Spatio-Temporal Pattern Classification with KernelCanvas and WiSARD

Diego F. P. de Souza
and Felipe M. G. França
System Engineering and Computer Science Program - COPPE
Universidade Federal do Rio de Janeiro
Rio de Janeiro, Brazil
Email: diegosouza@cos.ufrj.br
felipe@cos.ufrj.br

Priscila M. V. Lima
Instituto Tércio Pacitti (NCE)
Universidade Federal do Rio de Janeiro
Rio de Janeiro, Brazil
Email: priscilamvl@gmail.com

***Abstract*—This work proposes a new method, KernelCanvas, that is adequate to the Weightless Neural Model known as WiSARD for generating a fixed length binary input from spatio-temporal patterns. The method, based on kernel distances, is simple to implement and scales linearly to the number of kernels. Five different datasets were used to evaluate its performance in comparison with more widely employed approaches. One dataset was related to human movements, two to handwriteen characters, one to speaker recognition and the last one to speech recognition. The KernelCanvas combined with WiSARD classifier approach frequently achieved the highest scores, sometimes losing only for the much slower K-Nearest Neighbors approach. In comparison with other results in the literature, our model has performed better or very close to them in all datasets.**

I. Introduction

Spatio-temporal data (STD), which is data that represents temporal or sequential patterns, is present in many real problems, like: handwritten character recognition, speech recognition, human activity recognition, and others. In fact, according to [1], most of the data we deal today is STD in nature. This kind of data, is frequently generated from the observation of some kind of signal for an amount of time and is usually represented by a matrix of dimension NXM. Semantically, these two variables correspond to N signals observed during an amount of time, or length, M, which is not constant. In practice, even samples belonging to the same class may have different lengths, depending on the speed they are performed. This corresponds to one of the factors which make this type of problem hard to solve, since most of the classifiers in existence expect an input of fixed length.

There are basically three ways to deal with STD classification. The first approach consists in directly comparing the samples through the use of Dynamic Time Warping (DTW), which is a technique to calculate the distance between two patterns with different lengths, in conjunction with the K-Nearest Neighbors. Although this is a slow method, it can be combined to other approaches to achieve real-time performance [2]. The second approach assumes the variation in the length of the pattern occurred in a linear fashion. Thus, a simple resampling to a standard length is enough to equalize all pattern sizes. The third approach, consists in keeping an internal state inside the model which varies as a function of the observations that are presented to it. This is basically the idea behind models like Hidden Markov Model [3] and Echo State Networks [4]. In fact, this is also the idea behind the KernelCanvas, which resembles an Echo State Networks, constructed in a different way.

This paper is structured as follows. In Section II we describe the proposed model, along with other related works. Section III describes the five datasets chosen for evaluation, followed by the preprocessing steps performed on them. Section IV describes the six approaches taken to classify the datasets, together with respective configurations. That section also presents a User-Dependent Cross-Validation test comparing each approach, comparative results in standard tasks and extra analysis. Finally, Section V presents an overall description of results obtained and some considerations for future research.

II. Models

Four classifier models have been chosen in order to compare the results obtained. These are: The K-Nearest Neighbors (KNN) [5], the standard WiSARD (Wilkie, Stonham and Aleksander's Recognition Device) [6], an Echo State Networks (ESN) [4] and our proposed model called KernelCanvas combined with the model called WiSARD. The rest of this section briefly describes each of these models.

A. KNN

This is the standard K-Nearest Neighbors classifier using Dynamic Time Warping (DTW) [7] as distance measure. The training of this model consists in storing a copy of each sample presented during training phase, along with its corresponding classification. In the classification phase, the K most similar samples are selected, according to some distance measure, and the class mostly represented is selected as prediction. Naturally, selecting the K most similar samples is an expensive operation so, for this reason, it is a common practice to use this model in conjunction with some spatial structure, like k-d trees [8]. Unfortunatelly, this kind of structure is not compatible with DTW, since it involves a non-linear search in space when both samples are compared. Despite this, this model has the advantages of being easy to implement and usually providing good results.

978-1-4799-5618-0/14 $31.00 © 2014 IEEE
DOI 10.1109/BRACIS.2014.49

With respect to Dynamic Time Warping, this is a distance measure that tries to find the best matching between each sample's observation in a non-linear manner. For instance, consider two samples U and V, both with lengths n and m, respectively. In order to calculate the similarity between $U = [u_1, u_2, ..., u_n]$ and $V = [v_1, v_2, ..., v_m]$ with this technique, a matrix D of size nxm with each coordinate (i, j) containing the euclidian distance between observation u_i and v_j must be built. Then, a path search from coordinate $(1, 1)$ to coordinate (n, m), minimizing the sum of distances in the path is performed. The distance between U and V is the sum of the distances in this minimum path.

In order to optimize the search, we applied standard A* search algorithm [9] calculating only the necessary coordinates (i, j). Additionally, we shuffled all samples and kept the worst from the K best DTW distances found during each classification, interrupting searches whose cost already exceeded this value.

B. ESN

Echo State Networks [4], also named Reservoir Neural Network or Liquid State Machine, is a special type of recurrent neural network. The main difference when compared to other traditional networks is the presence of statically and randomly generated connections between most of its neurons. Furthermore, this network also contains an internal state which is updated according to Equation 1.

$$x(t+1) = f(W^{in}u(t+1) + Wx(t) + W^{back}y(t)) \quad (1)$$

where $x(t)$ is the network inner state at time t, f is the internal neurons output function, W^{in} is the weight matrix connecting the input vector with the inner state, $u(t)$ is the input at time t, W is the weight matrix connecting the inner state to itself, W^{back} is the weight matrix connecting the network output to the inner state and $y(t)$ is the network output at time t.

Note that none of the weight matrices in Equation 1 is defined during training. They are all statically and randomly generated, the only trainable weights matrix is W^{out}, which is shown in Equation 2.

$$y(t+1) = f^{out}(W^{out}(u(t+1), x(t+1), y(t))) \quad (2)$$

where f^{out} is the output function, W^{out} is the trainable output weight matrix and $(u(t+1), x(t+1), y(t))$ is the concatenation of the input vector, the inner state and the previous output state.

Multiple training strategies can be applied to W^{out}, as well as different output functions can be used. However, the most commonly used, and fastest, is linear regression, which is what we used in this work. We applied the Oger toolbox[1] with leaking neurons. A larger and more complete description of ESN, including leaking neurons, can be found in [4].

[1]Available at: http://organic.elis.ugent.be/organic/engine

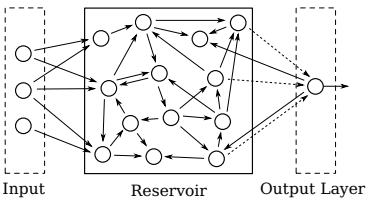

Fig. 1. An example of ESN network composed of 3 input nodes, 13 inner nodes and 1 output node. Dashed arrows indicate trainable connections

C. WiSARD

Created by Wilkie, Stonham and Aleksander, this classifier differs from traditional models in the way it stores information, which is in units of RAM memory instead of weights matrices [6]. WiSARD (Wilkie, Stonhan and Aleksander Recognition Device) architecture is composed by units called discriminators, each one corresponding to a class of the problem. Each discriminator, in turn, is composed by multiple RAM units, all initially containing zeros. Each RAM receives as input B addressing bits, used to address its content when a pattern is presented. Each of these bits is randomly and distinctly connected to the input vector. Since they are used to address the RAM, it is necessary to convert the pattern vector to a binary format previously.

During the training phase, the input pattern is presented only to the discriminator corresponding to the correct class. At this moment, the addressed positions inside the RAMs of the discriminator in question are set to 1. Classification works in a similar way, except that the input pattern is presented to all discriminators. Each discriminator returns its activation, which is the sum of the contents stored in its addressed memory positions, see Figure 2. The WiSARD prediction is the class whose discriminator had the highest activation.

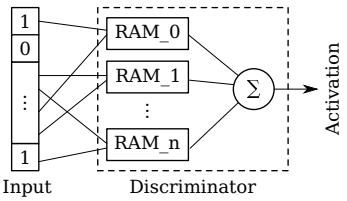

Fig. 2. A WiSARD discriminator containing n RAMs, each receiving two randomly selected address bits from the input ($B = 2$).

D. KernelCanvas + WiSARD

This model's idea comes from the analogy to painting a canvas. During this process, a paintbrush may paint the canvas multiple times until the painting is finished. Once it is completed, no matter how many strokes were drawn, the canvas will maintain its original dimension.

The KernelCanvas works analogously, except that instead of directly using a matrix to represent the canvas, the method uses randomly generated kernels to represent pieces of the canvas. The main reason for this is to accept larger than two-dimensional inputs without having to store multi-dimensional matrices in memory. In the traditional painting process, when a

stroke composed of N pairs of coordinates touchs the canvas it paints these coordinates and many other neighbor coordinates, depending on the size of the paintbrush. Similarly, when a pair of coordinates is presented to the KernelCanvas, the nearest kernels are selected and the bits they represent are set from 0 to 1. The final picture is composed by these bits, which are presented as input to the WiSARD.

Only three parameters are necessary to be defined: (*i*) the number of kernels K in the canvas; (*ii*) the percentage S of kernels that are activated when each coordinate is presented; and (*iii*) the number of bits B each kernel represents. The dimensionality of the kernels is not a parameter, since it must be the same of the coordinates that are presented to it. The basic ideia of the process is shown in Figure 3, each coordinate pair is presented to the canvas individually and the nearest kernels are selected. When all coordinates have been presented, the "painted" canvas is input to the WiSARD either for classification or training.

In this work, each kernel dimension was uniformly generated in the range $[-1.0, +1.0]$ and the comparison method adopted was the euclidian distance. Since all input coordinate variables were in the domain $[-\infty, +\infty]$, they also had to be normalized using $tanh(zscore(X))$ before being compared to the kernels.

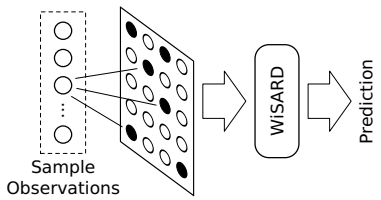

Fig. 3. A KernelCanvas structure composed of 20 kernels. Each pair is presented to the Canvas which is then presented as input to the WiSARD network.

III. Datasets and Pre-processings

In order to evaluate the models, five datasets were chosen. They were: Libras [10], Pendigits [11], UJIPenchars2 [12], Japanese Vowels [13] and Arabic Digits [14]. Datasets were chosen due to their distinct origins which would help to evaluate the models in different types of temporal problems. Below is a short description of them, additional information is summarized in Table I.

- *Libras:* This is a hand movement dataset for the Brazilian sign language composed by fifteen classes captured by video. All samples have been time resampled by the authors, fitting them in a 45 frame length. The input of this dataset is the coordinates of the user's hand centroid pixel, in each frame, normalized to fit into a unitary space.
- *Pendigits:* This is a handwriten digits dataset created by 44 writers, each one generating 250 samples. Samples were acquired through a pressure sensitive tablet with LCD display. Despite the capability to capture pressure signals, only the X and Y coordinates were provided by the author. This dataset is divided into two sets, a training set composed of 30 writers and a training set composed of 14 writers.

TABLE I. Summary of datasets characteristics

	Libras	**Pendigits**	**UJIpenchars2**	**Japanese Vowels**	**Arabic Digit**
Samples	360	7494	7440	640	8800
Classes	15	10	35	9	10
Length (std)	45 (0.0)	40.59 (12.18)	60.86 (25.89)	15.56 (3.62)	39.81 (8.55)
Recognition Task	Hand Movement	Handwritten Characters	Handwritten Characters	Speaker	Speech

- *UJIPenchars2:* This is the dataset with the largest number of classes. It contains 97 classes, including handwriten letters, digits and other non-ASCII symbols. Despite this, in order to support comparison with [2] we restricted ourselves to 35 of these classes. These included all 10 digits and all 26 letters (lower and uppercase), digit "0" was included in class "o". This dataset also provides separate training and testing sets. The former was generated from 40 first writers and the latter from the 20 remaining.
- *Japanese Vowels:* This is a dataset for speaker recognition. It is a 9-class problem created using nine male speakers uttering the vowels /ae/ successively. Each sample was preprocessed through the application of a 12-degree linear prediction analysis, generating 12 features per observation as input. Additionaly, this dataset is also divided into training and testing set for comparisson with the sizes of 270 and 370 samples, respectively.
- *Arabic Digits:* This is another digits-only dataset but with spoken input. This was created from 88 native arabic speakers (44 males and 44 females), each of them repeating all 10 digits 10 times. All samples are represented with 13 Mel Frequency Cepstral Coefficients (MFCCs).

As has alread been said, those datasets contain data collected in distinct ways and of multiple natures. Despite this, the preparation steps applied to them was quite similar. These steps can be described as follows and the order they are executed is shown in Figure 4.

- *Absolute Coordinates and Directions:* Some datasets are expressed in absolute coordinates (Libras, Pendigits and UJIPenchars2) and others are composed of frequency measures (Japanese Vowels and Arabic Digits). In order to properly classify all of them, we chose to normalize their format into both absolute coordinates and relative directions. This is done by calculating the relative direction in the absolute coordinates datasets. In the frequency datasets, we considered them as being similar to relative direction data and incrementally added their values generating absolute values.
- *Center in Origin:* Subtracts the individual average of each sample attribute, centering it in origin.
- *Rotations:* In datasets Libras, Pendigits and UJIPenchars2 each sample is rotated generating two others. The first rotated by -10 degrees and the second by +10 degrees. Naturally, this was applied only to the training set of each test performed.
- *Temporal or Spatial Resampling:* This is a commonly used technique where samples are resampled to fit in a fixed time interval. In temporal resampling points are selected by linearly spliting the time steps of the original

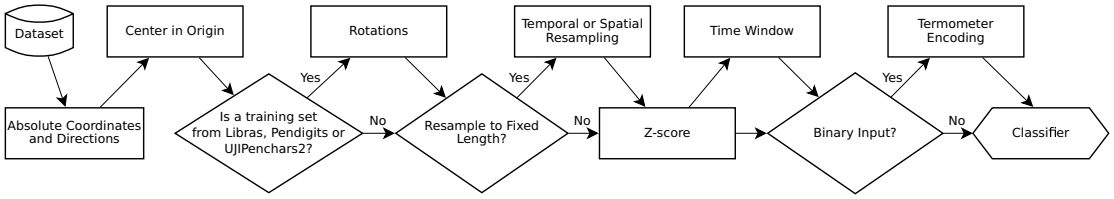

Fig. 4. Fluxogram illustrating the order each preprocessing step was applied before presenting the dataset to the classifier.

sample. In spatial resampling, points are selected by linearly spliting the length of the original sample. The main advandage of these approaches is their simplicity.

- *Z-score:* This is the standard technique used to normalize continuous variables. First, the mean and standard deviation of each attribute in the entire dataset is calculated, then we update each attribute by subtracting its mean and dividing by its standard deviation. After this step, all attributes have their new mean equal to zero and a standard deviation equal to one.
- *Time Window:* The observation in time t is the concatenation of the N previous observations. Datasets UJIPenchars2 and Pendigits were trained with a window of size two and the others had size three. Values outside the original sample data were considered to be equal to zero.
- *Termometer Encoding:* This step is responsible for transforming continuous variables into binary representations of length T and is only necessary for the WiSARD network. To accomplish this, the method first divides a continuous interval into T regular intervals, each one representing one bit. If the value we want to represent is higher than the interval's starting point its corresponding bit is set to one, otherwise it is set to zero. Since variables were already normalized by z-score, we simply applied hyperbolic tangent to distribute them in the range $[-1.0, +1.0]$ and then applied termometer encoding with $T = 20$.

IV. Experimental Results

A. Model Comparison

Six approaches were prepared varying the models described in Section II. These were:

(*i*) *KernelCanvas and WiSARD:* The KernelCanvas with 20 bits per kernel, 512 kernels in Libras and 2048 on the others, 7.5% painting activation in Libras and Japanese Vowels and 5% on the others. The WiSARD had 8 addressing bits in Libras and Japanese Vowels, 32 in UJIPenchars2 and 16 on all others.

(*ii*) *Temporal Resampling and WiSARD:* Samples were resized to 10 units of time (except UJIPenchars2, which was resized to 30) and encoded with termometer encoding using 20 bits. The encoded sample was then classified by WiSARD networks with 8 addressing bits in Libras and Japenese Vowels and 16 on remaining.

(*iii*) *Spatial Resampling and WiSARD:* Same as the previous configuration but using Spatial Resampling.

(*iv*) *ESN and Linear Regression:* The default ESN model with Linear regression as classifier. This ESN had 100 neurons and a leakink factor of 0.3 in all datasets.

TABLE II. Accuracy and Standard Deviation for User-Dependent Stratified Cross-Validation (%)

	Libras	**Pendigits**	**UJIpenchars2**	**Japanese Vowels**	**Arabic Digit**
i	96.78 (2.06)	99.55 (0.20)	91.57 (1.19)	97.17 (2.34)	99.05 (0.17)
ii	91.89 (4.42)	99.37 (0.19)	89.25 (1.35)	95.91 (2.66)	98.95 (0.27)
iii	91.11 (4.66)	99.01 (0.49)	86.60 (1.16)	95.60 (2.25)	98.27 (0.31)
iv	83.33 (3.95)	92.93 (0.88)	53.23 (1.13)	93.24 (4.17)	77.09 (1.31)
v	94.44 (3.05)	98.08 (0.34)	66.40 (1.64)	95.28 (2.93)	92.42 (0.72)
vi	94.78 (1.89)	99.87 (0.17)	93.66 (1.01)	96.24 (1.70)	99.74 (0.08)

(*v*) *ESN and WiSARD:* Same as the previous configuration but using a WiSARD as the output classifier with 16 addressing bits in Pendigits and UJIPenchars2, and 8 on all others. ESN output was also encoded with termometer encoding using 20 bits.

(*vi*) *KNN and DTW:* A 3-Nearest Neighbors using Dynamic Time Warping as distance measure.

The evaluation strategy chosen for model comparison was stratified 10-fold cross-validation. That is, each fold was generated by randomly selecting samples from the original dataset while keeping its original class distribution. In addition to this, folds were reused in all evaluations for a fair comparison.

The highest accuracies were, basically, divided between KernelCanvas (i) and KNN (vi), see Table II. The former achieved the best results in Libras and Japanese Vowels datasets, while the latter achieved the best results on the remaining. Additionally, the KNN model also obtained the smallest standard deviations in each model. Among the models using ESN, the WiSARD model (v) performed best in all datasets when compared to the Linear Regression approach (iv). Finally, Time Resampling (ii) and Length Resampling (iii) also performed well, achieving accuracies close to the best results on most datasets.

With exception of Japanese Vowels, which is a speaker recognition dataset, it is generaly more interesting to know how well these models generalize to users who are not in the training set. Such tests are called user-indepent and constitute a stronger test for the models generalization capacity. For this type of evaluation, and to facilitate comparison with other works in literature, we also applied our model to the standard tasks provided by each author. In this evaluation, only the

TABLE III. KERNELCANVAS RESULTS FOR LIBRAS STANDARD TASKS (%)

instance id	mean recognition rate	standard deviation	median recognition rate	max recognition rate
1	97.42	1.89	97.78	100.00
2	99.47	0.65	98.89	100.00
3	98.49	1.66	100.00	100.00
4	97.73	2.49	96.67	100.00
5	79.02	2.95	82.22	84.44
6	97.60	1.31	96.67	100.00
7	99.16	0.58	100.00	100.00
8	78.84	3.02	84.44	84.44
9	79.47	2.59	77.78	86.67

KernelCanvas was used since it presented the best results and acceptable classification times.

In Pendigits, we achieved 98.60% accuracy, recent results in literature achieved 99.40% [2]. In UJIPenchars2, we achieved 88.55% and the best results found were 91.80%, also in [2]. In Japanese Vowels, we achieved 96.80%. Comparatively, recent results obtained 98.00% [15]. Finally, accuracy in Arabic Digits was 96.70%, while recent results have found 99.31% [16].

Regarding Libras, five groups of data are provided and nine test instances are proposed. The five first are considered user-dependent tasks by the author, whereas the last four are user-independent. Instances 1, 4 and 6 were, each, divided into training and test sets and 25 models were trained. In the original article, only the best model, in each of the three instances, was selected and tested in the datasets from instances (2 and 3), 5 and (7, 8 and 9), respectively. Since neither WiSARD nor KernelCanvas suffer from overtraining, instead of just selecting one network to be tested we tested all of them and also collected average statistics about their results, without compromising the comparisons. A more detailed description about these tasks and how they must be performed can be found in [10]. Results are presented in Table III.

B. Performance Analysis

In this kind of problem it is also expected that the classifications occur in real-time. Therefore, it is also important to analyse the average time per classification. This section discusses both training (T) and classification (C) times found, summarized in Table IV. All values in this table are expressed in milliseconds.

As expected, the results collected verified that the fastest approach constitutes the simple resampling of all samples to a fixed length and classifying them (ii and iii). Despite the good results presented by the KNN in the previous section, this model required an extremely large amount of time to classify each sample (vi). That means that a direct aproach with this model would not be applicable in real-time problems. In respect to KernelCanvas and WiSARD (i), most of their times were higher than the other models, although they all still qualify for real-time problems. Finally, training and classification times with ESN and WiSARD (v) were substantially higher when compared to ESN and Linear Regression (iv). Further research could devise new ways to optimize their integration.

TABLE IV. AVERAGE TRAINING (T) AND CLASSIFICATION (C) SAMPLE TIMES PERFORMED BY EACH MODEL DURING CROSS-VALIDATION (MS)

	Type	Libras	Pendigits	UJIpenchars2	Japanese Vowels	Arabic Digit
i	T	0.922	3.666	6.356	6.158	14.934
ii	T	0.259	0.182	1.063	1.708	3.182
iii	T	0.301	0.200	1.145	1.946	3.269
iv	T	3.290	2.514	4.264	1.283	2.864
v	T	45.098	33.518	50.133	13.141	33.739
vi	T	-	-	-	-	-
i	C	1.085	3.880	6.741	6.629	15.748
ii	C	0.329	0.202	1.160	2.136	3.229
iii	C	0.381	0.227	1.233	2.356	3.292
iv	C	2.910	2.479	3.745	1.264	2.992
v	C	50.186	76.940	121.471	16.948	85.142
vi	C	121.309	1342.170	4663.740	17.659	887.985

C. Kernel Analysis

In this section we present the results obtained by varying the number of kernels in the KernelCanvas. The final objective was to analyse the scalability of the model and at the same time to find the highest and most stable accuracy, while still qualifying for real-time classification. In each analysis, the number of kernels was exponentially varied between 1 and 2048. Results are presented after application of log2 on the x-axis.

As expected, the higher the number of kernels the higher is the accuracy obtained, see Figure 5 (a). In addiction to this, the model has demonstrated to be relatively efficient for small quantities of kernels. For instance, for $x = 5$ (32 kernels) the model obtained accuracies above 90% for both Libras and Pendigits datasets. Indicating that the model may be used in cases where accuracy may be slightly reduced in favor of more performance. Furthermore, as of $x = 8$ (256 kernels) the accuracy does not seem to grow expressively anymore.

Other important factor found refers to the consistency of the results, see Figure 5 (b). The bigger the number o kernels, smaller appears to be the accuracy standard deviation, suggesting a bigger confidence in the model generalization when applied to external data.

Finally, both training and classification times appear to grow linearly as a function of the number of kernels, see Figure 5 (c) and (d), respectively. Demonstrating the good scalability of the model face its higher complexity. After balancing accuracy and training with classification times, we opted for using 512 kernels with the Libras dataset and 2048 with all the others.

V. CONCLUSION

The model proposed in this work is a new fast kernel based method for spatio-temporal data classification which uses a WiSARD network for output classification. Our work has demonstrated that this model performs quite well in a variety of problems involving spatio-temporal patterns, outperforming traditional approaches like ESN and KNN. When compared to other works, this model also achieved very competitive accuracies in all standard tasks, without too customized encodings.

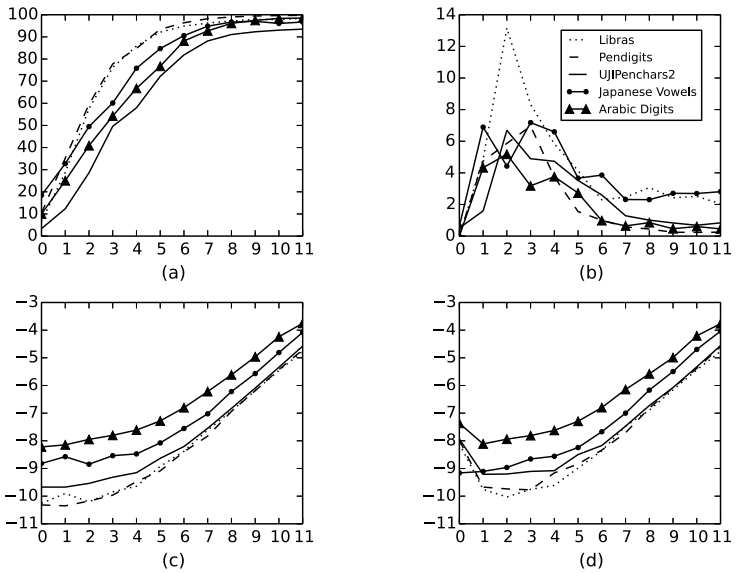

Fig. 5. Results obtained by KernelCanvas in function of the number of kernels (normalized In log2): (a) Average accuracy, (b) Standard deviation of the accuracy, (c) Log2 of the average time needed to train each sample, (d) Log2 of the average time needed to test each sample.

Additionally, we have evaluated the WiSARD model as an alternative to the traditional Linear Regression method used in conjunction with ESN. In this test, the former obtained results considerably highers than the latter. Furthermore, the use of WiSARD also extends the ESN allowing it to learn incrementally. Despite all this, its training and classification times were also higher, possibly due to current implementation. Finally, simplistic approaches like Time Resampling presented reasonably good final accuracies for extremely low training and classification times, making them ideal for slower devices.

REFERENCES

[1] H. N. A. Hamed, N. Kasabov, S. M. Shamsuddin, H. Widiputra, and K. Dhoble, "An extended evolving spiking neural network model for spatio-temporal pattern classification," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*. IEEE, 2011, pp. 2653–2656.

[2] M. J. Castro-Bleda, S. Espafia, J. Gorbe, F. Zamora, D. Llorens, A. Marzal, F. Prat, and J. Vilar, "Improving a dtw-based recognition engine for on-line handwritten characters by using mlps," in *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*. IEEE, 2009, pp. 1260–1264.

[3] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[4] H. Jaeger, "The echo state approach to analysing and training recurrent neural networks-with an erratum note," *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, vol. 148, p. 34, 2001.

[5] T. Cover and P. Hart, "Nearest neighbor pattern classification," *Information Theory, IEEE Transactions on*, vol. 13, no. 1, pp. 21–27, 1967.

[6] I. Aleksander, W. V. Thomas, and P. A. Bowden, "WISARD: A radical step foward in image recognition," *Sensor Review*, vol. 4, p. 120–124, July 1984.

[7] M. Müller, "Dynamic time warping," *Information retrieval for music and motion*, pp. 69–84, 2007.

[8] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[9] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, no. 2, pp. 100–107, 1968.

[10] D. B. Dias, R. C. Madeo, T. Rocha, H. H. Biscaro, and S. M. Peres, "Hand movement recognition for brazilian sign language: a study using distance-based neural networks," in *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*. IEEE, 2009, pp. 697–704.

[11] F. Alimoglu and E. Alpaydin, "Methods of combining multiple classifiers based on different representations for pen-based handwritten digit recognition," in *Proceedings of the Fifth Turkish Artificial Intelligence and Artificial Neural Networks Symposium (TAINN 96*. Citeseer, 1996.

[12] D. Llorens, F. Prat, A. Marzal, J. M. Vilar, M. J. Castro, J.-C. Amengual, S. Barrachina, A. Castellanos, S. E. Boquera, J. Gómez *et al.*, "The ujipenchars database: a pen-based database of isolated handwritten characters." in *LREC*, 2008.

[13] M. Kudo, J. Toyama, and M. Shimbo, "Multidimensional curve classification using passing-through regions," *Pattern Recognition Letters*, vol. 20, no. 11, pp. 1103–1111, 1999.

[14] N. Hammami and M. Bedda, "Improved tree model for arabic speech recognition," in *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, vol. 5. IEEE, 2010, pp. 521–526.

[15] S. P. Chatzis and D. I. Kosmopoulos, "A variational bayesian methodology for hidden markov models utilizing student's-*t* mixtures," *Pattern Recognition*, vol. 44, no. 2, pp. 295–306, 2011.

[16] N. Hammami, M. Bedda, and N. Farah, "Spoken arabic digits recognition using mfcc based on gmm," in *Sustainable Utilization and Development in Engineering and Technology (STUDENT), 2012 IEEE Conference on*. IEEE, 2012, pp. 160–163.