the degree of the scaling factor, we can instead pre-multiply these constants by the weights of the previous layers. More specifically, the square root of the 2^{nd} degree coefficient of the polynomial can be put inside the square, and be pre-multiplied by the weights of Equation 6. This way, we only need to multiply the square root of this coefficient by the 1^{st} degree coefficient of the activation function, and we obtain:

$$y_{act} = y_{fc+bn}^2 + (s\frac{c_1}{\sqrt{c_2}})y_{fc+bn} + s^2 c_0 \tag{7}$$

where c_0, c_1 and c_2 are the coefficients of the polynomial activation function. Alternatively, instead of pre-multiplying the constant coefficients, we can expand the squared term of the polynomial, and pre-compute every constant and obtain:

$$y_{fc+bn+act} = (A''_I \cdot x_{in})^2 + B''_I \cdot x_{in} + c''_I \tag{8}$$

where A and B are matrices with the same dimensions, of combined pre-computed weights, and c is the vector of the remaining constant terms. Neverthless, this form requires

two matrix dot products, doubling the number of operations required by Equation 7. From these equations, we get that the degree of the scaling factor at the exit of the first FC+BN+Act layer set will be 3, for the second layer it will be 6, for the third layer 14, and so on. As such, considering that in SEAL, the maximum value for the plaintext modulus t is 2^{60}, if we have a scaling factor of 100, the depth of the network can be at most 2, otherwise during the encrypted inference values will undergo reduction modulo t.

2) μ-Law Quantization: To quantize input features, we first used the μ-Law companding transformation, displayed in Equation 9, which maps inputs from $[-1, 1]$ to $[0, 1]$.

$$f(x) = sign(x)\frac{ln(1+\mu|x|)}{ln(1+\mu)} \tag{9}$$

Subsequently, to map the outputs of the function to $[0, \mu-1] \in \mathbb{Z}$, where μ is the number of quantization channels, we used the quantization function in Equation 10.

$$Q(x) \;=\; \left\lfloor \mu\frac{f(x)+1}{2} + \frac{1}{2} \right\rfloor \tag{10}$$

Additionally, to promote smaller values, each feature vector was zero-centered relative to its median value.

3) Language Verification Task - KALAKA-2 Dataset: To ensure the validity of the conclusions taken from the results of our experiments, we decided to use an additional dataset, besides the ones used in the previous sections. The reason for this was that these are very small and a larger dataset adds strength to the results obtained. For this purpose we selected the KALAKA-2 corpus [40], to perform a Language Verification Task. The recordings in this dataset are divided into three partitions, Training, Development and Evaluation, with those in the Development and Evaluation sets divided into 3, 10 and 30 second length files. In addition the dataset is composed of 6 languages: Spanish, Catalan, Basque, Galician, Portuguese and English. Results for this dataset are reported in terms of Accuracy together with a metric developed specifically for this task, C_{avg} [41]. To compute this metric, we used MATLAB R2017b [42], together with the FoCal v1.0 package [43].

To train models with this dataset, we followed the approach described in [44], and used a feature vector of Shifted Delta Cepstra (SDC) computed from 7 Perceptual Linear Prediction coefficients with log-RelAtive SpecTrAl speech processing (PLP-RASTA). The PLP-RASTA were computed from each segment with 25 ms windows and 10 ms shift, using openSMILE [34], and were zero-centered and normalized to unit variance. The SDC features were computed from the PLP-RASTA, with a 7-1-3-7 configuration, resulting in a 56 dimensional feature vector. These were later aggregated in groups of 10 feature vectors, resulting in a final vector of 560 components. Additionally, low energy frames were filtered out. The resulting predictions for each recording were combined by computing the average log-posterior estimate of each class.

4) Neural Network Architecture and Training: The network architecture for the experiments with KALAKA-2 was composed of two sets of two FC layers, each with 2048 nodes, followed by a polynomial activation function. The final FC layer included 6 nodes, and was followed by a Softmax Activation layer. Before each FC layer, with the exception of the first, a Dropout layer was inserted with 0.5 dropout probability. Although the Softmax cannot be implemented in HE, it helps increase the performance of the network. For this reason, we decided to trade-off between the network's accuracy, the privacy of the model, and the computational toll on the client's side, and assume the client can compute it, along with the log-posterior probabilities. This network was implemented and trained with Keras, using Tensorflow as backend, with early stopping, a learning rate of 0.0005, 5,000 epochs, the Categorical Cross-Entropy loss function, and using Adam with default parameters as the optimizer.

5) Encryption Parameters: For batching to work correctly, it is necessary that the plaintext modulus t be a prime number, congruent to 1 $mod\ 2n$, with n being the polynomial modulus [24]. To account for the large values that arise from scaling the network's weights we chose a plaintext modulus larger than 2^{59}, that fulfills the condition defined above. The polynomial modulus was chosen to be 16384, and in addition, for the coefficient modulus, SEAL's default value for a security level of 128 bits was selected. The encrypted predictions were computed using multithreading with 24 Intel(R) Xeon(R) CPU E5-2630 v2 @ 2.60GHz processors, in a machine with 250 GB of RAM.

Metric	Development			Evaluation		
	3 (s)	10 (s)	30 (s)	3 (s)	10 (s)	30 (s)
Accuracy (%)	66.5	81.1	87.2	57.0	68.0	71.8
100×C_{avg}	10.7	5.92	3.58	17.58	13.86	13.03

TABLE VIII: Baseline NN results for KALAKA-2.

C. Results

This section presents the results obtained in the experiments detailed in the previous section. In all tables, the acronym QNN refers to results obtained with quantized features, and Scaled QNN to results obtained with networks scaled according to the method described in the previous section. Of these scaled results, only the ones present in Table X were computed using encrypted data, due to the size and computational expense of the network used for KALAKA-2. The baseline results obtained for KALAKA-2 are presented in Table VIII.

Our preliminary experiments showed that with 4-bit feature quantization and a scaling factor of $s = 150$, we were able to obtain the best trade-off between accuracy, and the maximum absolute value of the network's computations. With this factor the largest value in the network never surpasses 2^{55}. Using these values we obtained the results displayed in Tables IX and X, for the KALAKA-2 corpus, with 30 seconds-long files, as well as for the three paralinguistic datasets used in Section IV, respectively. These tables show that the scaled versions of the network do not entail a large performance degradation, which validates this approach.

It is important to note, however, that the results for the Paralinguistic tasks were obtained with Equation 8, whereas the results for the LV task were obtained using Equation 7.

Method	Development		Evaluation	
	Accuracy (%)	100×C_{avg}	Accuracy (%)	100×C_{avg}
Baseline NN	87.2	3.57	71.8	13.0
QNN	84.9	4.18	75.9	10.3
Scaled QNN	85.3	4.45	73.9	11.0

TABLE IX: Results for the KALAKA-2 Corpus with 30-seconds-long utterances.

The reason for this is a large percentage of weights in the Paralinguistic task's networks were lower than 0.01. Since the scaling factor was set to 150, most were set to 0, resulting in incorrect results, which did not occur with Equation 8. For the LV task, this was not the case, and the computationally faster equation was used. The encrypted implementations of these networks take 23 minutes, and 23 seconds, for KALAKA-2's scaled network and for the scaled network developed for the paralinguistic tasks, respectively. Since 16,384 predictions can be made at the same time, the effective value for a prediction is 84.23 milliseconds for the Language Recognition task and 1.40 milliseconds for the paralinguistic tasks.

VII. Conclusions and Contributions

In this work we demonstrated that it is possible to build secure frameworks for health-related *paralinguistic* speech tasks with minimal accuracy degradation at the cost of a significant computational overhead.

In Section IV, we provided a proof-of-concept on how an NN can be adapted to work with a LHE scheme, receiving as input an encrypted feature vector, and outputting an encrypted prediction. This network obtained results with minimal degradation when compared to a regular NN, and our implementation was able to perform a single prediction in 4.5 seconds. Since the user's input data vector is encrypted neither the server nor malicious third parties can learn anything from the user's data. Nevertheless, this scheme relies on the user to perform a feature extraction stage, which provides the user with some information about the model and can become a limitation for devices with low computational power.

The experiments of Section V focused on minimizing the user's role on the framework. Although our initial objective was to remove the feature extraction stage altogether, and use raw audio as input to our network, this proved unfeasible due to the dimensionality of this data format. Instead we chose to trade-off between the amount of computations on the user's side and the model's performance, selecting log-Mel Filterbank Energies as a low-level and low-dimensional representation of the input. Although this approach improved the overall performance of the network, a single prediction took 36 minutes to perform, besides yielding incorrect results. Thus, this method cannot be considered suitable for a real world application. Nonetheless, the methods and reasoning used in this Section regarding dimensionality, and frame-posterior aggregation, can serve as a base for future related works to build upon.

In the final experiment of Section VI, we focused on transforming the network and its inputs into integer values, using feature quantization and weight scaling, in order to apply an LHE *batching* technique. Our results showed that this can be done with minimal accuracy degradation, and we were able to perform 16384 simultaneous predictions. However, this comes at the cost of having a maximum number of 2 activation layers in the network, and an increase in computational cost, as a single batch takes around 23 seconds to compute. Nevertheless, the effective time cost of a single prediction is 1.4 milliseconds as opposed to the original approach where a prediction took on average 4.5 seconds to compute.

VIII. Future Work

The frameworks used in this work have several advantages. The first is the relative ease with which one can implement an NN using LHE. Using already existing libraries and a previously trained model, that abides by the constrains of LHE, transforming an NN into an ENN requires only the replacement of each function with its homomorphic counterpart. The second advantage is the fact that this framework does not require the user to remain online while the computation is being performed. A third advantage is the fact that the architecture of the model is hidden, or at least partially hidden. Nevertheless, it poses some limitations. Although this framework does not induce a direct accuracy degradation from the activation functions, it induces it indirectly by restricting the depth of the network to a few layers, due to computational restraints. In addition, HE limits the types of layers that can be used, which might lead to worse model performances.

Taking these considerations into account, topics for future work should include

- Researching alternative weight and feature discretization techniques. These can be applied both in the frameworks implemented in this thesis and in the approach developed by [13].
- Applying privacy-preserving model training techniques to *paralinguistic* speech tasks.
- Applying HE to other ML algorithms commonly used for speech tasks.
- Further research GC and OT based PPML frameworks, to determine if they are suitable in the context of speech *paralinguistic* tasks.
- Improving results, by developing privacy-preserving frameworks that adapt specific methods used in health-related *paralinguistic* tasks.

Further research on this topic will allow the development of more accurate and computationally efficient private frameworks, that, in the future, may become real world tools to improve the quality of life of people living with speech-affecting conditions.

References

[1] B. Schuller, S. Steidl, A. Batliner, F. Burkhardt, L. Devillers, C. Möller, and S. Narayanan, "Paralinguistics in Speech and Language State-of-the-art and the Challenge," *Computer Speech & Language*, vol. 27, no. 1, pp. 4 – 39, 2013, special issue on Paralinguistics in Naturalistic Speech and Language.

[2] R. Bost, R. Ada Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," *NDSS*, 01 2015.

[3] M. Dias, "Privacy-preserving speech emotion recognition," Master's thesis, Instituto Superior Técnico, 2017.

Method	Cold			Depression (Class.)			Depression (Regr.)		Parkinson's Disease		
	F1 Score	Precision	Recall	F1 Score	Precision	Recall	RMSE	MAE	RMSE	MAE	ρ
Baseline ENN	48.3	56.3	66.7	59.2	59.7	60.2	6.77	5.64	16.0	12.5	0.45
QNN	53.0	56.7	66.8	60.3	60.2	60.6	6.74	5.62	15.9	12.6	0.53
Scaled QNN	50.2	56.5	66.9	59.8	60.0	60.5	6.67	5.63	15.8	12.6	0.52

TABLE X: Batching results for Paralinguistic Tasks

[4] J. Portêlo, "Privacy-preserving frameworks for speech mining," Ph.D. dissertation, Instituto Superior Técnico, 2015.

[5] M. A. Pathak and B. Raj, "Privacy-Preserving Speaker Verification and Identification Using Gaussian Mixture Models," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 2, pp. 397–406, Feb 2013.

[6] J. Portêlo, A. Abad, B. Raj, and I. Trancoso, "Secure Binary Embeddings of Front-end Factor Analysis for Privacy Preserving Speaker Verification," in *INTERSPEECH*, 2013, pp. 2494–2498.

[7] A. Jiménez, B. Raj, J. Portêlo, and I. Trancoso, "Secure Modular Hashing," in *Information Forensics and Security (WIFS), 2015 IEEE International Workshop*. IEEE, 2015, pp. 1–6.

[8] M. Dias, A. Abad, and I. Trancoso, "Exploring Hashing and Cryptonet based Approaches for Privacy-preserving Speech Emotion Recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2018 IEEE International Conference*. IEEE, 2018.

[9] R. Gilad-Bachrach, N. Dowlin, and K. Laine et al., "CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy." in *ICML*, ser. JMLR Workshop and Conference Proceedings, vol. 48, 2016, pp. 201–210.

[10] H. Chabanne, A. de Wargny, J. Milgram, and C. Morel et al., "Privacy-Preserving Classification on Deep Neural Network." *IACR Cryptology ePrint Archive*, vol. 2017, p. 35, 2017.

[11] E. Hesamifard, H. Takabi, and M. Ghasemi, "CryptoDL: Deep Neural Networks over Encrypted Data." *CoRR*, vol. abs/1711.05189, 2017.

[12] A. Sanyal, M. J. Kusner, A. Gascón, and V. Kanade, "Tapas: Tricks to accelerate (encrypted) prediction as a service," in *ICML*, 2018.

[13] F. Bourse, M. Minelli, M. Minihold, and P. Paillier, "Fast homomorphic evaluation of deep discretized neural networks," *IACR Cryptology ePrint Archive*, vol. 2017, p. 1114, 2017.

[14] B. D. Rouhani, M. S. Riazi, and F. Koushanfar, "Deepsecure: Scalable provably-secure deep learning," *CoRR*, vol. abs/1705.08963, 2017.

[15] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 19–38.

[16] J. Liu, M. Juuti, Y. Lu, and N. Asokan, "Oblivious neural network predictions via minionn transformations," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17, 2017, pp. 619–631.

[17] M. S. Riazi, C. Weinert, O. Tkachenko, E. M. Songhori, T. Schneider, and F. Koushanfar, "Chameleon: A hybrid secure computation framework for machine learning applications," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*. ACM, 2018, pp. 707–721.

[18] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Sept 2015, pp. 909–910.

[19] A. Acar, H. Aksu, A. Selcuk Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Computing Surveys*, vol. 51, p. 79, 04 2017.

[20] C. Gentry, "A Fully Homomorphic Encryption Scheme," Ph.D. dissertation, Stanford University, 2009.

[21] F. Armknecht, C. Boyd, C. Carr, K. Gjøsteen, A. Jäschke, C. A. Reuter, and M. Strand, "A guide to fully homomorphic encryption," *IACR Cryptology ePrint Archive*, vol. 2015, p. 1192, 2015.

[22] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) Fully Homomorphic Encryption without Bootstrapping," *ACM Trans. Comput. Theory*, vol. 6, no. 3, pp. 13:1–13:36, Jul. 2014.

[23] J. Fan and F. Vercauteren, "Somewhat Practical Fully Homomorphic Encryption." *IACR Cryptology ePrint Archive*, vol. 2012, p. 144, 2012, informal publication.

[24] K. Laine, H. Chen, and R. Player, "Simple Encrypted Arithmetic Library - SEAL v2.3.1," https://www.microsoft.com/en-us/research/publication/simple-encrypted-arithmetic-library-v2-3-0/, Microsoft, Tech. Rep., December 2017.

[25] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[26] J. Krajewski, S. Schnieder, and A. Batliner, "Description of the Upper Respiratory Tract Infection Corpus (URTIC)." in *INTERSPEECH*, 2017.

[27] B. Schuller, S. Steidl, A. Batliner, and E. Bergelson et al., "The INTERSPEECH 2017 Computational Paralinguistics Challenge: Addressee, Cold & Snoring," in *Computational Paralinguistics Challenge (ComParE), Interspeech 2017*, 2017.

[28] J. Gratch, R. Artstein, G. M. Lucas, and G. Stratou et al., "The Distress Analysis Interview Corpus of human and computer interviews." in *LREC*, 2014, pp. 3123–3128.

[29] M. F. Valstar, J. Gratch, B. W. Schuller, and F. R. et al., "AVEC 2016 - Depression, Mood, and Emotion Recognition Workshop and Challenge," *CoRR*, vol. abs/1605.01600, 2016.

[30] J. R. Orozco-Arroyave, J. D. Arias-Londoño, J. F. V. Bonilla, and M. C. Gonzalez-Rátiva et al., "New Spanish Speech Corpus Database for the Analysis of People Suffering from Parkinson's Disease." in *LREC*, 2014, pp. 342–347.

[31] B. W. Schuller, S. Steidl, A. Batliner, and S. Hantke et al., "The INTERSPEECH 2015 Computational Paralinguistics Challenge: Nativeness, Parkinson's & Eating Condition." in *INTERSPEECH*, 2015, pp. 478–482.

[32] F. Eyben, K. Scherer, B. Schuller, and J. Sundberg 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, 4 2016, open access.

[33] A. Pompili, A. Abad, P. Romano, and I. P. Martins et al., "Automatic Detection of Parkinson's Disease: An Experimental Analysis of Common Speech Production Tasks Used for Diagnosis." in *TSD*, ser. Lecture Notes in Computer Science, vol. 10415, 2017, pp. 411–419.

[34] F. Eyben, F. Weninger, F. Gross, and B. Schuller, "Recent developments in openSMILE, the Munich Open-source Multimedia Feature Extractor," in *Proceedings of the 21st ACM International Conference on Multimedia*, ser. MM '13, 2013, pp. 835–838.

[35] N. Srivastava, G. E. Hinton, A. Krizhevsky, and I. Sutskever et al., "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[36] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." *CoRR*, vol. abs/1502.03167, 2015.

[37] F. Chollet *et al.*, "Keras," https://github.com/keras-team/keras, 2015.

[38] 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 *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 5200–5204.

[39] B. Milde and C. Biemann, "Using representation learning and out-of-domain data for a paralinguistic speech task," in *INTERSPEECH*, 2015.

[40] L. J. Rodríguez-Fuentes, M. Peñagarikano, A. Varona, M. Díez, and G. Bordel, "Kalaka-2: a tv broadcast speech database for the recognition of iberian languages in clean and noisy environments," in *LREC*, 2012.

[41] L. J. Rodriguez-Fuentes, M. Penagarikano, A. Varona, M. Diez, and G. Bordel, "The albayzin 2010 language recognition evaluation." in *Proceedings of Interspeech*, 01 2011, pp. 1529–1532.

[42] MATLAB, *version 7.10.0 (R2010a)*. Natick, Massachusetts: The MathWorks Inc., 2010.

[43] N. Brummer, "Focal toolkit v1.0," https://sites.google.com/site/nikobrummer/focalmulticlass, 2007.

[44] A. Abad, O. Koller, and I. Trancoso, "The l2f language verification systems for albayzin-2010 evaluation," in *Proceedings FALA*, 2010.