6

A Speech Recognition System for Embedded Applications Using the SOM and TS-SOM Networks

Amauri H. Souza Júnior[1], Guilherme A. Barreto[1] and Antonio T. Varela[2]
[1]*Federal University of Ceará, Department of Teleinformatics Engineering*
[2]*Federal Institute of Education, Science and Technology of Ceará*
[1,2]*Brazil*

1. Introduction

The self-organizing map (SOM) (Kohonen, 1982) is one of the most important neural network architecture. Since its invention it has been applied to so many areas of Science and Engineering that it is virtually impossible to list all the applications available to date (van Hulle, 2010; Yin, 2008). In most of these applications, such as image compression (Amerijckx et al., 1998), time series prediction (Guillén et al., 2010; Lendasse et al., 2002), control systems (Cho et al., 2006; Barreto & Araújo, 2004), novelty detection (Frota et al., 2007), speech recognition and modeling (Gas et al., 2005), robotics (Barreto et al., 2003) and bioinformatics (Martin et al., 2008), the SOM is designed to be used by systems whose computational resources (e.g. memory space and CPU speed) are fully available. However, in applications where such resources are limited (e.g. embedded software systems, such as mobile phones), the SOM is rarely used, especially due to the cost of the best-matching unit (BMU) search (Sagheer et al., 2006).
Essentially, the process of developing automatic speech recognition (ASR) systems is a challenging tasks due to many factors, such as variability of speaker accents, level of background noise, and large quantity of phonemes or words to deal with, voice coding and parameterization, among others. Concerning the development of ASR applications to mobile phones, to all the aforementioned problems, others are added, such as battery consumption requirements and low microphone quality.
Despite those difficulties, with the significant growth of the information processing capacity of mobile phones, they are being used to perform tasks previously carried out only on personal computers. However, the standard user interface still limits their usability, since conventional keyboards are becoming smaller and smaller. A natural way to handle this new demand of embedded applications is through speech/voice commands. Since the neural phonetic typewriter (Kohonen, 1988), the SOM has been used in a standalone fashion for speech coding and recognition (see Kohonen, 2001, pp. 360-362). Hybrid architectures, such as SOM with MultiLayer Perceptrons (SOM-MLP) and SOM with Hidden Markov Models (SOM-HMM), have also been proposed (Gas et al., 2005; Somervuo, 2000). More specifically, studies involving speech recognition in mobile devices systems include those by Olsen et al. (2008); Alhonen et al. (2007) and Varga & Kiss (2008).
It is worth noticing that Portuguese is the eighth, perhaps, the seventh most spoken language worldwide and the third among the Western countries, after English and Spanish. Despite

that, few automatic speech recognition (ASR) systems, specially commercially available ones, have been developed and it is available worldwide for the Portuguese language. This scenario is particularly true for the Brazilian variant of the Portuguese language, due its large amount of accent variation within the country. Scanzio et al. (2010), for example, report experiments with a neural network based speech recognition system and include tests with the Brazilian Portuguese language. Their work is focused on a hardware-oriented implementation of the MLP network.

In this context, the current paper addresses the application of self-organizing maps to the Brazilian Portuguese isolated spoken word recognition in embedded systems. For this purpose, we are particularly interested in evaluating several software strategies to speedup SOM computations in order to foster its use in real-time applications. The end-user application is a speaker-independent voice-driven software calculator which is embedded in the Nokia N95 smartphone.

The remainder of the paper is organized as follows. In Section 2 the SOM architecture and its learning process are described. In Section 3, the techniques used for reducing the SOM's computational cost are introduced. Section 4 presents the evaluation of SOM-based methods and other classical algorithms for speech recognition. The paper is concluded in Section 5.

2. Evaluated architectures

2.1 The basic SOM algorithm

The *Self-Organizing Map* (SOM) (Kohonen, 1982) is a well-known unsupervised competitive learning algorithm that learns, from examples, a projection from a high-dimensional continuous input space $\mathcal{X}$ onto a low-dimensional discrete space (lattice) $\mathcal{A}$ of M neurons which are arranged in fixed topological forms, e.g., as a 2-dimensional rectangular array. The map $i^*(\mathbf{x}) : \mathcal{X} \rightarrow \mathcal{A}$, defined by the weight matrix $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_q), \mathbf{w}_i \in \mathcal{X}$, assigns to each input vector $\mathbf{x} \in \mathcal{X}$ a winning neuron $i^* \in \mathcal{A}$. Using Euclidean distance, one of the simplest strategies to find the winning neuron, $i^*(t)$, is given by

$$i^*(t) = \arg\min_{\forall i} \|\mathbf{x}(t) - \mathbf{w}_i(t)\|_2, \tag{1}$$

where $\mathbf{x}(t) \in \mathbb{R}^p$ denotes the current input vector, $\mathbf{w}_i(t) \in \mathbb{R}^p$ is the weight vector of neuron i, and t symbolizes the time steps associated with the iterations of the algorithm.

Adjustment of the weight vectors of the winning neuron (also called best matching unit - BMU) and of those neurons belonging to its neighborhood:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \alpha(t) h(i^*, i; t)[\mathbf{x}(t) - \mathbf{w}_i(t)] \tag{2}$$

where $0 < \alpha(t) < 1$ is the learning rate and $h(i^*, i; t)$ is usually the Gaussian neighborhood function:

$$h(i^*, i; t) = \exp\left(- \frac{\|\mathbf{r}_i(t) - \mathbf{r}_{i^*}(t)\|^2}{2\sigma^2(t)} \right) \tag{3}$$

where $\mathbf{r}_i(t)$ and $\mathbf{r}_{i^*}(t)$ are the coordinates of neurons i and i^* in a predefined output array respectively, and $\sigma(t) > 0$ defines the radius of the neighborhood function at time t.

The variables $\alpha(t)$ and $\sigma(t)$ should both decay with time to guarantee convergence of the weight vectors to stable steady states. The operations defined in Eqs. (1) and (2) are repeated for a certain numbers of iterations until a steady state of global ordering of the weight vectors has been achieved.

	BMU search	updating phase	param. adaptation	total
multi.	$NnM\epsilon$	$NM\epsilon(4+k+n)$	2ϵ	$\epsilon[2+NM(2n+k+4)]$
divi.	-	$NM\epsilon$	ϵ	$\epsilon(NM+1)$
adi.	$NM\epsilon(n-1)$	$NM\epsilon(k-1+n)$	-	$NM\epsilon(2n+k-2)$
sub.	$NnM\epsilon$	$NM\epsilon(k+n)$	-	$NM\epsilon(2n+k)$
comp.	$(M-1)N\epsilon$	-	-	$N\epsilon(M-1)$
exp.	-	$NM\epsilon$	2ϵ	$\epsilon(2+NM)$
total	$N\epsilon(3Mn-1)$	$MN\epsilon(3n+3k+5)$	5ϵ	

Table 1. Number of operations in the basic SOM algorithm.

As mentioned in the introduction, the SOM has been widely used in engineering and data analysis but rarely used in real-time applications. The main reason is the computational cost of the SOM, especially with respect to the search for the BMU and the weight updating phase. Table 1 reports the number of atomic operations (multiplication, division, subtraction, comparison and exponentiation) in the basic training SOM algorithm with a Gaussian neighborhood function. In this table, N denotes the number of input vectors, M is the number of neurons, n is the dimension of input vectors, k represents the dimension of the map, and ϵ is the number of iterations.
If the exhaustive BMU search is performed, the algorithm requires the computation of the distances between the input vector and the weight vector for all neurons in the map (see Eq. (1)). The updating phase is the most computationally demanding phase when a Gaussian function is used (for being an asymptotic function, all neurons have their weights updated). Thus, the SOM algorithm has linear complexity, $O(M)$, in numbers of neurons.

2.2 Tree structured SOM

The tree structured self-organizing map (TS-SOM) was proposed by Koikkalainen & Oja (1990) as a fast alternative to the SOM training/testing procedures. The tree search reduces the time complexity of the BMU search from $O(M)$ to $O(\log M)$. The TS-SOM is comprised of several layers of SOM networks with different resolutions, where the neurons in the same layer are laterally connected. In addition, there are hierarchical connections among neurons of different layers. Usually, each neuron is related to 2^D neurons in the next layer, where D is the dimension of the grid. In the training procedure the neurons are adapted layer by layer.
The TS-SOM feature that differs the most from the basic SOM algorithm is the BMU search. In the conventional tree search, the search starts from the root node and proceeds to look for the most similar node to the input pattern in the next layer. In the TS-SOM search, the winner of the next layer is selected from a larger set of nodes: the BMU is the best matching child node among the parent nodes in the immediate neighborhood. The inclusion of the children nodes of the nearest neighbors is computationally light because it does not depend on the number of neurons in the layer. The updating step of the TS-SOM algorithm is similar to the basic algorithm: only the winning neuron and its immediate neighborhood are updated.

3. Reducing the computational costs of the SOM

Recently, several hardware-oriented fast implementations of neural network algorithms, including the SOM, using graphics processing units (GPU) have been proposed (Xiao et al., 2010; Scanzio et al., 2010; Oh & Jung, 2004), but we are particularly interested in speeding up the SOM computations by simple and efficient software methods, to be described in the next sections.

Use Squared Euclidean Distances - By definition, textbook SOM algorithm requires the execution of the square root function in order to *correctly* compute euclidean distances. However, since we are not interested in the absolute values of the distances, but rather in their relative magnitudes, the square root function does not need to be executed. In this case, the squared euclidean distance can be computed, for example, through a dot product:

$$\|\mathbf{x}(t) - \mathbf{w}_i(t)\|^2 = (\mathbf{x}(t) - \mathbf{w}_i(t))^T(\mathbf{x}(t) - \mathbf{w}_i(t)). \tag{4}$$

Partial Distance Search - The Partial Distance Search (PDS) (Bei & Gray, 1985) is a heuristic method that involves a simple modification in the way distances are calculated. A pseudocode for the PDS technique is given below. During the computation of the accumulated distance sum, d, if the squared partial distance exceeds the smallest distance, d_{min}, to the nearest neighbor found so far, the computation is stopped. The additional time required by the evaluation of the stopping rule $(IF\ (d > d_{min}),\ THEN...$ is, on average, shorter than the time used in exhaustive search (Niskanen et al., 2002). It is reported that PDS reduces the computational cost of the BMU search by half or more (Bei & Gray, 1985).

Algorithm 3.1: PDS($\mathbf{x}$)

```
$i^* \leftarrow 1$
$d_{min} \leftarrow$ SQUAREDEUCLIDEANDISTANCE($\mathbf{x}, \mathbf{w}_1$)
for $i \leftarrow 2$ to $M$
do
    $discard \leftarrow$ false
    $d \leftarrow 0$
    for $j \leftarrow 1$ to $n$
    do
        $aux \leftarrow \mathbf{x}(j) - \mathbf{w}_i(j)$
        $d \leftarrow d + aux * aux$
        if $(d > d_{min})$
        then
            $discard \leftarrow$ true
            exit
    if $(discard =$ false $)$
    then
        $i^* \leftarrow i$
        $d_{min} \leftarrow d$
return $(i^*)$
```

Shortcut Winner Search - The Shortcut Winner Search (SWS) was proposed by Kohonen (1997). This method requires a partially ordered map. Thus, the probability of the winning neuron for an input vector $\mathbf{x}$ being in the neighborhood of last winning neuron is high. Thereby, Kohonen (1997) recommends storing a pointer relating a vector $\mathbf{x}$ to the winner neuron $i^*(t)$, in the last iteration t. Then, in the iteration $t+1$ the BMU search can be performed in the immediate neighborhood of the neuron $i^*(t)$, and only if a neuron most similar to $\mathbf{x}$ is found, the search continues in the new neighborhood and so on, until the winner is in the search center.

/um/ (1)	/dois/ (2)	/tres/ (3)
/quatro/ (4)	/cinco/ (5)	/seis/ (6)
/sete/ (7)	/oito/ (8)	/nove/ (9)
/zero/ (0)	/mais/ (plus)	/menos/ (minus)
/vezes/ (times)	/dividido/ (divided by)	/limpar/ (clear)
/voltar/ (back)	/resultado/ (result)	–

Table 2. Words recorded (in Brazilian Portuguese).

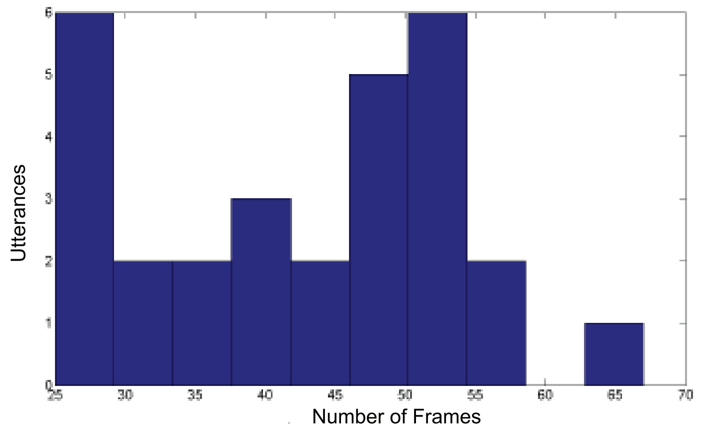
Neighborhood Functions - Rather than using the Gaussian neighborhood function, one can use the rectangular function since it does not require the computationally expensive exponential function. Let $R(t)$ be the winner's neighborhood radius in iteration t and $N_{i^*}(t)$ the set of neurons such that $\| \mathbf{r}_{i^*}(t) - \mathbf{r}_i(t) \|^2 < R^2(t)$. Then $h(i^*,i;t) = 1$ is valid for all neurons $i \in N_{i^*}(t)$ and $h(i^*,i;t) = 0$ otherwise. Another neighborhood function is the truncated Gaussian. In this case, values of the Gaussian functions are calculated only for the neurons within a distance range from the winner. The computational advantage is that a smaller number of neurons have their weights updated in comparison to the standard Gaussian neighborhood function.

4. Simulations and discussion

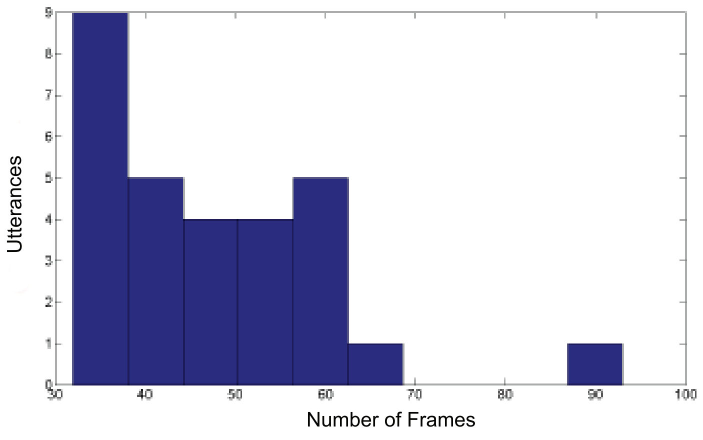
To evaluate the SOM-based algorithms, we recorded Brazilian Portuguese spoken words consisting of the basic mathematic operations and digits needed for the development of an embedded voice-driven software calculator. The data set includes speech data from 14 speakers. For each speaker, 51 utterances were acquired (17 different words repeated 3 times). Table 2 shows the selected words. In this table, the meaning of each word in English is shown within parentheses.
All the words were recorded in an indoor environment corrupted by air-conditioning noise. The involved speakers (11 men and 3 women) spoke freely, maintaining their respective accents. This fact contributed to a more difficult recognition tasks because even the same utterances had different durations after the endpoint detection. For example, Figure 1 shows the histogram of frame numbers of two words: /dividido/ in Figure 1(a) and /resultado/ in Figure 1(b).
In the front-end processing we use a 8 KHz sampling rate and 8 bits for the quantization of the signal amplitudes. Furthermore, we detected the endpoints by speech energy using the algorithm described in Wang et al. (2008).
In the feature extraction step, each utterance is segmented into frames of 20 milliseconds. For each frame, a short-term Linear Predictive Coding (LPC) analysis is performed and then 10 cepstral coefficients are obtained from the LPC coefficients (Deller et al., 2000) by means of the following expression:

$$\widehat{c}_h(k) = \begin{cases} \log G, & k = 0, \\ \widehat{a}(k) + \sum_{i=1}^{P-1} (i/k)\widehat{c}_h(i)\widehat{a}(k-i), & k = 1, ..., P, \end{cases} \tag{5}$$

where $\widehat{a}(i)$ are the LPC coefficients, $\widehat{c}_h(i)$ are the cepstral coefficients, G is a gain factor (associated with the AR model) estimated as $G = r_s(0) - \sum_{i=1}^{P} \widehat{a}(i) r_s(i)$ (Rabiner & Schafer, 1978) and $r_s(i)$ is the autocorrelation value at lag i, $i = 0, 1, ..., P$.



(a) Word /dividido/ (divided by).



(b) Word /resultado/ (result).

Fig. 1. Histograms of frame numbers.

We compared the SOM-based neural networks with supervised methods commonly used for speech recognition, such as the MLP network, the sequential K-means algorithm and the *Dynamic Time Warping* (DTW) technique (Sakoe & Chiba, 1978).
The approach used in the simulations of the SOM variants and the K-means algorithm is that described in Rabiner & Juang (1993), where each word category (e.g. /um/) is associated with a vector quantizer that maps all examples from this word category (see Figure 2). When a new utterance is presented, all the vector quantizers are evaluated and the one with the lowest quantization error, $E_q^{(i)}$, $i = 1, ..., U = 17$, is the winner.
The SOM and TS-SOM networks used in the simulations had 10 input units and 256 neurons arranged in a 16×16 array. For the sake of comparison, the K-means algorithm also used $K = 256$ prototype vectors. It is worth highlighting that each VQ in the multiple vector quantizer approach described in Figure 2, be it a SOM-based one or the K-means algorithm, use 256 prototype vectors. Hence, the total number of prototypes used is $17 \times 256 = 4352$, demanding considerable computational efforts if the classifier is to be embedded in a mobile phone.
For the MLP network, we used 10 input units, 50 hidden neurons and 17 output neurons. For its training, we used the 1-out-of-N encoding for the target outputs, a learning rate of 0.01 and logistic activation functions for the hidden/output neurons. All the aforementioned

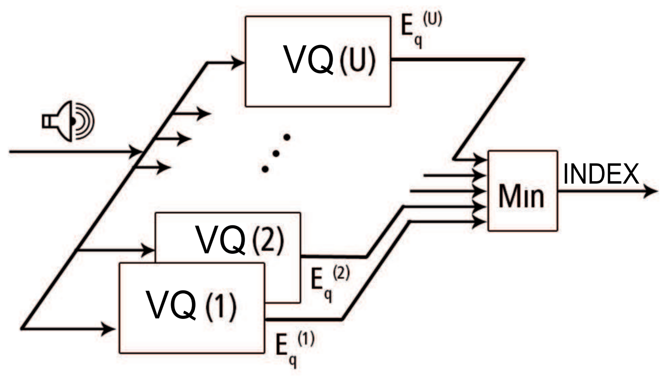


Fig. 2. Multiple vector quantizer approach for speech recognition as described in Rabiner & Juang (1993).

neural networks and the *K*-means algorithm were trained for 500 epochs and evaluated by the hold-out method for 20 independent training/testing runs. For training, it is used 80% of the total number of data vectors, randomly selected at each run, while the remaining 20% data vectors are used for testing purposes.
Table 3 reports the statistics of the performances of all evaluated algorithm, averaged over 20 independent training/testing runs. These simulations correspond to offline tests, executed on a personal computer, not in an embedded device. They were necessary in order to help in the choice of the classification algorithm to be embedded in the mobile phone. The offline tests were performed in a Dell personal computer, processor's clock of 2.33 GHz, 1 GB of RAM memory, in a Windows XP system and Java programming language. In addition, we used a standard (i.e. non-professional) desktop microphone to capture the utterances.
The results presented in Table 3 were gathered after comprehensive and exhaustive tests with different combinations of architectures and input variables, such as the amount of LPC/cepstral coefficients. Analyzing only the average recognition rates, one can see that the worst rate was achieved using the MLP network (81.1 %). With regard to the SOM, the *K*-means and the DTW algorithm, the achieved recognition rates were similar, with the best performances being achieved by the SOM:PDS and the DTW algorithm.

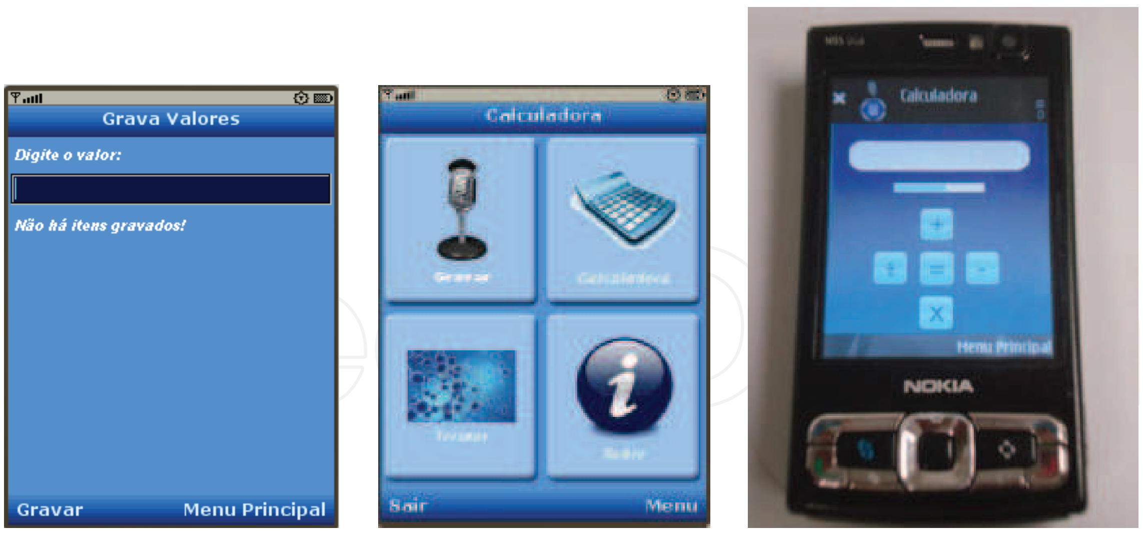
Algorithms	**Recognition rates (%)**				**Time (ms)**	
	mean	*max*	*min*	*std*	*training*	*test*
SOM original	87.7	92.9	82.9	3.22	2,911,873.47	8.14
SOM:SWS+PDS+Rectangular	85.8	92.2	79.4	4.41	2,275,220.35	4.48
SOM:PDS	89.6	92.9	85.1	2.39	2,860,639.27	3.97
SOM:PDS+Rectangular	88.2	92.9	84.4	2.80	2,275,061.98	4.07
SOM:PDS+TruncatedGauss	87.9	93.6	83.7	2.69	2,661,322.49	5.80
TS-SOM	82.5	88.6	75.9	3.69	3,310.01	7.71
MLP	81.1	86.5	72.3	4.09	93,642.47	0.21
KMeans	88.4	92.9	82.3	2.31	154,029.93	8.12
DTW	89.8	93.6	82.9	3.02	0.0012	3.266.43

Table 3. Performances of the evaluated classifiers.

	/0/	/1/	/2/	/3/	/4/	/5/	/6/	/7/	/8/	/9/	/+/	/-/	/*/	/÷/	/r/	/l/	/v/
/0/	**99.3**					0.7											
/1/		**96 .5**	0 .6		1 .2	0 .6				1 .2							
/2/		0 .6	**86 .1**	0 .6			2 .9			1 .2			8 .7				
/3/			1 .4	**83 .9**			7 .7						7 .0				
/4/					**88 .1**				1 .7	9 .0	0 .6					0 .6	
/5/						**96.2**	0.5						2.2			1.1	
/6/	1.8		1.2	13.9			**80.6**					1.2	1.2				
/7/	2 .8		0 .7		2 .1	2 .8	1 .4	**87.9**		2 .1							
/8/	1.2	6.1	3.0		0.6		1.8		**85.4**				0.6		0.6		0.6
/9/		3.5			0.6					**95.3**						0.6	
/+/							2.9			1.7	**94.9**		0.6				
/-/	0.6	2.5			0.6	3.2	0.6				0.6	**88.6**	3.2				
/*/			5.9	4.9			0.5					5.9	**82.7**				
/÷/	2.7					6.7								**89.9**			0.7
/r/									0.7						**98.7**		0.7
/l/		0.6													4.3	**93.8**	1.2
/v/																0.5	**99.5**

Table 4. Confusion matrix for the DTW algorithm.

Table 4 shows the confusion matrix for the DTW algorithm. On the one hand, the digit /0/ (zero) and the word /v/ (back) presented average successful recognition rates of 99.3% and 99.5%, respectively. On the other hand, most of the errors occurred for the digits /6/ and /3/. This can be partly explained if we note that the utterances of the words /tres/ (3) and /seis/ (6) are very similar in Portuguese, specially their final parts.
Despite the good performance the DTW algorithm, it is important to emphasize that the choice of the best classifier to be embedded in the smartphone should also take into account the required training and testing times. In what concern the online tests, there are two main approaches to embed a speech classifier in a mobile phone. The first performs the training phase off-line in a desktop and, once the classifier is trained, one just uploads the classifier's parameters to the phone. The second approach performs both the classifier training and testing in the phone itself.
From Table 3 one can see that the SOM:PDS does not satisfy the time restrictions to perform the online training phase, but it can be trained offline and their parameters then can be uploaded to the phone. Furthermore, although DTW had achieved good results, its testing time is too long, and then, it cannot be inserted into the embedded device. The option that achieved the best tradeoff between good performance and low processing times relied on the TS-SOM network. We also performed
In order to evaluate the feasibility of embedding the TS-SOM and SOM:PDS networks for real-time speech recognition purposes, we developed a speaker-independent voice-driven calculator on the Nokia N95 smartphone. This phone has a ARM11 processor with clock about 300 MHz. The application was developed using the JME (*Java Micro Edition*) framework. Some snapshots of the graphical interface developed for this embedded application are illustrated in Figure 3.
We defined two metrics to evaluate the TS-SOM and the SOM:PDS classifiers in terms of their processing times when running in the N95 phone, from the user point of view. The first one is the processing time that represents the time actually elapsed to classify an utterance, calculated by the mobile phone itself. The second one is the time perceived by user (response time), calculated by user with a chronometer. The tests were performed with four different words, two of them considered as long words, /*dividido*/ (divided by) and /*resultado*/ (result) and two of them considered short ones, /*um*/ (one) and /*dois*/ (two). The average values for the evaluated metrics are illustrated in Figure 4.
One can see from Figure 4 that while the processing times are all smaller than 1.25 seconds, the response time lasts from 2.10 to 3.25 seconds. The difference between these metrics occurs because a progress bar was created in the application in order to wait for the user utterance.



(a) Speech recording screen. (b) Main menu screen. (c) Calculator screen embedded in Nokia N95 smartphone.

Fig. 3. Snapshots of the voice-driven software calculator.

The pre-processing and recognition algorithms are performed only when this bar reaches its endpoint. Thus, there is fixed time which is wasted while the bar is moving.
Finally, the TS-SOM was evaluated in the online training task. The tests indicated an average time of 14 seconds of training, considering that 34 utterances (17 classes) were recorded on the mobile phone for a single speaker (the owner of the phone!). After this training period, the performance of the TS-SOM classifier was evaluated, reaching values similar to those achieved for the off-line tests (around 78-80%). It is worth noticing, that there is a natural degradation of the recognition performance of the classifier when it is embedded in the smartphone, since the

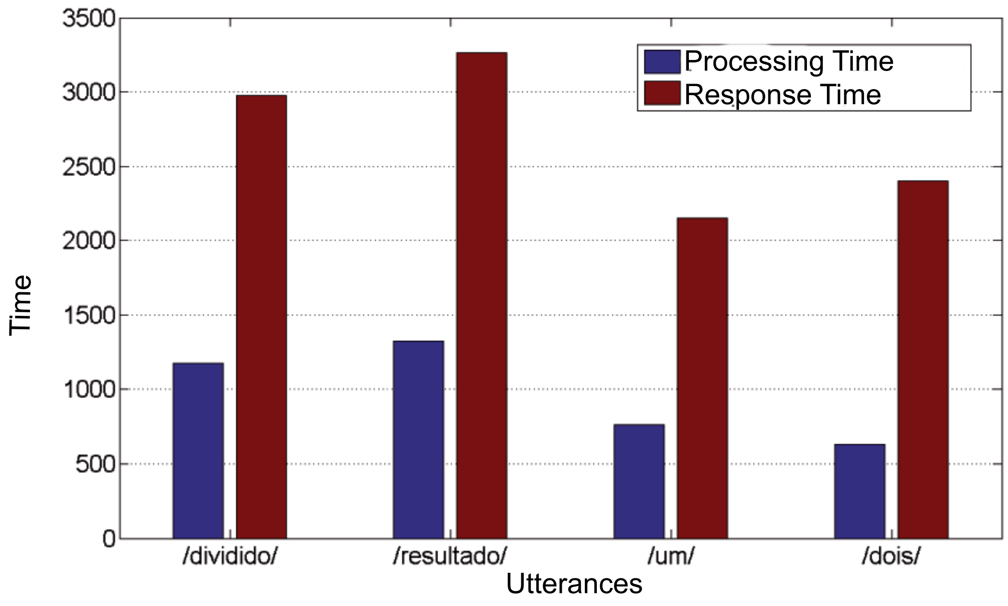


Fig. 4. Processing and response times (in milliseconds) for four words: /*dividido*/ (divided by), /*resultado*/ (result), /*um*/ (one) and /*dois*/ (two).

acquisition of the speech data is made under conditions which are hard to control in real-world scenarios, for example, the distance of the user from the microphone, noise level and the low-quality of the microphone device.

5. Conclusions

In this paper we analyzed the feasibility of embedding neural network based speech recognition systems into smartphones. We were motivated by the interest in developing a voice-driven software calculator for the Brazilian Portuguese language. For this purpose we first described a number of software techniques to speedup SOM computations, then described the TS-SOM network. The next step was to compare the offline classification performances of the SOM and the TS-SOM networks with those provided by classifiers commonly used for speech recognition purposes, such as the MLP, the *K*-means and the DTW algorithms. From these experiments we were able to select those architectures that can be embedded into a Nokia N95 smartphone for online recognition of words. The choice was based on the tradeoff between high recognition rates and a low processing times. Based on these restrictions, the selected models were the SOM:PDS and TS-SOM networks. From these two classifiers, only the TS-SOM classifier can be trained and tested online, i.e. inside the smartphone itself.
Currently we are optimizing the Java code used to implement the TS-SOM and the SOM:PDS in the embedded application in order to further diminish their processing times. We are also evaluating other speech parameterization techniques, such as wavelets, in order to increase the recognition rates of those two classifiers.

6. References

Alhonen, J., Cao, Y., Ding, G., Liu, Y., Olsen, J., Wang, X. & Yang, X. (2007). Mandarin short message dictation on symbian series 60 mobile phones, *Proceedings of the 4th international conference on mobile technology, applications, and systems (Mobility '07)*, pp. 431–438.

Amerijckx, C., Verleysen, M., Thissen, P. & Legat, J.-D. (1998). Image compression by self-organized Kohonen map, *IEEE Transactions on Neural Networks* 9(3): 503–507.

Barreto, G. A. & Araújo, A. F. R. (2004). Identification and control of dynamical systems using the self-organizing map, *IEEE Transactions on Neural Networks* 15(5): 1244–1259.

Barreto, G. A., Araújo, A. F. R. & Ritter, H. J. (2003). Self-organizing feature maps for modeling and control of robotic manipulators, *Journal of Inteligent and Robotic Systems* 36(4): 407–450.

Bei, C.-D. & Gray, R. (1985). An improvement of the minimum distortion encoding algorithm for vector quantization, *IEEE Transactions on Communications* 33(10): 1132–1133.

Cho, J., Principe, J., Erdogmus, D. & Motter, M. (2006). Modeling and inverse controller design for an unmanned aerial vehicle based on the self-organizing map, *IEEE Transactions on Neural Networks* 17(2): 445–460.

Deller, J. R., Hansen, J. H. L. & Proakis, J. G. (2000). *Discrete-Time Processing of Speech Signals*, John Wiley & Sons.

Frota, R. A., Barreto, G. A. & Mota, J. C. M. (2007). Anomaly detection in mobile communication networks using the self-organizing map, *Journal of Intelligent and Fuzzy Systems* 18(5): 493–500.

Gas, B., Chetouani, M., Zarader, J. & Feiz, F. (2005). The predictive self-organizing map :

application to speech features extraction, *in* M. Cottrell (ed.), *Proceedings of the 5th Workshop on Self-Organizing Maps (WSOM'2005)*, pp. 497–504.

Guillén, A., Herrera, L., Rubio, G., Pomares, H., Lendasse, A. & Rojas, I. (2010). New method for instance or prototype selection using mutual information in time series prediction, *Neurocomputing* 73(10-12): 2030–2038.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps, *Biological Cybernetics* 43: 59–69.

Kohonen, T. (1988). The 'neural' phonetic typewriter, *Computer* 21(3): 11–22.

Kohonen, T. K. (1997). *Self-Organizing Maps*, 2nd extended edn, Springer-Verlag, Berlin, Heidelberg.

Kohonen, T. K. (2001). *Self-Organizing Maps*, 3rd edn, Springer-Verlag, Berlin, Heidelberg.

Koikkalainen, P. & Oja, E. (1990). Self-organizing hierarchical feature maps, *International Joint Conference on Neural Networks (IJCNN'90)*, pp. 279–284 vol.2.

Lendasse, A., Lee, J., Wertz, V. & Verleysen, M. (2002). Forecasting electricity consumption using nonlinear projection and self-organizing maps, *Neurocomputing* 48: 299–311.

Martin, C., Diaz, N. N., Ontrup, J. & Nattkemper, T. W. (2008). Hyperbolic SOM-based clustering of DNA fragment features for taxonomic visualization and classification, *Bioinformatics* 24(14): 1568–1574.

Niskanen, M., Kauppinen, H. & Silvan, O. (2002). Real-time aspects of SOM-based visual surface inspection, *Proceedings SPIE Machine Vision Applications in Industrial Inspection*.

Oh, K.-S. & Jung, K. (2004). Gpu implementation of neural networks, *Pattern Recognition* 37: 1311–1311.

Olsen, J., Cao, Y., Ding, G. & Yang, X. (2008). A decoder for large vocabulary continuous short message dictation on embedded devices, *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2008)*, pp. 4337–4340.

Rabiner, L. & Juang, B.-H. (1993). *Fundamentals of speech recognition*, Prentice-Hall International.

Rabiner, L. R. & Schafer, R. W. (1978). *Digital Processing of Speech Signals*, Prenctice-Hall, New Jersey.

Sagheer, A., Tsuruta, N., Maeda, S., Taniguchi, R.-I. & Arita, D. (2006). Fast competition approach using self-organizing map for lip-reading applications, *International Joint Conference on Neural Networks (IJCNN'06)*, pp. 3775–3782.

Sakoe, H. & Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition, *IEEE Transactions on Acoustics, Speech and Signal Processing* 26(1): 43–49.

Scanzio, S., Cumani, S., Gemello, R., Mana, F. & Laface, P. (2010). Parallel implementation of artificial neural network training for speech recognition, *Pattern Recognition Letters* 31: 1302–1309.

Somervuo, P. (2000). Competing hidden markov models on the self-organizing map, *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00)*, Vol. 3, pp. 169–174.

van Hulle, M. (2010). Self-organizing maps, *in* G. Rozenberg, T. Baeck & J. Kok (eds), *Handbook of Natural Computing: Theory, Experiments, and Applications*, Springer-Verlag, pp. 1–45.

Varga, I. & Kiss, I. (2008). Speech recognition in mobile phones, *in* Z.-H. Tan & B. Lindberg (eds), *Automatic Speech Recognition on Mobile Devices and over Communication Networks*, Springer, pp. 301–325.

Wang, J.-F., Wang, J.-C., Mo, M.-H., Tu, C.-I. & Lin, S.-C. (2008). The design of a speech interactivity embedded module and its applications for mobile consumer devices, *IEEE Transactions on Consumer Electronics* 54(2): 870–875.

Xiao, Y., Leung, C. S., Ho, T.-Y. & Lam, P.-M. (2010). A GPU implementation for LBG and SOM training, *Neural Computing and Applications* . DOI: 10.1007/s00521-010-0403-7.

Yin, H. (2008). *Computational Intelligence: A Compendium*, Vol. 115 of *Studies in Computational Intelligence*, Springer-Verlag, chapter The Self-Organizing Maps: Background, Theories, Extensions and Applications, pp. 715–762.