Performance evaluation and implementations of MFCC, SVM and MLP algorithms in the FPGA board

Original Scientific Paper

Salaheddine Khamlich
National School of Applied Sciences, Research teams "SEIA" LaSTI, Sultan Moulay Slimane University, KHOURIBGA, Morocco
s.khamlich@usms.ma

Fathallah Khamlich
LTI Lab. Faculty of Sciences Ben M'sik
Hassan II University, Casablanca- Morocco
khamlich.fathallah@gmail.com

Issam Atouf
LTI Lab. Faculty of Sciences Ben M'sik
Hassan II University, Casablanca- Morocco
issamatouf@yahoo.fr

Mohamed Benrabh
LTI Lab. Faculty of Sciences Ben M'sik
Hassan II University, Casablanca- Morocco
benrabh@yahoo.fr

Abstract *– One of the most difficult speech recognition tasks is accurate recognition of human-to-human communication. Advances in deep learning over the last few years have produced major speech improvements in recognition on the representative Switch-board conversational corpus. Word error rates that just a few years ago were 14% have dropped to 8.0%, then 6.6% and most recently 5.8%, and are now believed to be within striking range of human performance. This raises two issues - what is human performance, and how far down can we still drive speech recognition error rates? The main objective of this article is the development of a comparative study of the performance of Automatic Speech Recognition (ASR) algorithms using a database made up of a set of signals created by female and male speakers of different ages. We will also develop techniques for the Software and Hardware implementation of these algorithms and test them in an embedded electronic card based on a reconfigurable circuit (Field Programmable Gate Array FPGA). We will present an analysis of the results of classifications for the best Support Vector Machine architectures (SVM) and Artificial Neural Networks of Multi-Layer Perceptron (MLP). Following our analysis, we created NIOSII processors and we tested their operations as well as their characteristics. The characteristics of each processor are specified in this article (cost, size, speed, power consumption and complexity). At the end of this work, we physically implemented the architecture of the Mel Frequency Cepstral Coefficients (MFCC) extraction algorithm as well as the classification algorithm that provided the best results.*

Keywords *– Automatic speech recognition, Real time, SVM, MLP, ANN, MFCC, FPGA.*

1. INTRODUCTION

Automatic speech recognition is a computer technique that analyzes speech picked up by a microphone and transcribes it into machine-readable text. The first speech recognition system was created in 1952 [1]. This electronic system, developed by Davis, Biddulph and Balashek at Bell Labs [2], was essentially composed of relays and its performance was limited to recognizing isolated digits [1]. Research has been then considerably increased during the 1970s by IBM (1972-1993). Nowadays, speech recognition is related to many different areas of science: automatic language processing, linguistics, information theory, signal processing, neural networks, artificial intelligence, etc. In the field of speech processing, most algorithms require more complex mathematical operations and a very rich database. This field also requires the use of fast electronic circuits, reconfigurable for the training phase and capable of manipulating large amounts of information generated by the audio source. The main objective of this paper is to develop techniques for the software and hardware implementation of speech recognition algorithms and to compare these performances.

In the first part of our study, we propose two methods to design a robotic system capable of communicating with users. The first method is intended for the task of pre-processing (extraction of audio signals) and the second one consists in performing the task of speech recognition of the audio signal just after the extraction of the useful signal. The algorithms proposed to design this system are the MFCC algorithm for extracting the speech signals, and the Support Vector Machine (SVM) and Artificial Neural Network (ANN) algorithms for the classification of the speech signals of the names of teachers in our institution obtained after the MFCC extraction.

The objective of this work is the evaluation of performance (strengths and limitations encountered during the use of such a model) between ANN of type MLP and SVM (one against all and one against one) in the field of speech recognition. The three algorithms mentioned above require a very fast processor to perform the tasks of processing, coding, extraction and recognition. The second Part presented in this paper is limited to the chain of audio signal acquisition, processing and restitution of an audio signal provided by a microphone. We will also show how to implement an electronic system called NIOSII capable of processing audio signals on the DE2-70 FPGA (*Field Programmable Gate Array*) board. We have proposed this version of the new generation of reconfigurable embedded processors, which can be created by modifying the internal structure of the FPGA circuit to implement the algorithms studied in part 1. At the end of this work, we compared our implementation with traditional digital implementations [3], [4] and [5].

2. WORK LITERATURE REVIEW

In this part, we will point out some references who have treated subjects approaching the hardware and / or software implementations of object recognition algorithms in general and speech recognition in particular. They are based on classical methods for hardware implementation.

The study discussed in [6] demonstrated an HMM-free approach to training a speech recognizer which uses a neural network to directly predict transcript characters given the audio of an utterance. This approach discards many of the assumptions considered in modern HMM-based LVCSR systems in favor of treating speech recognition as a direct sequence transduction problem. The approach trains a neural network using the connectionist temporal classification (CTC) loss function, which amounts to maximizing the likelihood of an output sequence by efficiently summing over all possible input-output sequence alignments. Using CTC the authors were able to train a neural network to predict the character sequence of test utterances with a character error rate (CER) under 10% on the Wall Street Journal LVCSR corpus. While impressive in its own right, these results are not yet 1 competitive with existing HMM-based systems in terms of word error rate (WER).

In [7], the authors gave an overview of the hardware architectures of reconfigurable computing machines, and the software that targets these machines, such as compilation tools. They described two main methods of traditional computer science for running algorithms. The first is to use an application specific integrated circuit, or ASIC, to perform operations in the hardware. Because these ASICs are designed specifically to perform a given calculation, they are very fast and efficient when performing the exact calculation for which they were designed.

The study established in [8] describes an implementation of an ANN based on the FPGA circuit, of the FAST architecture (Flexible Topology Adaptable-Size), an artificial neural network (ANN) which dynamically adapts its size. Most ANN models base their ability to adapt to problems on changing the strength of the interconnections between computational elements based to a given learning algorithm. However, constrained interconnection structures can limit this capacity. Programmable Peripherals, are very well suited to the implementation of ANNs with an adaptation circuit structure. To achieve this implementation, in this study they authors used a network of Labomat-3 (a reconfigurable platform developed in their lab), which communicates with each other using TCP / IP or a hardware connection.

In 2005 D. Verstracten et al. [9] used an analog neuron to build an RC (reservoir computing) system on FPGA, using stochastic neurons which communicate using random bit streams instead of fixed-point values. This greatly simplified the hardware implementation of arithmetic operations such as addition, non-linearity, and multiplication.

In 2002, SJ Melnikoff et al. [10], investigated the FPGA efficiency to implement a decoder based on Continuous Hidden Markov Models (HMM) representing monophones, and demonstrated that it can process speech 75 times in real time. Recurrent neural networks are generally difficult to use because there is no practical learning program. In [11], a recursive learning scheme for recurrent neural networks has been developed based on simultaneous disturbance method. Unlike ordinary correlation learning, this method is applicable to analog learning and learning of oscillatory solutions of recurrent neural networks. They physically implemented Hopfield neural networks using an FPGA.

In [12], the authors showed mathematical basis of stochastic neurons as well as specific circuits necessary to implement the processing of each neuron. They also proposed a new methodology to reproduce the nonlinear activation function. Special MATLAB toolkits are used in this work for the training and execution of neural networks [13].

In [14] the authors proposed a new hardware / software system for the implementation of ANN with two hidden layers. The first layer has three neurons and the second has two neurons, and an output layer has two neurons. Learning ANN is done using MATLAB software (the design of ANN in Simulink). The hardware synthesis of the proposed algorithm, is performed by the Generator System. Routing and implementation are done on Spartan2E type FPGAs.

In [15] a novel design for a hyperbolic tangent activation function (Tanh) has been proposed to be used in memristor-based neuromorphic architectures. The purpose of the implementation of a CMOS-based design for Tanh is to decrease power dissipation and area usage. This design also increases the overall speed of computations in ANNs, while keeping the accuracy in an acceptable range.

Another big theme in semi-supervised learning is consistency [16], [17]. In this line of research, a consistency-based task that can be trained on unlabeled data is introduced and used to pre-train networks so that they can learn a good representation of the data. These networks are in turn fine-tuned on supervised data [18], [19].

In this article we will demonstrate the advantages of our hardware implementation based on a new generation of reconfigurable processors based on FPGAs with old implementations.

3. MEL FREQUENCY CEPSTRAL COEFFICIENTS ALGORITHM (MFCC), SVM AND ARTIFICIAL NEURAL NETWORKS (ANN):

Each model of speech recognition has advantages and disadvantages. In this part, we will summarize the strengths and limitations encountered while using such a model in ASR [15], [20].

GMM (Gaussians Mixture Model):

Advantages:

- The use of a mixture of several multi-dimensional Gaussian densities allows to give a very good representation of the acoustic vectors.
- The use of the GMM model allows to accurately estimate random probability densities such as that of the acoustic vectors.
- The learning time is relatively small compared to other models such as the HMM model.

Disadvantages:

- Although they are capable of capturing a speaker's longer-term information, they do not contain dynamic aspects. For a good modeling (i.e. a lot of Gaussians) require a lot of data.

DTW (dynamic time waping):

Advantages:

The DTW algorithm is fast, well adapted to speech because it is able to take into account the temporal variations of the signal. It does not require a lot of data to run and function properly.

Disadvantages:

- DTW is very sensitive to signal segmentation. Indeed, if the starting point of the dynamic calculation is not good, the algorithm can quickly deviate from the optimum path.

HMM (Hidden Markov Model):

Advantages:

- This technology offers powerful algorithms for learning and recognition, thanks to which HMMs have been shown to be best adapted to speech recognition problems.

Disadvantages:

- Hidden Markov models present certain limitations and difficulties, which lie mainly in the choice of a good initial model for learning, that is generally random and often leads to a local optimum.

SVM (Support Vector Machines):

Advantages:

- The big advantage, over other techniques, is the ability to generalize the classification. -This method is suitable for applications with a large intra-class variation.
- SVMs have a more efficient behavior for a very small training set.

Disadvantages:

- The disadvantage of SVMs is the empirical choice of the kernel function suited to the problem.
- A second drawback is the computation time which increases as a cubic function of the number of data to be processed.

ANN (Artificial Neural Network):

Advantages:

- Neural networks are useful for the classification of static patterns.

Disadvantages:

- Neural networks cannot model long term time evolution.
- Neural networks cannot model long-term temporal evolution. They are therefore poorly adapted to the processing of sequential signals such as speech.

In the following section, we will study the SVM Support Vector Machines and the MLP Neural Network, as well as the problem of classification when we have several classes. Finally, we have implemented and tested the MLP and SVM by NIOSII processors created in the FPGA board for the speech recognition application.

3.1. EXTRACTION OF VOICE SIGNALS AND CREATION OF A DATABASE BY THE MFCC ALGORITHM:

In our study, we used the MFCC speech signal extraction algorithm [5], [21]. The speech signal is represented, in general, in the frequency domain showing the temporal evolution of its spectrum. MFCC coding is based on the analysis by a Mel-scale filter bank to produce the MFCC cepstral parameters. In this algorithm, first the Discrete Fourier Transform (DFT) is used to calculate the frequency spectrum of the signal, and then the Discrete Cosine Transform (DCT) is used to further reduce redundant information in the speech signal. DFT and DCT can be used for any speech segment with a fixed time-frequency resolution. Equation (1) allowing the hertz to Mel is the most widely used [22] (it is a

common model for the relationship between frequencies in Mel scales and linear frequencies):

$$mel(f) = 2595 \times log_{10}(1 + \frac{f_{Linear\ frequency}}{700}) \quad (1)$$

For each tone with a real frequency f, measured in Hz, a subjective terrain is measured on a scale called the Mel scale. The pitch of a 1 kHz tone, 40 dB above the perceptual hearing threshold, is defined as 1000 mels. At the level of creating our voice database, we break down the database of teachers' names into two parts, the learning part and the test part. In our work the database consists of 1800 names pronounced by 24 people. Each person recorded 5 times the names of each teacher (each speaker pronounced 5 times each of the names of the teachers - see table 1). We used a total of 1125 names (15 teachers*75 times) for training and 675 names for testing. These numbers correspond to all the professors' names of the Electrical Engineering department of ENSAK school, so there were about 75 training data points and 45 test data points per professor.

Table 1. Number of examples for each class

NAMES (Classes)	Learning phase	Test phase
C1 : AILANE	75	45
C2 : AMHARECH	75	45
C3 : ATOUF	75	45
C4 : BENDAOUD	75	45
C5 : CHIKH	75	45
C6 : ELBARBRI	75	45
C7 : ERRACHID	75	45
C8 : EL JOURMI	75	45
C9 : HAMDOUN	75	45
C10 : KADIRI	75	45
C11 : KHAMLICH	75	45
C12 : LOKRITI	75	45
C13 : MAAIDER	75	45
C14 : MASSOUR	75	45
C15 : RHOFIR	75	45
Total	**1125**	**675**

After creating our voice database and encoding it in MATLAB, we tested some voice classification algorithms but we got a result that was not optimal. Then we tested on MATLAB the display of voice signals of the same message spoken by the same speaker under identical conditions, the display of the signals produces several different spectral shapes (see Fig.1).

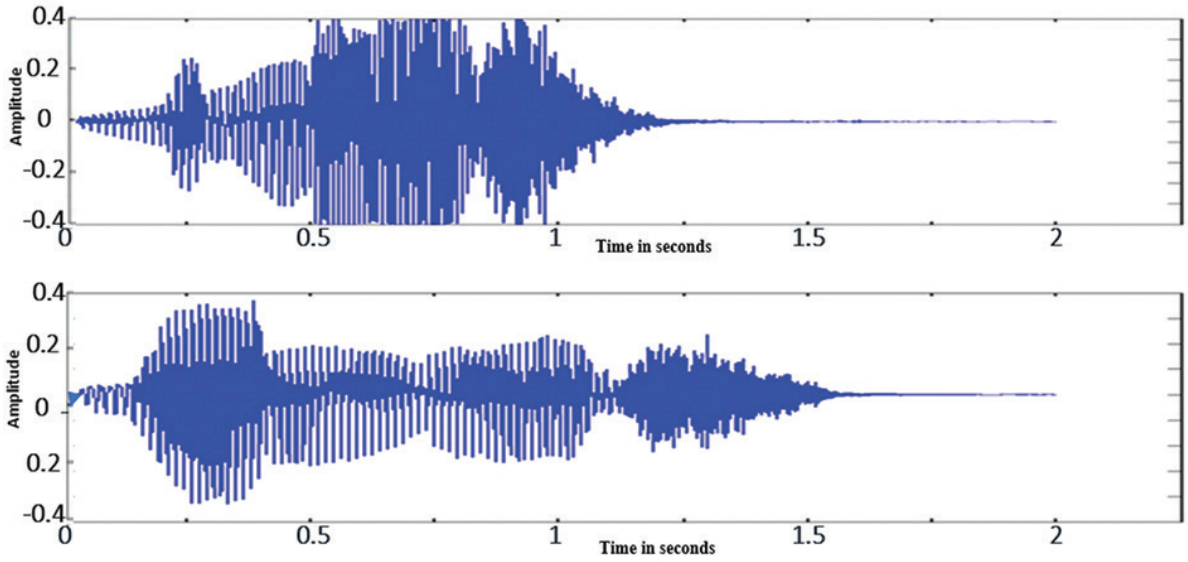


Fig.1. Name of Hamdoun pronounced by the same individual.

This variability is called intra-speaker [23]. The quality of the voice, the rate of speech, the degree of articulation are all factors at the basis of acoustic variations for a given signal. These variations lead to non-linear transformations over time of the speech signal. The non-linearity comes from the fact that the transformations affect the stable parts of the signal more than the phases of transitions [24]. To overcome this problem and that of computation time due to the size of the input vector we will use a preprocessing algorithm before using the recognition algorithms.

We move on to the step of storing the results (coded signal) in a \"text\" file to subsequently establish the previous steps of the extraction of speech characteristics by the MFCC algorithm [25]. Figure 2 shows the basic structure of audio signals and their extraction by the MFCC algorithm.

Individual Files N° 1 and **2** contain the audio files, their figures and their encodings. The contents of **Matrice_wav** folder are audio (matrix) files encoded by MATLAB the p' number of matrix elements is 16000

elements and the contents of **Vector_MFCC** folder are audio signal voiceprints. These are **p** input elements (p 1100 elements) of ANN and SVM.

After creating the fingerprints by MFCC, the vectors are divided into 2 parts one for training and another for testing. This method of extracting voice characteristics is an initial step and takes place before the achievement of voice recognition by the SVM and ANN algorithms. The main quality of the MFCC method is its biological plausibility since it uses a psycho-acoustic scale of frequencies similar to that of the inner ear [26-27].

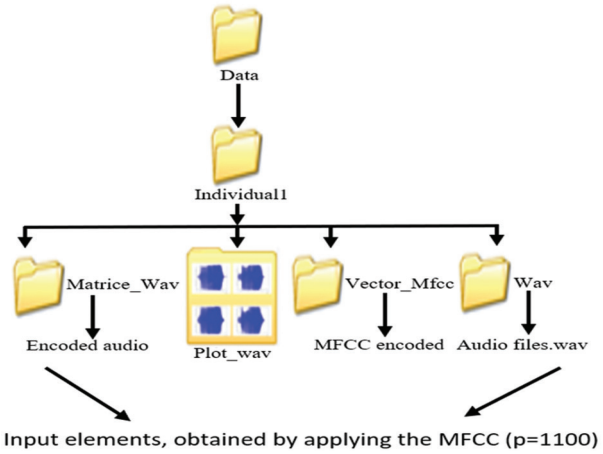


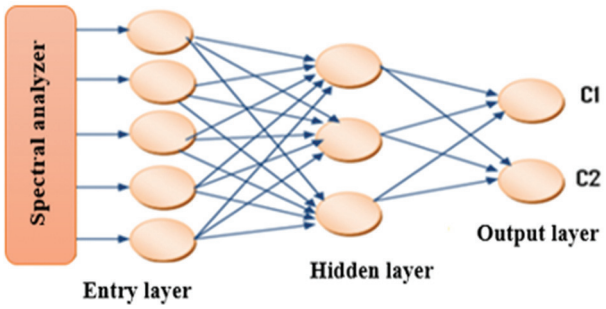
Fig. 2. Creation of a voice database

3.2. ARTIFICIAL NEURAL NETWORKS:

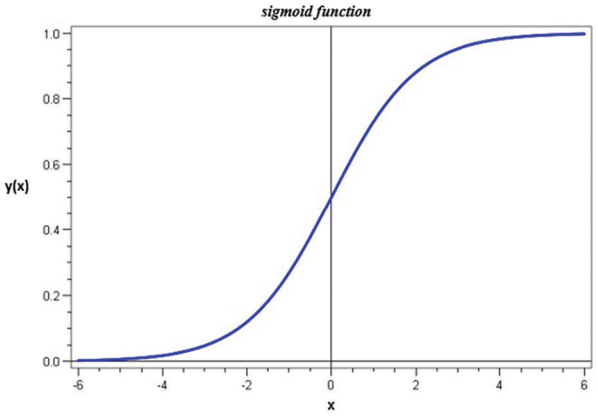
Automatic speech recognition is the process by which a computer issues an acoustic speech signal to text. It is carried out in this work by SVM and artificial neural networks (ANN) [28], in this part we have worked on the ANN. Artificial neural networks are mathematical and computational models [5] that mimic the functioning of the brain, including its ability to learn from examples and to generalize its knowledge [29]. There are different types of artificial neural network architectures, such as multilayer perceptron (MLP), recurrent networks, etc. [30]. These architectures require an example base to learn and an example base to test. The most widely used neural classifier is the MLP multilayer perceptron, which has also been widely analyzed and for which many learning algorithms have been developed [31].

A-Artificial neural networks MLP-type:

The creation of the MLP architecture depends on the parameters, such as the number of iterations, the number of hidden layers, the number of neurons in each layer, the training database, the test database and the learning rate. In the architecture of the multilayer Perceptron (See Fig.3 (a)), neurons in one layer are linked to all neurons in adjacent layers. The inputs of the functional unit of the first layer are calculated based on the inputs $\mathbf{X}_i$ and the weights of the links $\mathbf{W}_j$, the inputs of the functional unit of the second layer are the outputs of the first layer as well as their associated weights. The same principle holds for the output layer. The overall output of each MLP neuron is based on a sigmoid function which aims to keep the output in the interval [0,1] (See Fig.3 (b)).



a)



b)

Fig.3. a) Example of the MLP for isolated word recognition; b) Most significant interval of the sigmoid activation function.

Equations (2-4) represent the calculation of the outputs of each layer (example of two hidden layers). The synaptic weights are stored in the weight matrices denoted $\mathbf{W}_1$ (synaptic weights between the input vector $\mathbf{X}_i$ and the neurons of the 1st hidden layer) and $\mathbf{W}_2$ (synaptic weights between the 1st hidden layer and the outputs). The element (**i**, **j**) of a weight matrix represents the weight of the connection between neuron **j** of the downstream layer and neuron **i** of the upstream layer. The output of each neuron from each hidden layer is a sigmoid function. **F1** is the activation function associated with the first hidden layer, **F2** is the activation function associated with the second hidden layer (output layer in our architecture). In general, the activation functions are nonlinear and sigmoid in type (its main advantage is its derivability at all points).

The output layer:

$$S(z) = F2(ys(z)) = 1/(1 + e\text{^}(-y_s(z))) \quad (2)$$

Such as:

$$y_s(z) = \sum_{i=0}^{j} W_2(i,z) * F_1\,(y(i)) \quad (3)$$

So:

$$y(j) = \sum_{i=0}^{n} W_1(i,j) * X_i \quad (4)$$

Setting up a multilayer perceptron to solve a problem therefore involves determining the best weights applicable to each of the inter-neuronal connections. In this work, we have focused on automatic speech recognition. This determination is carried out through an MLP (multilayer perceptron) error back propagation algorithm.

B-Phases of MLP learning and testing

The multilayer perceptron architecture consists of an input layer which represents the object elements of the speech signal after extraction by the MFCC algorithm, of one or more hidden layers and an output layer representing the Classes (in our case the classes are the 15 names of the professors the department). After modifying these parameters and testing several MLP architectures, we obtained different results, but the best obtained results are presented in the following.

In this study, the number of neurons in the p 1100 input layer of the MLP corresponds to the number of characteristics of the input signal obtained by the application of the MFCC. The number of neurons in the output layer is fixed at 15 (the number of classes used to train the MLP).

To determine the number of hidden layers and the number of neurons to assign to each hidden layer, we have previously performed an experimental work. After the variation of the characteristics (e.g. change of neurons for a single hidden layer, increase of numbers of hidden layers, variation of iteration numbers and the database) we carried out the study with the aim of choosing the right values to obtain the right classification and recognition results. Figure 4 illustrates the learning steps that consist of calling a library (**hpp** folder) by the main program (src folder) as well as the MLP parameters (**arch_MLP** folder) and loading the database (step 3). We follow the same steps for the test but this time **step 3** is replaced by **step 4**.

Fig.4. Stages of learning and generalization

In our experiments, we first tested the MLP network with a single hidden layer. By varying the number of neurons in this layer, we found that the best results are obtained with a hidden layer of 7 neurons. In Fig.5 (A), we report the obtained results (global error) and express them in terms of learning and comprehension rate. The comprehension rate refers to the percentage of recognition of a word spoken by a speaker (in the testing phase) that does not already exist in the database

Then we worked out further experiments by adding another hidden layer. We fixed the number of neurons in the first hidden layer at 7 and in the second layer we varied the number of its neurons in order to find the most suitable architecture for our application. From the results shown in Fig.5 (B) we notice that the architecture using two hidden layers (7 in the first and 8 in the second) presents better results compared to other candidate architectures with two hidden layers. Despite the good results of this architecture, the MLP with a single hidden layer (containing 7 neurons) achieves better results compared to the MLP with two hidden layers.

For an architecture of three hidden layers, we followed the same steps as the second hidden layer. We set the number of neurons of the first hidden layer to 7, the number of neurons of the second to 8 and we varied the number of neurons of the third hidden layer, the results in Fig.5 (C) show that the MLP with a single hidden layer (containing 7 neurons) still offers the best performance.

In the architecture comprising a single hidden layer made up of 7 neurons, we varied the number of iterations depending on the overall error (see Fig.5 (D)). By analyzing this figure, we see that the increase in the number of iterations leads to a decrease in the overall error during the learning phases.

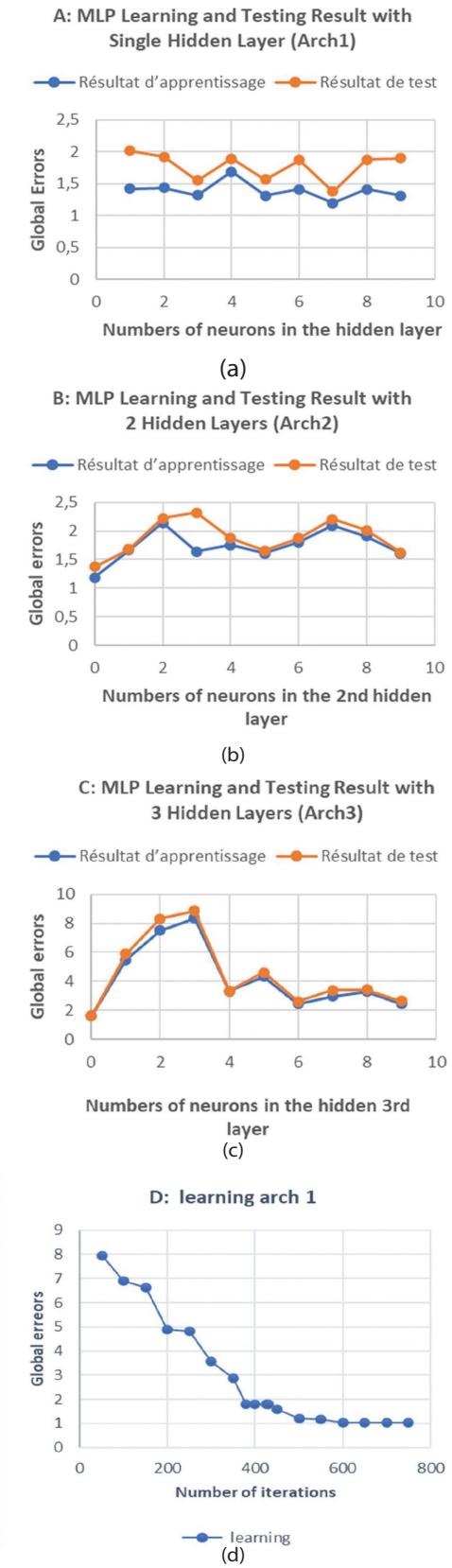


Fig. 5. a): Understanding rate and efficiency obtained with our MLP by varying the number of neurons using Single hidden layer (Arch1). **b)** Comprehension rate and efficiency obtained with MLPs with two hidden layers (Arch2). **c)** Comprehension rate and efficiency obtained with MLPs with three hidden layers (Arch3). **d)** Global error change depending on the number of iterations of Arch1.

Table 2 summarizes the best results of the 3 architectures discussed previously, with Tmax representing the maximum number of iterations. Testing of MLP with two and three hidden layers did not perform better results than those obtained using a single hidden layer. The architecture we have proposed for speech recognition has proven its effectiveness in our application and the result is quite significant. This confirmed the interest of the approach adopted in this study.

Table 2. Results of the neuronal classification of the three architectures 1, 2 and 3.

	Hidden layer 1	Hidden layer 2	Hidden layer 3	Number of iterations	Minimum Error %	Tmax
Arch 1	7 neurons	---	---	600	1.03851	3407
Arch 2	7 neurons	9 neurons	---	620	1.592432	3424
Arch 3	7 neurons	9 neurons	6 neurons	570	2.42865	3729

In this study, we have obtained several satisfactory experimental results that can be use in other similar applications. Among these values, we choose:

- One single hidden layer.
- 7 neurons in the hidden layer.

In comparison to the best result of our architecture after implementation with traditional digital implementations of artificial neural networks, our implementations simplify the complexity of the calculation and the economy of digital resources with a reduced footprint.

3.3. MULTI-CLASS SVM FOR AUTOMATIC RECOGNITION OF 15 NAMES.

SVMs (Support Vector Machines) are new techniques of supervised statistical learning that allow to create a decision surface between two classes defined in the same space (binary and statistical classifiers) [32]. One needs to provide a training dataset to build the classifier.

Several studies have been able to show the effectiveness of these techniques mainly in image processing [33]. The essential idea of SVM consists in projecting the input space's data (belonging to two different classes) non-linearly separable in a space of greater dimension called the characteristic space so that the data becomes linearly separable (See Fig.6). In this space, the optimal hyperplane construction technique is used to calculate the ranking function separating the two classes.

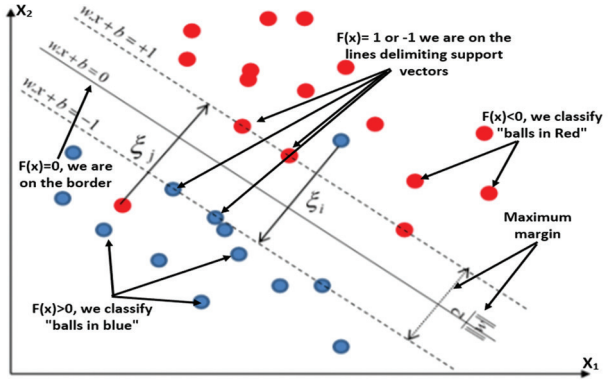


Fig. 6. Illustration space of characteristics of SVMs

The conventional SVM-based pattern recognition system can be schematized, as shown in Fig.7. SVMs require input vectors of fixed lengths that represent a whole word. They therefore propose a method to overcome this problem. After extracting the parameters of the speech signal by the MFCC method, these parameters are used as data input for the classification component (SVM), which will look for a separating hyperplane that separates the examples in the learning phase and makes a classification decision in the identification phase. The approach we have adopted is based on removing MFCC vectors at times when MFCCs change the least until there are only MFCC vectors remaining in the feature set, the algorithm in [32] represents the reduction method.

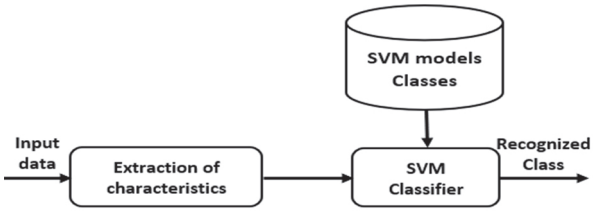


Fig. 7. Conventional SVM-based pattern recognition system [34].

Then, MFCC delete frames for which the difference between two successive frames is minimum. In the SVM module, there are two phases like the method used in the MLP-type ANN one for learning and the other for testing. In this part, we will use the two main methods of multi-class binary classification (Multi-class SVM) as a pattern recognition technique, one-against-one and one-against-all, in order to increase the reliability of the resulting system. One-against-one (1vs1), also called 'pairwise', is due to Kner et al. [35]. One-against-all (1 vs all) is the easiest and oldest method. According to the formulation of Vapnik [36], it consists in determining for each class k a hyperplane Hk (wk, bk) separating it from all the other classes. Figure.8 represents a case of separation of 3 classes.

In the next part we have combined classifiers of the same multi-class SVM type (SVM one against all and SVM one against one).

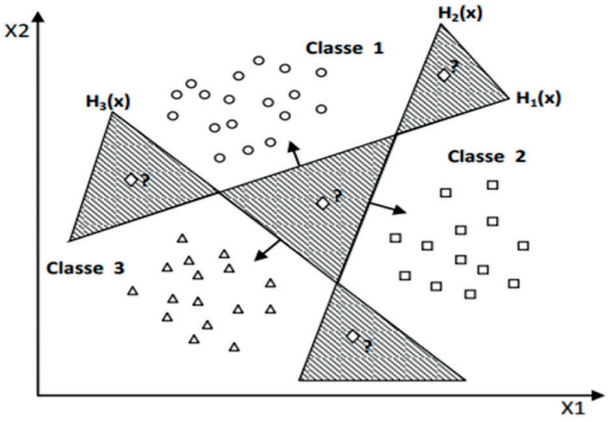


Fig. 8. One-against-rest approach with areas of indecision.

3.3.1. Summary of the SVM algorithm used: SVM learning and testing phases:

We used the same database which is used in ANN. This represents approximately 75 training points and 45 test points for each name (see table 1). All the speakers are adults (women and men). The signal is sampled at 8kHz, encoded on 32 bits. The learning and testing of SVMs were carried out using the LIBSVM software (Install and implement in MATLAB) which allows the classification of the algorithms that are described in [37], using MATLAB on a computer with a 2.8 GHz Intel i7-7700HQ processor with 16 GB of RAM. The LIBSVM library is developed with the aim of simplifying the use of SVMs as a tool. In this work we are using the libsvm-3.24 version in MATLAB. It was released on September 11th, 2019. The results of the training are the following parameters (Alpha, SVs, Bias b, nSV) necessary for the construction of the SVM classifier. In this work we combined classifiers of the same multi-class SVM type in parallel and then we tested their learning capacity. We used the same training base as well as combining the Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) in the same chain with the alignment algorithm. Tables 3 and 4 give the confusion matrices for the two classifiers.

3.3.2. SVM learning and testing phases:

A-chain based on one-on-one SVM with the fusion between ADL and ACP:

If we use the fusion of LDA and PCA, the retained values are (C=10 and γ=0.0028) which made it possible to obtain an almost zero error rate on the learning basis (test by the corpus of learning), and an error rate of 1,813% on the test basis. The PCA algorithm reduces the dimensionality of the space by eliminating the lowest eigenvalues, and the ADL aims to maximize interclass variations while minimizing intra-class variations. The confusion matrix for the next string: Pre-emphasis → MFCC → Reduction by PCA and LDA (Size vector adapted to SVM classifiers) → SVM one against one is shown in table 3:

Table 3. Confusion Matrix for SVM (One to One)

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15
C1	98.88	0.90	0	0	0.16	0	0	0	0	0	0	0.05	0	0	0
C2	0.04	98.35	0	0.05	0	0.28	0.2	0.01	0.22	0.1	0	0.6	0	0.15	0
C3	0	0	98.23	0	0	0.34	1.33	0	0	0	0	0	0.1	0	0
C4	1.2	0	0	97.89	0	0	0	0.2	0.45	0	0	0	0	0	0.25
C5	0.88	0	0	0.44	96.9	0	0	0.88	0	0	0	0.8	0.1	0	0
C6	0	0	0.33	0	0	99.56	0	0	0	0	0	0	0.11	0	0
C7	0.05	0	0	0	0	1.70	98.02	0.06	0	0	0	0	0	0	0.14
C8	1.33	0.32	0	0	0	0.12	0	97.83	0	0	0	0	0	0.40	0
C9	0	0	0	0	1.77	0	0.44	0	96.90	0	0	0	0.2	0.4	0.28
C10	0	0	0.46	0	0	0	0	0	0	97.79	0.42	0	0	0.44	0.88
C11	0	1.11	0	0.10	0	0	0	0	1.1	0	97.36	0	0	0	0.33
C12	0.88	0	0	0	0	0	0	0	0	0.88	0.88	96.46	0	0.88	0
C13	0.44	0	1.10	0	0	0	0	0	0.44	0	0.33	0	96.81	0	0.88
C14	0.21	0	0	0.10	1	0	0	0	0	0	0	0	0	98.48	0.21
C15	0.33	0	0.33	0.22	0	0	0	0	0.44	0	0	0	0	0.33	98.35

B- One-against-all SVM-based chain with the merger between LDA and PCA

If we use the one-against-all SVM classifier instead of the one-against-one SVM classifier with the serial combination of PCA and LDA in the recognition chain, we get an almost zero error rate on the learning basis (test by the learning corpus) and an error rate of 5.38% on the test basis. The recognition rate in this case shows that the classification using the one-on-one SVM algorithm is more efficient than the classification using the one-on-all SVM algorithm. The confusion matrix for the following chain: Pre-emphasis → MFCC → Reduction by PCA and LDA (Size vector adapted to SVM classifiers) → SVM one against all is shown in Table 4:

Table 4. Confusion matrix for SVM (one vs. all)

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15
C1	95.42	1.12	0.2	0	2.2	0	1.05	0	0	0	0	0	0	0	0
C2	0	96.02	0	0	0	0.44	1.33	0.88	0.00	0	0	1.33	0	0	0
C3	0	0	97.35	0	0	0.24	0.64	0	0	0	0	0	0.88	0	0.88
C4	0	0	0	96.46	0	0	0	0	0.88	0	0	0	2.21	0	0.44
C5	0.44	0	0	0.88	94.2	0	0	1.33	0	0	0	0.88	1.77	0.05	0.44
C6	0	0	2.21	0	0	93.36	0	0	0	0	0	3.10	1.33	0	0
C7	0.88	3.10	0	0	0	0	93.81	0.44	0	0	0	0	0.44	0	1.33
C8	0.44	0	2.10	1	0.22	0	0	93.81	0.22	0	1.33	0	0	0	0.88
C9	0.44	0	0	0	1.33	0	0	0	96.46	0	0	0	0.44	1.33	0
C10	0.88	1	0	0	0	0	0	1.8	0	92.01	0	0	0	0	2.10
C11	0	0	0.33	0.40	0	0	0	0	0.11	0	96.70	0.11	00.13	0.55	1.66
C12	0.3	1.1	0	1.6	0.7	0.41	0	0.2	1.1	0	0	93.36	0	0	1.03
C13	1.77	0.88	0	1.33	0	0.88	0	0.44	0	0.44	0	0	94.25	0	0
C14	3.62	0.44	0.44	1.54	0.88	0	0.44	0	0	0.44	0	0	0.77	90.97	0.44
C15	0.88	0	2.20	0.22	0	0	0	0.1	1.33	0	0	0	0.44	0.33	95.13

Depending on the table and the learning phase, the results show a lower average recognition rate than the previous chain but the classification rate obtained by this chain is also acceptable.

4. RESULTS AND DISCUSSIONS

Figure 9 shows the performance of the three architectures (Arch1, Arch2 and Arch3) of the MLP (this is a summary of Fig.5 (a), (b) and (c) and (d)) in terms of overall errors, number of hidden layers, number of neurons in each layer and number of iterations (see Table 2). The study is based on the application of voice recognition of the names of 15 people (15 classes) in our department.

For the multi-class classification, two approaches have been adopted: the first consists of a one-to-all classification, the second approach consists of a 1-to-1 classification. Figure 10 gives a comparison of the error

rates between the two SVM and MLP classifiers, during the learning and testing phase (We have based on Tables 2, 3 and 4 for the three classifiers MLP-ANN (a single hidden layer consisting of 7 neurons), SVM (one vs all) and SVM (one vs one)). By simply comparing our results (SVM one against all / SVM one against one) we notice that the recognition by a classifier using the one against one strategy with the merger between LDA and PCA is much more satisfactory compared to the classifier using the one against all strategy.

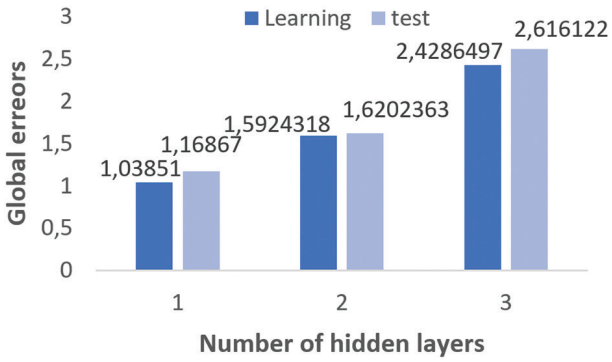


Fig. 9. Comparison between the three best architectures (Arch1, Arch2 and Arch3)

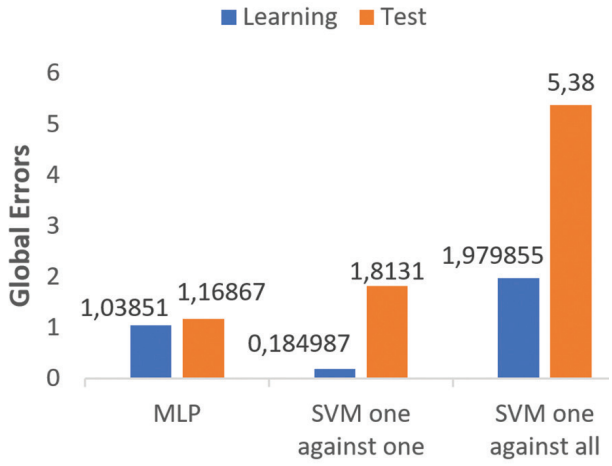


Fig. 10. Comparison between the best results of the three classifiers (Global error rate)

We notice that the classification rate is high even with few training samples, which shows its power of generalization. Since SVM is used to solve support vectors using quadratic programming, that will involve the computation of the order matrix. When the number of samples is large, storing and computing the array will consume a large amount of machine memory and runtime. We can say that there is not a great deal of difference between SVM one against one and the MLP-ANN. But the classification method based on MLP-ANN is the fastest and its size also leads to a allocation of reduced space in the hardware during implementation.

Among the objectives in our application, we cite the development and implementation of the techniques described previously in an FPGA card based on the embedded NIOSII processor. This technique is presented in the next section.

5. HARDWARE IMPLEMENTATION OF MFCC AND MLP-ANN ALGORITHMS:

In this work we have created a voice recognition system based on SVM and artificial neural networks of MLP type. In this part we will present the hardware architecture of the proposed MLP which is based on an input vector (voice signal after extraction by MFCC) and an output vector representing the 15 classes.

5.1. MFCC HARDWARE IMPLEMENTATION:

This part deals with the hardware implementation in the FPGA version DE2-70 board, produced by the company Altera [38] of the MFCC algorithm. According to Fig.11, the system receives a signal of an analog nature which comes from the microphone and sends it to the analog-to-digital converter ADC integrated in the codec circuit [39] to sample our signal at the frequency of 8Khz producing an output of digital samples. Each sample is coded on 32 signed bits. The output of the converter is linked with 2 registers (left register and right register) the size of each is 16 bits. The Hamming windowing and the discrete fourrier transform DFT are applied to the output of these registers, which will be responsible for calculating the DFT of this vocal piece. The windowing and DFT modules are included in the on-board NIOS II processor which reads the outputs of the DFT, and checks whether the vocal piece corresponds to silence or to a speech signal [40]. If it detects a speech signal, the NIOS II processor performs the various calculations [41], [42]. such as normalization, feature extraction, and fingerprint storage in SDRAM memory.

The block diagram below represents the entire system:

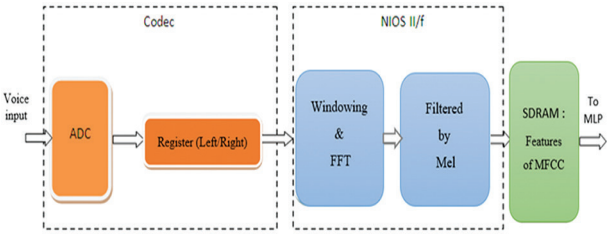
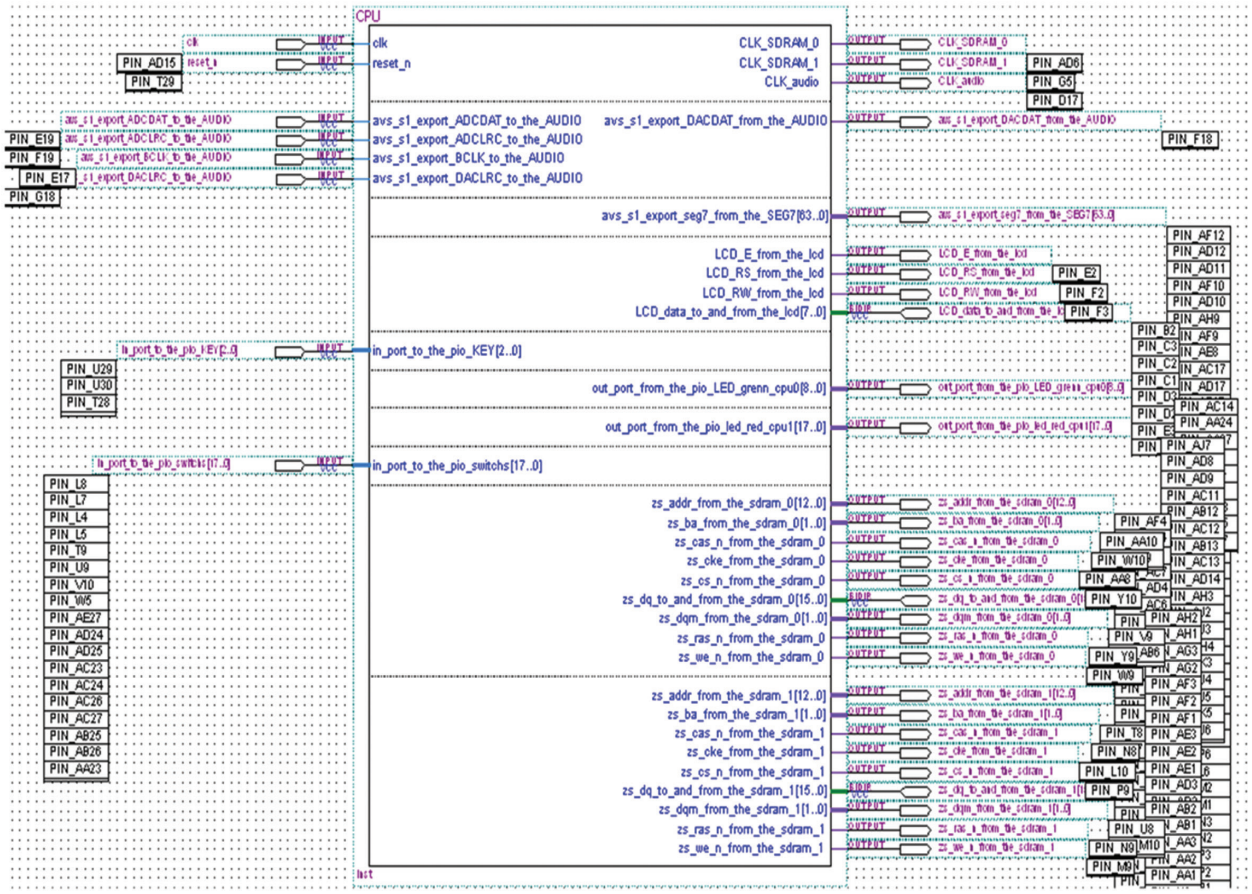


Fig.11. Hardware voice signal extraction

Figure 12 (a) shows the complete design of our NIOSII and Fig.12 (b) shows the compilation result of this Hardware processor by QUARTUS software [38], [43] and [44].

The implementation of the hardware part of the various peripherals constituting our system leaves enough space on the programmable FPGA component version EP2C70F896C6 for the addition of other peripherals or the hardware integration of speech processing algorithms.



a)

Flow Status	Successful - Sat Jul 13 20:00:55 2013
Quartus II Version	9.1 Build 222 10/21/2009 SJ Web Edition
Revision Name	mupross
Top-level Entity Name	mupross
Family	Cyclone II
Device	EP2C70F896C6
Timing Models	Final
Met timing requirements	No
Total logic elements	6,607 / 68,416 (10 %)
Total combinational functions	5,230 / 68,416 (8 %)
Dedicated logic registers	4,151 / 68,416 (6 %)
Total registers	4289
Total pins	249 / 622 (40 %)
Total virtual pins	0
Total memory bits	472,128 / 1,152,000 (41 %)
Embedded Multiplier 9-bit elements	4 / 300 (1 %)
Total PLLs	1 / 4 (25 %)

b)

Fig.12. (a): Hardware processor created in the FPGA circuit. (b): Hardware space reserved by our processor.

5.2. MLP HARDWARE IMPLEMENTATION:

In this study we obtained many satisfactory values, that we can use in applications similar to our application. Among these values, we choose:

- The number of neurons in the MLP input layer are p = 1100.
- A single hidden layer.
- 7 neurons in the hidden layer.
- The number of neurons in the output layer is fixed at c = 15.

As we can see in equations (2) and (3), the basic calculations of a single neuron are the multiplication of the outputs of the connected neurons by their associated weights, and the sum of these multiplied terms. Figure 13 describes the basic structure of the functional unit used for the serial hardware implementation [45] that performs these calculations. It includes a multiplier for multiplying the elements of the input vector with their corresponding weights. A sign extender is placed immediately after the multiplier. The input of the accumulative adder is connected to the output of the expansion unit.

The output of the accumulator is linked to the input of the adder. This functional unit has been implemented in the FPGA board.

For the implementation of MLP, we established the data encoding namely, inputs, outputs, weights, activation function, etc. therefore, it is necessary to limit the number of different variables:

- Two among MLP inputs and outputs of the activation functions of different neurons must have the same range to be able to easily manage multiple layers of processing. In this context, we have chosen 32-bit encoding.
- Weight of connections of MLP neurons (32-bit coding).

The primary storage strategy is to use SDRAM memory modules such that inputs, outputs, and connection

weights are stored in these modules. In what follows, we will describe in detail the software implementation of MLP (parallel). This type of implementation is given at two different levels of abstraction [45].

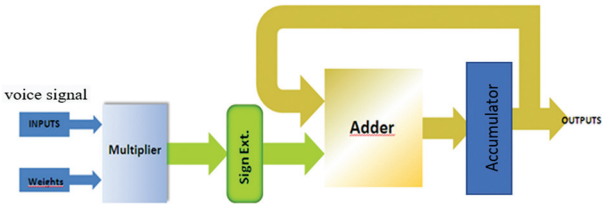


Fig.13. Hardware structure of the functional unit

To program the HW processor (see figure 12 (a)), figure 14 shows the structure of our software directory for the MFCC and MLP implementation (see HW processor Fig.11 and 13) based on the NIOSII multiprocessor.

The distribution of our multiprocessor software system for speech processing:

Software_khamlich file is containing our multiprocessor system NIOSII software This file is subdivided into the following subfolders.

- **Khamlich_CPU1** folder: main folder containing the folders and files of our program for audio signal recording, status display etc (same folder for a single processor).
- **Khamlich_CPU2** folder: main folder of the MFCC algorithm to extract the audio signal before recording.
- **Khamlich_CPU1_bsp** and **khamlich_CPU2_bsp** folders: the libraries of useful functions to define the process.

The connection between the hardware and software parts is done through the system.h file which contains the address of the registers of our IP block. The data transfer between the coprocessor and the memory is done through the Avalon bus. After the compilation of our processor HW (Fig.12 (a)) and SW (Fig.14) we created a file for the operating simulation (recording and playback) of the voice signal. The program of each processor must be located in its own memory area and to avoid the conflict between the program memory areas of each processor, it is better to put the program of each processor in a different memory.

Fig.14. Program structure of our NIOSII multiprocessor

6. IMPLEMENTATION RESULTS

Figure 15 illustrates the results of our processor implementation and speech processing algorithms. The results of this application after the implementation of the algorithms can be characterized by the following parameters number of ALUTs, number of onboard SDRAM memory blocks, maximum clock frequency, etc. To solve the real-time speech recognition problem, our application requires NIOSII / fast processors. After the implementations of these processors and the MFCC and MLP algorithms, the maximum recorded execution time obtained in our result, has undergone approximately 45% of reduction.

Family	Cyclone II
Device	EP2C70F896C6
Timing Models	Final
Met timing requirements	No
Total logic elements	6,339 / 68,416 (9 %)
Total combinational functions	5,261 / 68,416 (8 %)
Dedicated logic registers	3,782 / 68,416 (6 %)
Total registers	3920
Total pins	534 / 622 (86 %)
Total virtual pins	0
Total memory bits	147,392 / 1,152,000 (13 %)
Embedded Multiplier 9-bit elements	4 / 300 (1 %)
Total PLLs	1 / 4 (25 %)

Fig. 15. Size of our hardware architecture

So, the FPGA hardware / software design method described in this article offers many advantages over microcontroller-based solutions, including tailoring the hardware configuration to the needs of the application, protecting against obsolescence and for real-time processing, the integration of custom hardware accelerators. With the NIOSII processor and the Cyclone2 FPGA, embedded system designers can configure a processor-based system to perfectly suit their needs, which is impossible to achieve with frozen and peripheral-limited sets of a standard microcontroller. Our Nios II processor is credited with unprecedented flexibility for cost-sensitive, real-time processing, and security-critical systems [42].

7. CONCLUSION

In this work we used the multi-speaker mode and we examined the applications of wide margin separators (SVMs) and MLP for the classification of names of professors in ENSAK school. SVM algorithms with the one-on-one approach and MLP-ANN are the most suitable because of their efficiency. They outperform SVM (one against all) in both classification rate and execution time. After testing several classification architectures, we chose the MFCC coding method for the parameterization of the acoustic signal and the MLP-ANN method for the recognition of isolated words. The two multi-class classification strategies: one against one, and one against all, were used in order to choose the best classifiers. The MFCC is used for extracting voice data and to increase the reliability of the resulting system.

Following several changes in the internal FPGA circuit architecture, we have chosen the best hardware processors that are suitable for our speech processing application. The cores of these NIOSII programmable processors and the elements for mapping the MFCC and MLP architectures are implemented in parallel in the FPGA circuit. These processors were used for the purpose of testing our application and choosing the best one. We have created two on-board processors; the first processor is used by MLP-ANN for classification and the second is used for future voice signal extraction by MFCC and, their implementation of MLP. We used only the adjusted synaptic weights, numbers of hidden layers, numbers of neurons in each layer and numbers of classes. This gives us an advantage of MLP. We don't need a memory to store a base of the signals for comparison with the voice input signal. So, our implementation makes it possible to simplify the complexity of the calculation and to save digital resources, figure 12 (b) illustrates the size reserved in the FPGA circuit by our NIOSII.

8. REFERENCES

[1] R. Gemello, F.Mana, D. Albesano, R. Mori, "Multiple resolution analysis for robust automatic speech recognition", Computer Speech and language, Vol. 20, No. 1, 2006, pp. 2–21.

[2] K. H. Davis, R. Biddulph, S. Balashek, The Journal of the Acoustical Society of America, Vol. 24, No. 6, 1952, p. 637.

[3] C. Heelan, A. Nurmikko, W. Truccolo, "FPGA implementation of deep-learning recurrent neural networks with sub-millisecond real-time latency for BCI-decoding of large-scale neural sensors (104 nodes)", Proceedings of the 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Honolulu, HI, 18-21 July 2018, pp. 1070-1073.

[4] X. Zhu, Y. Chen, "Improved FPGA implementation of Probabilistic Neural Network for neural decoding", Proceedings of the International Conference on Apperceiving Computing and Intelligence Analysis Proceeding, Chengdu, China, 17-19 December 2010, pp. 198-202.

[5] S. Maya, R. Reynoso, C. Torres, M. A. Estrada, "Compact Spiking Neural Network Implementation in FPGA", Proceedings of the 10th International Workshop on Field Programmable Logic and Applications, Austria, 27-30 August, 2000, pp 270-276.

[6] A. Graves, N. Jaitly, "Towards End-to-End Speech Recognition with Recurrent Neural Networks", Proceedings of the 31st International Conference on Machine Learning, Beijing, China, June 2014.

[7] J. L. Beuchat, J. O. Haenni, E. Sanchez, "Hardware Reconfigurable Neural Networks, Parallel and Distributed Processing", Lecture Notes in Computer Science, Vol 1388, Springer, Berlin, Heidelberg, 1998, pp 91-98.

[8] K. Compton, S. Hauck, "An Introduction to Reconfigurable Computing", IEEE Computer, April 2000.

[9] D. Verstraeten, B. Schrauwen, D. Stroobandt, "Reservoir Computing with Stochastic Bit stream Neurons", Proceedings of the 16th annual Prorisc workshop, November 2005, pp. 454–459.

[10] S. J. Melnikoff, S. F. Quigley, M. J. Russell, "Implementing a simple continuous speech recognition system on an FPGA", Proceedings. 10th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, Napa, CA, USA, 24-24 April 2002, pp. 275-276.

[11] Y. Maeda, M. Wakamura, "Simultaneous perturbation learning rule for recurrent neural networks and its FPGA implementation", IEEE Transactions on Neural Networks, Vol. 16, No. 6, 2005, pp. 1664-1672.

[12] J. L. Rosselló, V. Canals, A. Morro, "Hardware implementation of stochastic-based Neural Networks", Proceedings of the International Joint Conference on Neural Networks, Barcelona, Spain, 18-23 July 2010, pp. 1-4.

[13] H. Demuth, M. Beale, "Neural Network Toolbox for use with MATLAB", The Mathworks Inc., 1993.

[14] M. Marouf, J. Popovic-Bozovic, I. Popovic, "FPGA implementation of neural network as processing element in ice detector", Proceedings of the 11th Symposium on Neural Network Applications in Electrical Engineering, Belgrade, Serbia, 20-22 September 2012, pp. 81-84.

[15] F. M. Shakiba, M. Zhou, "Novel Analog Implementation of a Hyperbolic Tangent Neuron in Artificial Neural Networks", IEEE Transactions on Industrial Electronics, 2020. (in press)

[16] M. R. Karim, O. Beyan, A. Zappa, I. G. Costa, D. R.-Schuhmann, M. Cochez, S. Decker, "Deep learning-based clustering approaches for bioinformatics", Briefings in Bioinformatics, Vol. 22, No. 1, 2021, Pages 393-415.

[17] K. Sohn, D. Berthelot, C. L. Li, Z. Zhang, N. Carlini, E. D Cubuk, A. Kurakin, H. Zhang, C. Raffel. Fixmatch, "Simplifying semi-supervised learning with consistency and confidence", arXiv:2001.07685.

[18] W. N. Hsu, J. Glass. "Extracting domain invariant features by unsupervised learning for robust automatic speech recognition", Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Calgary, AB, Canada, 15-20 April 2018, pp. 5614-5618.

[19] S. Ling, Y. Liu, J. Salazar, K. Kirchhoff. "Deep contextualized acoustic representations for semi-supervised speech recognition", Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Barcelona, Spain, 4-8 May 2020, pp. 6429-6433.

[20] M. Hamza, T. Khodadadi, Sellappan Palaniappan "A novel automatic voice recognition system based on text-independent in a noisy environment" International Journal of Electrical and Computer Engineering, Vol. 10, No. 4, 020, pp. 3643-3650.

[21] Y. Liu et al., "Classifying respiratory sounds using electronic stethoscope", Proceedings of the IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation, San Francisco, CA, USA, 4-8 August 2017, pp. 1-8.

[22] W. Han, C. F. Chan, C. S. Choy, K. P. Pun, "An efficient MFCC extraction method in speech recognition", Proceedings of the IEEE International Symposium on Circuits and Systems, Kos, Greece, 21-24 May 2006.

[23] J. H. L.Hansen, H. Bořil, "On the issues of intra-speaker variability and realism in speech, speaker, and language recognition tasks", Speech Communication, Vol. 101, 2018, pp. 94-108.

[24] E. B. Holmberg, R. E. Hillman, J. S. Perkell, C. Gress, "Relationships Between Intra-Speaker Variation in Aerodynamic Measures of Voice Production and Variation in SPL Across Repeated Recordings", Journal of Speech, Language, and Hearing Research, Vol. 37, No. 3, 1994, pp. 484-495.

[25] D. Ellis, "Reproducing the feature outputs of common programs using Matlab and melfcc.m", (Online: http://labrosa.ee.columbia.edu/matlab/rastamat/mfccs.htmla)

[26] S. B. Davis, P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences",. IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. 28, No 4, 1980, pp 357-366.

[27] B. A. Mellor, A. P. Varga. “Noise masking in a transform domain”, Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 2, Minneapolis, MN, USA, 27-30 April 1993, pp. 87-90.

[28] R. P. Lippmann, “Review of neural networks for speech recognition”, Neural Computing, Vol. 1, o. 1, 1989, pp. 1-38.

[29] S. Ahmad, “A Study of Scaling and Generalization in Neural Networks”, Technical Report CCSR-88-13, University of Illinois, Center for Complex Systems Research, 1988.

[30] M. W. Gardner, S. R. Dorling, “Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences” Atmospheric Environment, Vol. 32, No. 14-15, 1998, pp. 2627-2636.

[31] B. Widrow, M. Lehr, “30 years of adaptive neural networks: Perceptron, Madaline and Backpropagation”, Proceedings of the IEEE, Vol. 78, No. 9, 1990, pp. 1415-1442.

[32] Z. Hao, L. Shaohong, S. Jinping, "Unit Model of Binary SVM with DS Output and its Application in Multi-class SVM", Proceedings of the 4th International Symposium on Computational Intelligence and Design, Hangzhou, China, 28-30 October 2011, pp. 101-104.

[33] J. Kharroubi, "Etude de Techniques de Classement_ Machines à Vecteurs Supports pour la Vérification Automatique du Locuteur", Thèse de doctorat. Ecole Nationale Supérieure des Télécommunications. Paris, 2002.

[34] X. Wang, K. P. Kuldip, "Feature extraction and dimensionality reduction algorithms and their applications in vowel recognition", Journal. School of Microelectronical Engineering, Nathan Campus, Griffith University, Brisbane, Qld 4111, Australia. 2002.

[35] S. Knerr, L. Personnaz, G. Dreyfus, “Single-layer learning revisited: a stepwise procedure for building and training a neural network”, Soulié F.F., Hérault J. (Editors) Neurocomputing, NATO ASI Series, Vol 68. Springer, 1990.

[36] B. Liu, Z. Hao, X. Yang, "Modifying the Decision Function in One-Against-All Algorithm for Multi-Classification", Proceedings of the International Conference on Machine Learning and Cybernetics, Dalian, China, 13-16 August 2006, pp. 3389-3394.

[37] C.-C. Chang, C. J. Lin. “LIBSVM: a library for support vector machines”, ACM Transactions on Intelligent Systems and Technology, Vo. 2, No. 3, 2016, pp.1-27.

[38] A. Silitonga, M. Hutabarat, "An emulation of transparent interface design based on TCP/IP implemented onto FPGA of an Altera Nios® Board", Proceedings of the 9th International Conference on Telecommunication Systems Services and Applications, Bandung, Indonesia, 25-26 November 2015, pp. 1-6.

[39] S. Khamlich, F. Khamlich, I. Atouf, M. Benrabh “Parallel implementation of NIOS II multiprocessors, Cepstral coefficients of Mel frequency and MLP architecture in FPGA: apply in speech processing”, WSEAS Transactions on signal processing, Vol. 16, 2020.

[40] S. Khamlich, M. El Jourmi, A. Ailane, “ANN, MFCC and its applications in speaker recognition”, International Journal of Artificial Intelligence and Neural Networks, Vol. 7, No. 1, 2017.

[41] M. B. Christopher, “Pattern Recognition And Machine Learning”, Springer, 2006.

[42] D. González, G. Botella, U. Meyer-Baese, C. García, C. Sanz, M. Prieto-Matías, F. Tirado, “Low A: Cost Matching Motion Estimation Sensor Based on the NIOS II Microprocessor”, Sensors, Vol. 12, 2012, pp. 13126-13149.

[43] P.P. Chu, “Embedded SoPC Design with Nios II Processor and VHDL Examples”, John Wiley & Sons, 2011.

[44] Quartus II Software information and download. (Online: http://www.altera.com/products/software/quartus-ii)

[45] S. Khamlich, A..Hamdoun, I. Atouf, M. Madiafi, “Serial Hardware Implementation of the MFCC and MLP Architecture on FPGA Circuit”, International Journal of Engineering and Technology, Vol. 5, No. 4, 2013, pp. 3520-3526.