Optimizing Hierarchical Temporal Memory for Multivariable Time Series

David Rozado*, Francisco B. Rodriguez, and Pablo Varona

Grupo de Neurocomputación Biológica (GNB)
Dpto. de Ingeniería Informática, Escuela Politécnica Superior,
Universidad Autónoma de Madrid. Calle Francisco Tomás y Valiente,
11, 28049 - Madrid, Spain
{david.rozado,F.Rodriguez,pablo.varona}@uam.es
http://www.ii.uam.es/~gnb

Abstract. Hierarchical Temporal Memory (HTM) is an emerging computational paradigm consisting of a hierarchically connected network of nodes. The hierarchy models a key design principle of neocortical organization. Nodes throughout the hierarchy encode information by means of clustering spatial instances within their receptive fields according to temporal proximity. Literature shows HTMs' robust performance on traditional machine learning tasks such as image recognition. Problems involving multi-variable time series where instances unfold over time with no complete spatial representation at any point in time have proven trickier for HTMs. We have extended the traditional HTMs' principles by means of a top node that stores and aligns sequences of input patterns representing the spatio-temporal structure of instances to be learned. This extended HTM network improves performance with respect to traditional HTMs in machine learning tasks whose input instances unfold over time.

Keywords: Sequence Encoding, Motion Analysis, Multidimensional Signal Processing, Neural Network Architecture, Pattern Recognition.

1 Introduction

Hierarchical Temporal Memory (HTM) [1], [2] is a conexionist paradigm with a novel set of bio-inspired assumptions encapsulating theories about neocortical function into a set of algorithms [3]. HTM theory incorporates the hierarchical organization of the mammalian neocortex into its topological architecture [4], [5]. HTM also uses spatio-temporal codification as a way to encapsulate and learn the structure and invariance of problems' spaces [6]. Spatio-temporal coding and hierarchical organization are well documented principles of information processing in living neural systems [7], [8].

* This work was supported by grants from 'Consejería de Educación de la Comunidad de Madrid' (C.A.M), the 'European Social Fund (E.S.F.)' and the 'Ministerio de Ciencia e Innovacion': MICINN BFU2009-08473, TIN 2007-65989 and CAM S-SEM-0255-2006

K. Diamantaras, W. Duch, L.S. Iliadis (Eds.): ICANN 2010, Part II, LNCS 6353, pp. 506–518, 2010.

HTM hypothesizes that time is used by the neocortex as a supervisory signal for clustering together spatially different input patterns that tend to present themselves close together in time. The usage of temporal integration minimizes storage requirements and reduces the need for supervised training.

The theoretical aspects of the HTM paradigm were throughly described in [2], [9]. An up to date version of the theory with a probabilistic model of temporal aggregation and feedback information flowing from parent nodes to children nodes to disambiguate noisy inputs can be found in [1]. HTM algorithms can be used to solve problems on different domains: pattern recognition, control theory or behavior generation among others [10]. In this paper we center our attention to HTMs applied within the realm of temporal pattern recognition.

The main objective of an HTM network trained for pattern recognition is the development of invariance capabilities [11]. That is, given a set of categories, each one of them represented by a set of instances, the system should learn to properly separate the category-space using a small subset of training instances from each category set. After training, the system should be able to generalize and properly assign the correct categories to unseen instances from the category space.

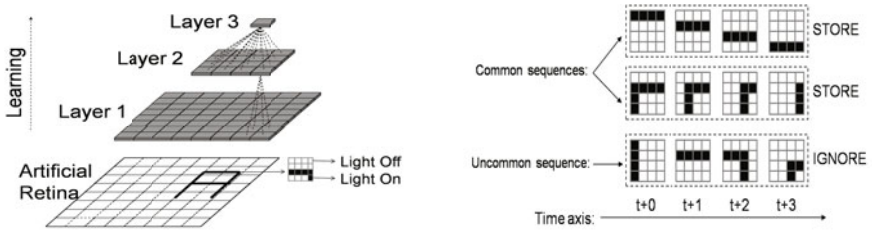
HTM algorithms perform robustly in traditional machine learning tasks such as image recognition [11]. Problems where HTM excel are those with an inherent spatio-temporal structure and whose instances are represented completely at any given time instant. For problems where an instance is composed of a time series of spatial arrangements, HTMs performance is not as robust.

In this paper we develop a feature for HTMs to perform better on learning tasks whose instances unfold over time: we modify the HTM's top node by enabling it to store sequences of spatio-temporal input patterns arriving at the top node over time. This top node also performs similarity measurements among incoming sequences in order to map unseen instances to known categories. The rationale for using sequences to map stimuli to categories has been justified in [12], [7], [13].

We illustrate the performance of our modified HTM system in the problem of sign language recognition. Sign language is used by deaf people to communicate by means of using sequences of hand movements instead of speech. Sign Language constitutes a good fit for the type of problem that we wanted to tackle: category spaces whose instances are composed of an ordered sequence of spatial arrangements. Therefore, we chose a data set containing time series of hand movements representing signs from Australian Sign Language (ASL) as a proof of principle that our expanded HTM system can perform well on problems whose instances develop over time.

2 HTM Formalism

HTM's network topology consists of a set of layers arranged in a hierarchy, Fig. 1(a). Each layer is composed of one or several computational nodes operating in discrete time steps. Nodes are related through children-parent relationships. Each node throughout the hierarchy possesses an intrinsic receptive field formed



(a) (b)

Fig. 1. HTM Basis. Panel *a* shows an HTM's topology containing 3 layers. Each layer is composed of one or several nodes. Bottom level nodes are fed with input from a sensor, in this case an "artificial retina". Figure adapted from [3]. Panel *b* shows instances of 2-Dimensional spatial coincidences from a small receptive field in an artificial retina. The spatial coincidences in the first two rows do not have a high pixel-to-pixel similarity but since they appear often in a sequence they are temporally adjacent, so an HTM node clusters them into the same temporal group. Non frequent sequences of spatial coincidences are not clustered together. Figure adapted from [2].

by its children nodes or, in the case of bottom level nodes, a portion of the sensors' input space. Nodes of HTM networks can receive many types of input vectors that specifically encapsulate the input space properties of the receptive field to which they are exposed. All nodes through-out the hierarchy, except the top node, carry out unsupervised learning.

Each node in an HTM network can be thought of as composed of two critical elements: a *spatial pooler* and a *temporal pooler*. These two identifiers serve as abstractions to refer to the form by which input vectors received during training are partitioned and stored into sets of temporally adjacent vectors. All nodes start off with their spatial and temporal poolers empty. Nodes undergo two modes of function during their lifetime: a training mode and an inference mode.

During the training mode, a node stores input vectors coming from children nodes that are different enough, according to a threshold, from the vectors already stored on its spatial pooler. These input vectors are also referred to in the HTM literature as *spatial coincidences* [11].

When the spatial pooler is full or a sufficiently large number of training iterations has been carried out, the node creates a time-adjacency matrix. Each row and each column of this matrix represents a spatial coincidence, and the intersection between a row and a column maintains a probability value indicating the likelihood of a spatial coincidence, represented by the row, transitioning into the spatial coincidence, represented by the column, in the next time step.

Training is completed when a time-adjacency matrix' modification coefficient falls bellow a certain threshold. The node initiates then a segmentation algorithm that uses the transition frequencies stored in the time adjacency matrix to cluster the vectors in the spatial pooler into groups according to how frequently they tend to follow each other in time. These temporally adjacent groups are stored in the temporal pooler of the node. The rationale for this is the fact that input

patterns with a large spatial distance might be closely related or have a common cause if they tend to follow each other in time repeatedly, Fig. 1(b).

After the node spatial pooler has been partitioned into temporal groups, the node switches its state from training mode towards inference mode. During inference mode, the node continues to receive input vectors, but it does not perform any more learning of new vectors, it just calculates to which temporal group the incoming input vector most likely belongs. This is done by calculating the similarity between the incoming input vector and the vectors stored in the temporal groups. The node then emits an output vector containing as many elements as temporal groups are stored on its temporal pooler. Each element of the output vector could indicate a probability value of that temporal group being active. In a simplified version of the system, all elements of the output vector can be binary with all elements of the vector being 0s except for the element representing the temporal group to which the node believes the actual input vector belongs, which contains a 1. This vector is propagated toward the node's parent node. Fig. 2(a) shows the stages a node goes through during its life time.

In traditional HTM, the network's top node functions in a slightly different fashion to the rest of the nodes in the hierarchy. The top node does receive input vectors from children nodes, but it does not perform temporal aggregation of the data and it does not emit output vectors. During training, a signal is given to the top node as a cue about the particular category to which the system is being exposed to at a particular time instant. The top node just maps incoming input vectors to the signaled category in a supervised fashion.

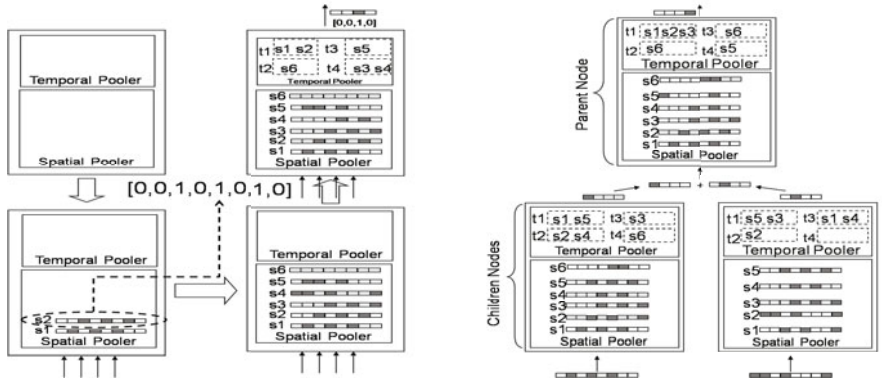


Fig. 2. Node inner-workings. Panel *a* shows how a node starts out with its spatial and temporal poolers empty. Input vectors formed by 1s and 0s (s1,s2. . . ,s6) received during training are stored in the spatial pooler. After training, the time adjacency matrix is used to determine the temporal groups of the node (t1,. . . ,t4). During inference mode, the node maps input vectors to temporal groups and emits the corresponding output vector. In Panel *b*, a parent node aggregates output vectors from children nodes to form its own input vector. These aggregated patterns from several children nodes constitute the spatial coincidences of a parent node's input space.

Training starts off on the bottom level layer and propagates throughout the hierarchy one layer at a time, with parent nodes transitioning into training mode right when their children nodes initiate their inference mode. First, only nodes at the bottom layer of the hierarchy are active and in training mode, the rest of the nodes in the hierarchy are silent. Once the bottom layer has completed its training mode, its nodes are switched to inference mode, and the nodes starts to emit output vectors, which conform the input data for the nodes higher up in the hierarchy. Parent nodes aggregate incoming vectors from several children nodes, Fig. 2(b), and use this aggregated pattern as their input vector.

When all the nodes in all the layers of the hierarchy have undergone their training mode, training for the whole network has been completed and the network is ready to perform inference.

3 Proposed Extension for the HTM Formalism

We have developed a version of the HTM theory as proposed in [2]. Message passing from children nodes to parent nodes nodes is implemented using binary vectors, containing just 1s and 0s in each element without feedback from parent nodes. This binary vector indicates which temporal group is active at a time. This extended HTM formalism has been developed in order to adjust the system to the specific needs of multivariable time-series problems whose instances develop over time.

The fundamental modification of our local HTM system with respect to traditional HTM networks is the modification of the network's top node whose task in original HTM algorithms is simply to map incoming vectors from children nodes to categories. The newly defined top node stores instead complete sequences of spatio-temporal input vectors in its *sequential pooler* and maps those sequences to categories.

Besides storing sequences of input vectors during training, the top node also performs similarity measurements among incoming sequences during both training and inference. The similarity calculation is needed in order to map unknown incoming sequences to the sequences already stored in the top node. Similarity measurements are carried out by sequence alignment using dynamic programming [14] as explained below.

We have tested our approach in the problem of Sign language recognition. Sign language recognition is fundamentally different from previously tried out problems within the HTM community, [11]. Most problems undertaken by HTMs, [10], consist of instances whose spatial configuration are fully represented at any time instant. Sign language is composed of sequences of spatial arrangements over time that together constitute a sign. At any given time t, the complete representation of the sign is not available, just a particular spatial arrangement of the hands. It is the particular temporal sequence of hand arrangements what constitutes a sign. The fundamentally different nature of this kind of problem and the poor performance of traditional HTM networks to deal with it justified the undertaking of modifications within the HTM inner-workings. Figure 3(a)

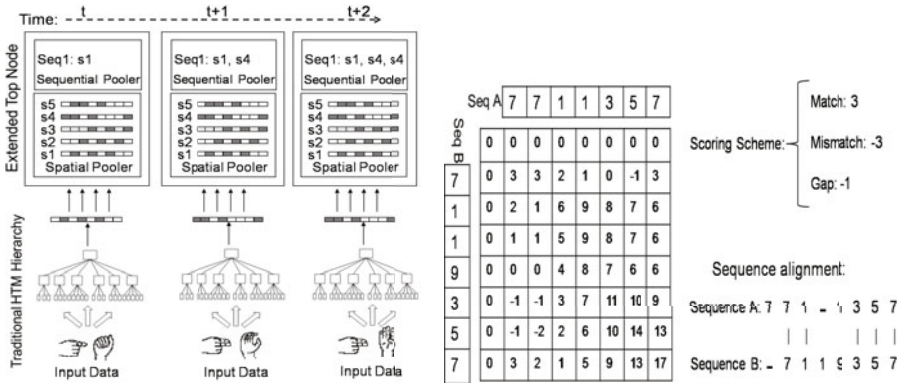


Fig. 3. Extended Top Node. Panel *a* shows how the top node stores sequences of spatial coincidences during training in its sequential pooler. First the node stores on its spatial pooler the set of spatial coincidences to which it is exposed to during training. Then it learns common sequences of those spatial coincidences. Panel *b* shows how the extended top node uses sequence alignment using dynamic programming to measure similarity among two sequences. The external row and column represent 2 sequences formed by spatial coincidences units. The score at the bottom right of the matrix indicates the top global alignment between both sequences. A scoring scheme was used to score matches, gaps and mismatches in the alignments.

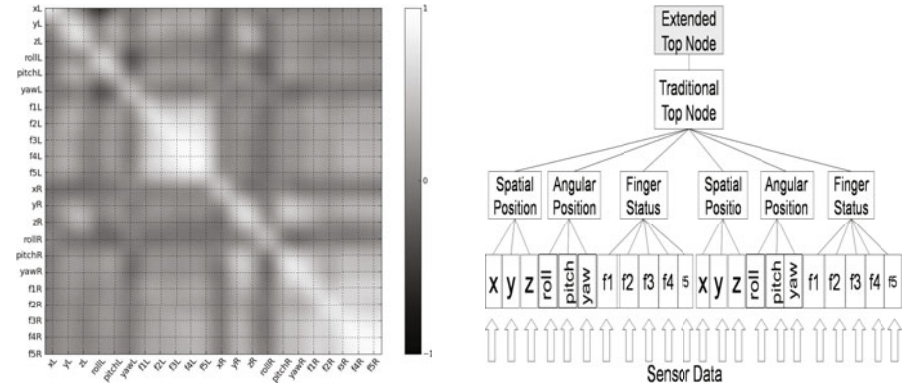
illustrates how the learning of a sign comes about over time in our modified top node by storing the sequence of spatial coincidences that follow each other in time during the "utterance" of the sign.

4 Example Application: Sign Language Recognition

We illustrate the extended HTM formalism on a data set consisting of several instances from sign language. The data set was obtained from [15] and it was captured using a pair of electronic data gloves[1] containing accelerometers and sensors to track 11 channels of information for each hand: x, y and z spatial coordinates of the hand, the roll, pitch and yaw rotation angles of the wrist and a bend coefficient for each finger. A particular configuration of all the channel variables at one instant in time is referred to in this paper as a frame.

When a user puts on this gloves and performs a particular sequence of hands movements representing a sign, the gloves provide dynamic data representing the spatio-temporal transitions of the tracked variables as they oscillate over time while the hands "utter" a particular sign. A total of 95 different sign-categories with 27 sign-samples per category were recorded from a single volunteer native signer using ASL [15].

[1] Data Gloves from 5DT Fifth Dimension Technologies (`http://www.5dt.com`).



(a) (b)

Fig. 4. Optimal Network Topology. Panel *a* shows a cross-correlation matrix of the 22 channels from the data set. This matrix was used to find the optimal topology of the network since highly correlated input channels should be grouped in the network's lower layers. All the fingers of each hand are highly correlated as wells as the x,y and z coordinates of each hand and the roll, pitch and yaw angles of the wrist. Panel *b* shows what was determined to be the optimal HTM network topology for the illustrated problem of sign recognition.

For each simulation, the data was automatically partitioned in a random fashion into two sets, 90% of the available data became the training set and the remaining 10% became the test set. The partitions were different for each simulation and were determined ad-hoc right before each simulation. This prevented over-fitting during the search for optimum HTM training parameters.

We created an HTM topology formed by 3 layers as shown in Fig. 4(b) that proved to be the one that optimized the performance of the network over several alternative network designs as explained below. The bottom layer nodes received their input data from the 11 channels of input data coming from each glove. Several topologies were tried out varying the number of layers, nodes, fan-in edges, fan-out edges, spatial-pooler specificity coefficients and clustering coefficients for the nodes.

The original data contained intrinsic noise, making it necessary to filter the data. For each channel an average value for each category was determined. Then, when any instance differed significantly from the category average, the difference was subtracted from the absolute values for that particular instance. This procedure visibly reduced noise deviations in the data set.

Since the original data contained continuous variables, the next filtering step was to discretize the continuous input values into discrete input values. That is, the continuous input space had to be converted into a discrete input space. For each channel, the range of possible values was segmented into a finite number of regions. Each region represented a particular discrete value of the input space. If the value of a continuous variable from an input channel fell into a particular region, the channel adopted the discrete value associated with that region.

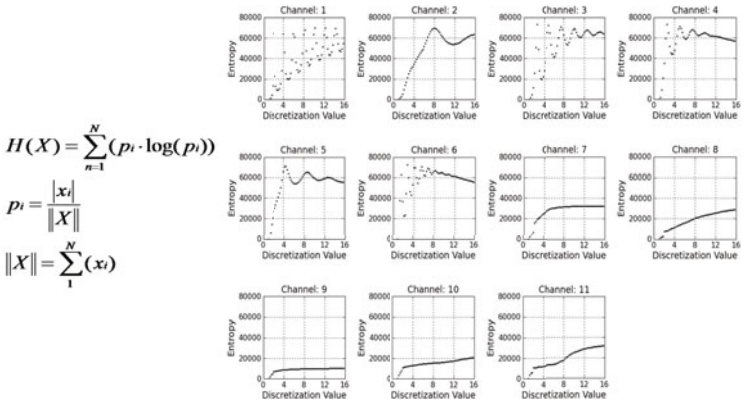


Fig. 5. Entropy analysis of left hand Determining the right partitioning values, ν, to transform the continuous input variable into ν discrete states. The optimum ν according to simulation trials coincided with low entropies when comparing different ν values' impact on performance. The entropy of a sequence of values in a channel $X = x_1, x_2, x_3, ..., x_N$ is defined by the formula in the figure with p_i being the probability of value i, $|x_i|$ being the number of times value x_i occurs and $|X|$ being the length of the sequence.

An entropy analysis for each channel was performed in order to determine the proper segmentation value for each channel, Fig. 5. That is, in how many discrete segments the continuous variable should be divided in order to minimize entropy. The optimal parameters suggested by the entropy analysis and confirmed by manual supervision settled down on segmentations values between 2-6 regions for each channel.

The input data's absolute x, y and z coordinates of the hands is sometimes not enough to completely describe a sign. Other variables such as first and second derivatives of the x, y, and z coordinates further describe a particular sequence of hand movements. Therefore, we used derived data from the absolute values provided by the data set and performed simulations using the derived velocity and acceleration information corresponding to the x, y and z variables.

Since our Extended HTM formalism consisted of a top node that stored sequences of spatio-temporal patterns, we needed some means to perform comparisons among stored and incoming sequences in order to determine similarity. The need for a measurement of similarity was two-fold: It was needed during training in order to determine which sequences to store and which ones to disregard, in case of high similarity to an already stored sequence. A similarity measurement was also needed during inference to determine which sequence, from the set of stored sequences in the sequential pooler of a top node had the highest degree of similarity to an incoming input sequence. We measured sequence similarity by performing sequence alignment using dynamic programming. Dynamic programming has been successfully used by the bioinformatics research community to calculate the degree of similarity between genetic sequences [14]. Dynamic Programming sequence alignment consists of using a scoring matrix to align two

sequences according to a scoring scheme by tracing down the optimal global alignment, Fig. 3(b).

Combinations of different HTM networks that exploit different data representations improves the performance of the algorithm. Therefore, a method was needed in order to carry out the combination of the results of several simulations. We settled down with a simple aggregated sum of results from different simulations as a way of pooling the results of several simulations into a combined result, Fig. 6(a). That is, each trained HTM network was tested by making it to perform inference over a set of unseen instances. For each inference over an unseen instance, a rank of sequences stored in the sequential pooler of the top node which were similar enough to the input instance was generated. Every sequence in the rank had a particular score and category associated with it. The score was the result of performing a global alignment between the input sequence and this specific sequence stored in the sequential pooler. Pooling the results of several simulations simply consisted of adding all the scores associated to a certain category in the similarity rank generated for each input instance, Fig. 6(b).

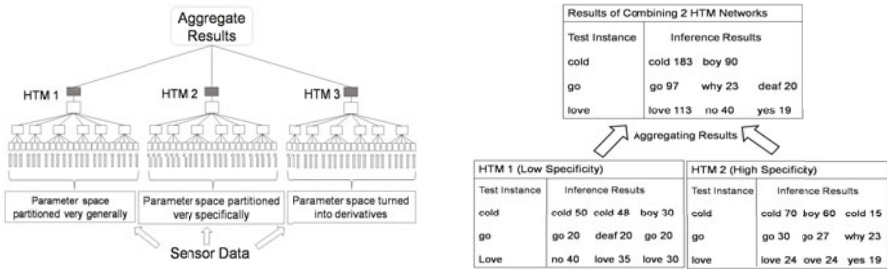


Fig. 6. Aggregation of Results. The results of HTM networks with different configuration parameters and fed with different data representations can be pooled to improve overall performance, Panel *a*. Panel *b* shows how the results tables of two different HTM network simulations can be combined into a single table by adding up the scores associated to each category in the similarity rank.

We measured the performance of the algorithms by testing what percentage of unseen signs' instances were assigned to their proper categories after training. Two types of measurements were used: Measurement A indicated the percentage of times the algorithm correctly first guessed an unseen utterance of a sign. Measurement B indicated the percentage of times the algorithm guessed the correct category of the unseen instances within its top three guesses.

Several simulations were carried out with different network parameters. The optimal topology used in the simulations was that of Fig. 4(b). The results of the simulations using a traditional HTM network were poor, just above 30% for Measure A, since they are not optimized to handle patterns unfolding over time. The addition of our own node with sequence storage and sequence alignment capabilities proved to be of critical importance with results jumping up to the

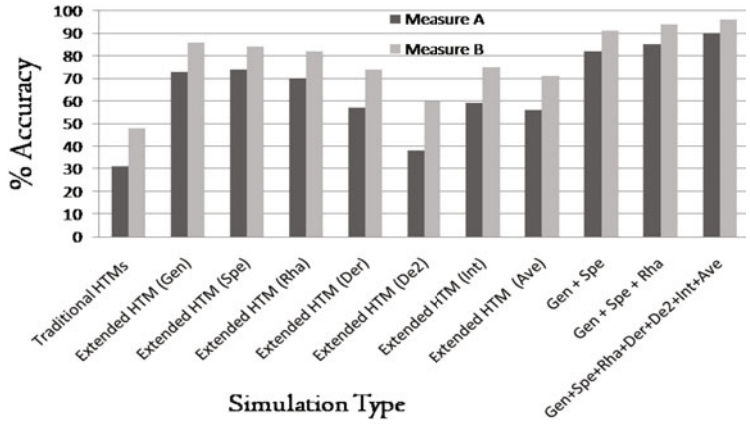


Fig. 7. Comparison of Simulations Performance under different arrangements. Measure A refers to the percentage of times the simulation correctly first guessed the appropriate category of an unseen instance. Measure B refers to the percentage of times the simulation correctly guessed the appropriate category of an unseen instance within its top 3 guesses. Aggregated results of several simulations are indicated with the + operator.

70-80% range for Measure A. Figure 7 provides a summary of results for different simulation types.

In Fig. 7, 'traditional HTMs' simulation type refers to HTMs as described in [2]. The rest of the simulations were carried out with the expanded top node as described previously in this paper. Several simulations with different data representations were carried out: Simulations optimized for specificity (Spe) or generalization (Gen), simulations using derived data from the original x, y and z coordinates: derivatives (Der), second derivatives (Der2), integrals (Int), averages (Ave) and simulations using data just from the right hand (Rha).

Simulations combining several single type simulations improve inference's performance. Combination of results from individual networks are referred to in Fig. 7 as addition of the specific tags associated with each simulation type.

5 Discussion

HTMs perform successful pattern recognition over data sets whose instances' spatial components are perceived completely at a particular time instant, for instance, image recognition [11]. That is, through one single flash-shot, the system perceives the whole spatial representation of an instance from a category.

Instances whose composition unfolds over time as a particular timed sequence of spatio-temporal configurations of input variables have not been shown in the literature to be as robustly recognised by existing HTMs networks [10]. Our data set consisted of precisely that: instances of sign language composed by an ordered sequence of frames.

Traditional HTM algorithms as described in the literature perform poorly on our selected dataset, Fig. 7. This comes about because of the specific nature of our tackled problem which is fundamentally different from image recognition. In our data set, each frame considered in isolation does not represent any particular sign just a flash-view spatial arrangement of the hands in space. Only the appropriate sequence of spatial arrangements over time uniquely constitutes a sign. The top node in traditional HTMs just tries to guess the proper category of an instance for every given frame during the performance of a sign. But a given time frame or arrangement of the 11 channels of information for each hand can be shared by several signs and hence the poor results.

To overcome the traditional HTMs shortcomings, our Extended HTM formalism creates a top node whose main task is to store sequences of spatio-temporal input vectors as incoming input instances unfold over time 3(a).

A cross correlation analysis was performed in order to find out correlations among input channels, Fig. 4(a). This information was used to design the optimal network topology, Fig. 4(b). HTMs work better if highly correlated channels are grouped locally in the lower levels of the hierarchy [2], leaving the upper layers to find out more complex, distant or not so obvious correlations. Accordingly our network topology grouped in its bottom layers those variables that were highly correlated according to Fig. 4(a).

A critical aspect for the performance of the algorithm is the degree of granularity used for the discretization of the continuous input space. The discretization process represents a trade-off between specificity and generalization capabilities of the network. Excessive granularization, that is, partitioning the continuous input space into too many regions, increases the specificity of the learning, but leads to over-fitting and less generalization abilities of the network. Obviously, the more instances available for training, the higher the degree of specificity that can be reached, but our data set was constrained to just 27 samples for each sign. On the other hand, a very unspecific partition of the input space, favors generalization capabilities but also decreases the specificity of the network, that is, the number of false positives increases. An entropy analysis was performed, Fig. 5, to find out the optimum degree of granularity needed to transform the continuous input space into a discrete space.

The types of errors committed for different simulations were different in terms of sensibility and specificity. Simulations optimized for generalization improved the sensibility with the cost of getting too many false positives. Simulations optimized for specificity were very accurate in terms of always getting true positives, yet they would miss several true positives due to their lack of generalization capabilities. Too much specificity also lead to an explosion in terms of storage and processing requirements.

The relatively good results of simulations using just information from the right hand (Rha) 7 are due to the fact that in Australian Sign Language most signs are perform just with the right hand while the left hand stays still.

Performing several simulations for different data representations of the data set and then using a pooling system to aggregate the results of different network

simulations improves overall performance, Fig. 7. This is due to the fact that some granularizations, or data representations, are optimal just for the recognition of some signs with no obvious optimal data representation for all signs. Therefore, combinations of HTM networks that exploit different data representation overcomes this limitation.

Our approach improves performance of HTMs significantly for recognition of multivariable time series. There is however a trade-off since the sequence alignment methodology used by our modified top node is NP-hard underlining the high computational costs of this approach.

The results of applying our method on the ASL Date Set, Table 7 are slightly worse than those obtained by cite [16] which achieves accuracy percentages of up to 96%. However, the method described by [16] is intrinsically of a highly supervised nature, since the features to be matched by the algorithm were previously defined by the author. The nature of our algorithm is fundamentally different since it is highly unsupervised in its feature learning methodologies and therefore highly adaptable to a wide array of problems.

Other authors have recently used HTMs for Polish sign language recognition [17]. These authors have used a video based recognition approach while we used data captured with a data glove. Although in [17] they get slightly better results for Polish Sign Language than our algorithm for ASL, around 94%, they also fed the HTM algorithm with additional channels about movement type, and visemes [17] something which our data set lacked [17]. Also their training set contained more instances for each sign, 40, as opposed to our data set which only contained 27.

In summary, our reformulation of the top node in an HTM system by providing it with sequence storage and alignment capabilities improves performance when used upon classifying signs from a data set containing Australians sign language data recorded using electronic gloves. This approach can be easily generalized for machine learning applications where the patterns to be learned also unfold over time.

References

1. George, D., Hawkins, J.: Towards a mathematical theory of cortical micro-circuits. PLoS Comput. Biol. 5(10), e1000532 (2009)
2. Hawkings, J.: Hierarchical temporal memory, concepts, theory, and terminology. Numenta, Tech. Rep. (2006)
3. George, D., Jarosy, B.: The HTM learning algorithms. Numenta, Tech. Rep. (2007)
4. Mountcastle, V.: The columnar organization of the neocortex. Brain 120(4), 701–722 (1997)
5. Douglas, R.J., Martin, K.A.: Neuronal circuits of the neocortex. Annual Review of Neuroscience 27(1), 419–451 (2004)
6. Dean, T.: Learning invariant features using inertial priors. Annals of Mathematics and Artificial Intelligence 47, 223–250 (2006)
7. Rabinovich, M.I., Varona, P., Selverston, A.I., Abarbanel, H.D.I.: Dynamical principles in neuroscience. Reviews of Modern Physics 78(4), 1213+ (2006)

8. Pöppel, E.: A hierarchical model of temporal perception. Trends in Cognitive Sciences 1(2), 56–61 (1997)
9. Hawkings, J.: On Intelligence. Cambridge University Press, Cambridge (1991)
10. Numenta: Problems that fit htm, Numenta, Tech. Rep. (2006)
11. George, D., Hawkins, J.: A hierarchical bayesian model of invariant pattern recognition in the visual cortex. In: Proceedings of 2005 IEEE International Joint Conference on Neural Networks, IJCNN 2005, July 4-August, vol. 3, pp. 1812–1817 (2005)
12. Abeles, M.: Corticonics: neural circuits of the cerebral cortex. Henry Holt and Company (2004)
13. Rodríguez, F.B., Huerta, R.: Analysis of perfect mappings of the stimuli through neural temporal sequences. Neural Netw. 17(7), 963–973 (2004)
14. Giegerich, R.: A systematic approach to dynamic programming in bioinformatics. Bioinformatics 16(8), 665–677 (2000)
15. Kadous, M.W.: Australian sign language signs data set, UCI Machine Learning Repository, Tech. Rep., `http://archive.ics.uci.edu/ml/datasets/`
16. Kadous, M.W.: Temporal classification: Extending the classification paradigm to multivariate time series. Ph.D. dissertation, The University of New South Wales (2002)
17. Tomasz Kapuscinski, M.W.: Computer Recognition Systems 3, vol. 57, pp. 355–362. Springer, Heidelberg (2009)