Video Sequence Learning and Recognition via Dynamic SOM

Qiong Liu, Yong Rui, Thomas Huang, Stephen Levinson
Beckman Institute for Advance Science and Technology
University of Illinois at Urbana-Champaign
405 North Mathews, Urbana, IL 61801, U.S.A.
Email:{q-liu2, yrui, huang, sel}@ifp.uiuc.edu

ABSTRACT

Information contained in the video sequences is crucial for an autonomous robot or a computer to learn and respond to its surrounding environment. In the past, robot vision is mainly concentrated on still image processing and small "image cube" processing [1]. Continuous video sequence learning and recognition is rarely addressed in the literature due to its high requirement on dynamic processing. In this paper, we propose a novel neural network structure called Dynamic Self-Organizing Map (DSOM) for video sequence processing. The proposed technique has been tested on real data sets, and the results validate its learning /recognition ability.

1. INTRODUCTION

Video sequences contain visual information that a robot/computer needs to see the world. Current computer vision research has been mainly focused on static analysis of still images, or simple derivatives on video sequences [1][2][3][4]. How to process dynamic video sequences more naturally is still an open problem in research.

A video sequence is a typical spatio-temporal signal. In general, its processing techniques can be separated into two categories, static processing and dynamic processing, according to their different time treatment strategy.

Static video processing (SVP) is a commonly used and successful technique [2][3][4]. It processes video based on short video segments. This method is straightforward and easy to be analyzed, but it requires buffering the whole segment, and costs big memory space.

Dynamic video processing (DVP) handles a video sequence based on its current inputs and past states, but it does not need to save many past inputs. Compared with SVP, DVP is not so straightforward. However it costs less memory space than SVP in general, and it is more similar to human's processing strategy.

The continuity property of a video sequence makes it difficult to cut out video segments for static comparison/recognition. Moreover, this property makes it difficult to save all data for future reference. Due to the above difficulties, it seems to be more appropriate to use DVP instead of SVP for video sequence learning/recognition.

Next, we will present our basic ideas and experiments in details. We hope this presentation can inspire useful discussions in this research area. In section 2, we introduce some basic ideas of Kohonen Map and dynamic neural network. Because we will use knowledge from these networks to construct DSOM, this section is mainly for those readers who are not familiar with these structures. Section 3 gives the detailed construction procedure of DSOM and training equations. It includes the most important contents of this paper. Section 4 is the current simulation result of DSOM. It uses the DSOM's classification map and classification result to lead the readers to a more concrete comprehension of this technique. Section 5 concludes this paper and discusses some future work.

2. RELATED WORK

In the neural network society, Kohonen Map and dynamic neural network are two popular network structures for learning. Both of them have their own big advantages for solving specific problems, and have their own big disadvantages for processing large set of dynamic data.

2.1. Kohonen Map

The Kohonen Map (or Self-Organizing Map) is an algorithm for visualizing and interpreting large high-dimensional data sets [5]. The map consists of a regular grid of processing nodes. A vector of features through high-dimensional observation is associated with every processing node. The algorithm tries to use the restricted vector map to represent all available observations with an optimal accuracy. For optimal representation of all available observations, the vectors are ordered on the map so that similar vectors are close to each other and dissimilar vectors are far from each other [5]. This representation strategy is analogous to the map representation in human's brain cortex, and proved to be effective in many applications [6].

The search and organization of the representation vectors on the map can be described with the following regressive equation, where t =1, 2, 3, … is the step index, x is a observation, $m_i(t)$ is the vector representation on node i at step t, c is the winner index, and $h_{c(x),i}$ is the neighborhood updating function [5].

$$\forall i, \| x - m_c(t) \| \le \| x - m_i(t) \| \quad (1)$$

$$\vec{m}_i(t+1) = \vec{m}_i(t) + h_{c(x),i} \cdot (\vec{x} - \vec{m}_i(t)) \quad (2)$$

After iterative training process, a SOM will become ordered. An ordered SOM can be used as a classifier, but the classification accuracy is not very high in many applications. To increase the classification accuracy, a nodal vector refinement stage called Learning Vector Quantization (LVQ) must be performed on the ordered SOM [1]. At this refinement stage, an input vector is picked at random from the input space. If the class label of the input vector and its SOM representation vector label agree, the representation vector will move toward the direction of the input vector. Otherwise, the representation vector will move away from the input vector. LVQ is a supervised learning process. It can help the ordered SOM to refine its classification boundary and improve the classification accuracy. It has nothing to do with the map topology formation process.

2.2. Dynamic Neural Network

In the traditional McCulloch-Pitts neuron model or Rosenblatt's perceptron, every synapse is considered as a weight [7]. With this model, the perceptron output is only related to the current input. That makes it impossible for us to use this simple model to capture time variance of the signal. In the real world, a synapse works in a more complicated manner. It has resistance, capacitance, transmission chemicals etc. [7]. Equipped with these components, a synapse could capture the time variance of an input signal. To simulate a synapse in a more accurate way, scientists have developed many dynamic models in the past decades [8][9][10]. Among all these proposed models, the neural filter model [11] (Fig.1.) is a promising one for presentation and application [12].

In Fig.1, the whole diagram can be viewed as an individual neuron. All synapses in this neuron model are represented by linear filters. The inputs of these filters are connected to the input signals of the neuron. The outputs of these filters are connected to the summation node of the neuron as usual. The summation node and the squashing function block are still similar to their representations in most other neuron models. The linear digital filters for every synapse can be IIR filters or FIR filters [13]. The parameters of these filters can be trained through the popular Back-Propagation training algorithm.

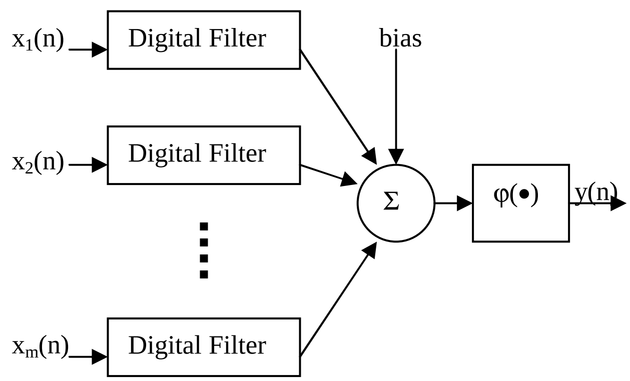

Fig.1. Filter model of an individual neuron

3. DYNAMIC SOM CONSTRUCTION

From the above descriptions, we know that Kohonen Map is good for processing large high-dimensional data set [5], and dynamic neural network is good for time sequence pattern recognition. Both of these networks have their big advantages. But their limitations cannot permit us to use them separately for dynamic video learning and recognition. The limitation of Kohonen Map for time sequence processing is caused by the static feature vector representation in the map model. The limitation of dynamic neural network is caused by its dramatic connection increase for large data set. In the following, we will propose a construction method to combine these two networks and inherit the advantages from both of them without losing generality.

The construction of DSOM is to substitute every static-processing unit in the SOM with a single output dynamic neural network. With this substitution, every SOM node will be able to deal with time sequences instead of static vectors. At the same time, we got a huge number of dynamic processing units for dealing with large sets of time sequences.

The DSOM map structure is sketched in Fig.2. In this construction, every synapse has three adjustable parameters, the input connection, the hidden node feedback connection, and the hidden node output connection. The node on the map grid functions as the summation node and the squashing block. The arrow that leaves the summation node is an output. The map grid has the same function of the traditional SOM map grid, and the neighborhood definition is also similar to the original SOM.

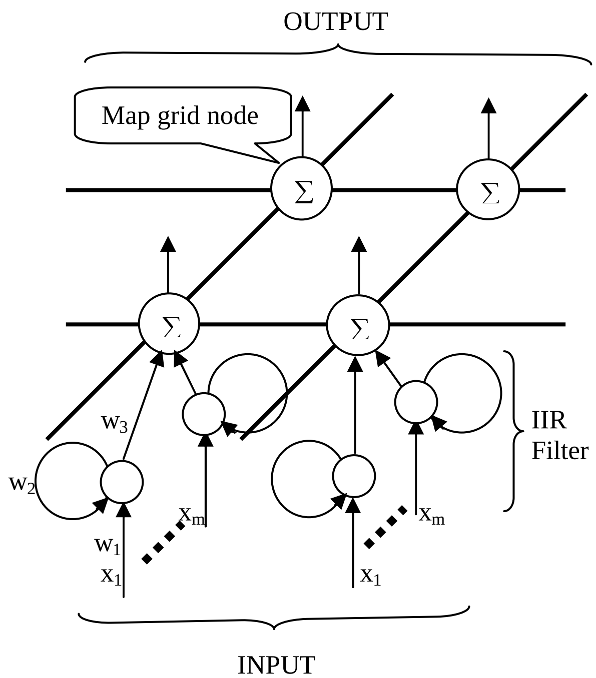

Fig.2. DSOM Diagram

The formation of DSOM can be separated into three distinctive stages. These stages are the competitive stage, the learning stage, and the re-labeling stage. In the competitive stage, the input time sequence is tried on all map nodes. With a vector sequence input, every node will output a one-dimensional sequence as the response. All output sequences of the map nodes will be integrated separately. The node which has the highest output integral in all map nodes will be chosen as the winner of the map. The node which has the lowest output integral in all map nodes will be chosen as the loser of the map. For the simple construction we present here, the competitive process can be described with the following equations:

$$\vec{s}(n)=\vec{w}_1\cdot\vec{x}(n)+\vec{w}_2\cdot\vec{s}(n-1) \tag{3}$$

$$v(n)=\sum_{i=1}^{m} w_{3i}\cdot s_i(n) \tag{4}$$

$$y(n)=\varphi(v(n)) \tag{5}$$

$$I=\sum_{n=0}^{l} y(n) \tag{6}$$

where x is the input, s is the output of the digital filter, w_1 is the connection weight from the input to the hidden node, w_2 is the hidden node feedback connection weight, w_3 is the connection weight from the filter output to the summation node, v is the summation of the digital filter outputs, y is the squashed value of v, I is the integral value of y in the time domain, n is the time step, l is the total length of the input sequence, m is the dimension of the input vector at one time step. w_1, w_2, w_3 are different for different connections. The relation of these variables can be viewed in Fig.3.

The learning stage follows the competitive stage. At this stage, the winner's label[1] will be compared with the input sequence label. If the two class labels agree, the desired nodal output sequence[2] will be set to constant 1 for all map nodes. Then the map nodes will be trained with back-propagation algorithm based on the input sequence and the desired output sequence. The learning rates for different map nodes are different. The winner node will have the highest learning rate in all map nodes. Other map nodes will have lower learning rates based on their distance from the winner. The farther the distance to the winner, the lower the learning rate. Learning rate change is controlled by a neighborhood function. The width of the neighborhood function will shrink as the map training iteration number grows. Usage of this neighborhood function is very similar to that in the ordinary SOM approach. The learning process is controlled by equations 7-13.

$$\gamma(t)=\eta(t)\cdot h(t) \tag{7}$$

$$\Delta\vec{w}_3(t+1)=\alpha\cdot\Delta\vec{w}_3(t)+\gamma(t)\cdot\delta_o(t)\cdot\vec{s}(t) \tag{8}$$

$$\vec{w}_3(t+1)=\vec{w}_3(t)+\Delta\vec{w}_3(t+1) \tag{9}$$

$$\Delta\vec{w}_2(t+1)=\alpha\cdot\Delta\vec{w}_2(t)+\gamma(t)\cdot\delta_h(t)\cdot\vec{s}(t) \tag{10}$$

$$\vec{w}_2(t+1)=\vec{w}_2(t)+\Delta\vec{w}_2(t+1) \tag{11}$$

$$\Delta\vec{w}_1(t+1)=\alpha\cdot\Delta\vec{w}_1(t)+\gamma(t)\cdot\delta_h(t)\cdot\vec{x}(t) \tag{12}$$

$$\vec{w}_1(t+1)=\vec{w}_1(t)+\Delta\vec{w}_1(t+1) \tag{13}$$

In equations 7-13, γ is a combined learning rate, t is the time, η is the time varying learning rate, h is the neighborhood function, α is the learning moment constant, δ_o is the local gradient at the neuron output, δ_h is the local gradient at a hidden node of the neuron, other variables are defined in equations 3-6. $\eta(t)$ is a positive real value. It decreases gradually as the time passes. "$h(t)$" is a neighborhood function, whose width will decrease as the time increases.

---

[1] Input sequence label is its class name. Each map node is also labeled with a class name or "unused". See the following re-labeling stage for detail.

[2] Each node is a dynamic neuron model. It outputs a time sequence for every time sequence input. The expected/desired time sequence output can be set to constant -1 or 1 sequence. We back-propagate the difference between the real output and the expected output for neuron model training on every map node.

After we train the map based on the winner's neighborhood, we train the map based on the loser's neighborhood. The training process follows the same equations as the winner-centered training. When the input label and the loser label agree, set the desired output to 1. Otherwise, set the desired output to –1. The reason for us to add the loser centered training is to balance the dynamic neuron training. If we use winner-centered training only, we can also get an organized map, but it is possible to get a biased output on every processing node.

The re-labeling step is very simple. Just pass all training data through the trained map, and count firings of each class on every node based on the competitive equations. A node is labeled by the class, which has the biggest firing count on the node. Nodes without firing will be labeled as "unused"[3]. After the re-labeling stage, the training process goes back to the competitive stage and learning stage again. The three-stage process is done iteratively till the map organization is stable.

Careful readers may find that this training algorithm is somehow similar to the LVQ learning in the SOM technique. The difference between LVQ learning algorithm and our algorithm is that LVQ uses a fixed label map in its class refining process, but we use a changing label map in our organizing stage. When the neighborhood function shrinks to a very small region, our algorithm will fix the label map for increasing the training speed[4].

The detailed training process of the DSOM can be described as follows:

1. Label all map nodes as "unused".
2. Set all adjustable connection weights with small random numbers.
3. Input a temporal sequence segment to every node on the map. It does not matter how long the sequence is if it is in a reasonable range. We have sequence length from about 40 to 100.
4. Record the output integral of every map node.
5. Find the winner node, which has the highest output integral among all nodes.
6. Find the loser node, which has the lowest output integral among all nodes.
7. Compare the sequence label with the winner/loser node label. If the winner is "unused", set its expected output to 1. If the loser is "unused", do nothing to the map. If the sequence label is the same as the node label, set the expected neuron output to constant 1. Otherwise, set the expected neuron output to constant –1. This value is the expected output value for every single step.
8. Use the back-propagation algorithm to train the network weights according to the input sequence and the expected output value. The training step size is set high for the winner/loser, and gradually decreases for remote nodes. The step size decrease follows the ordinary SOM neighborhood function. The winner-centered training and the loser-centered training are separated.
9. Pass all training sequences through the map, and re-label the map according to firing count on every node. Nodes without firing will be marked as "unused".
10. Go through step3 to step9 till the map is stable.
11. Fix the label map, and do step 5, 7, 8 without considering the loser and unused nodes till the map is stable.

4. SIMULATION RESULTS

For testing the learning ability of DSOM, we tried this model with some real video sequences. These video sequences are generated with mouse click on the screen. The paths of these video sequences reflect the writing procedure of 10 digits (0-9). The test/training data set has 10 classes. Each class has 10-16 noisy samples for training and testing respectively.

Before we process the digit sequences with DSOM, all of them are normalized to 9 by 9 frame size. We then project these 9 by 9 frame horizontally and vertically to produce an 18-element feature vector. Let p and q be the indices of horizontal and vertical axes. If the writing path passes point (p,q) at time t, the input vector at time t will be [x(0), …, x(i), …, x(17)] with x(p) =1, and x(9+q)=1, and x(i)=-1 for $i \neq p$ or $i \neq 9+q$. Interested readers can also try other features as the network inputs.

A 4x4 DSOM for the classification task is constructed with every processing unit having 18 inputs, 18 hidden units, and 1 output. The digital filters we used in every processing node are simple first order IIR filters. The neighborhood function we used for training is a 2D Gaussian function. Training/testing procedure is just as we described before. Fig.4. gives a typical training result (label map) of our experiment.

The numbers in the label map are class label number (0-9 correspond to digits 0-9. "x" corresponds to "unused".). At the beginning of the training process, most sequences keep on firing on a small number of neurons. As training iteration number goes up, sequence firings spread across the map, and gradually concentrate on their own centers. We can clearly notice the class grouping effect on the map after 6000 iterations of training. The correct classification rate with our examples was around

[3] How to update "unused" nodes will be explained in detail in the following step by step algorithm.

[4] Because the updating rates for neighborhood nodes become very small as the neighborhood function shrinks to a very small region, it is reasonable to omit the neighborhood nodes' updating. After our algorithm fixes the label map, it only needs to update one node for every input instead of updating all map nodes for every input. That can greatly increase the training speed.

60% to 70%. In a detailed firing distribution map, we can find that some 9s are mis-classified as 1 and vise versa, 6 and 0 are also difficult to be separated clearly. Considering that we only use a very simple neuron model on every node, this is already an amazing result.

```
0 x 8 1
x 8 x 8
0 x x 0
0 x x 2
```

(a)

```
2 5 4 1
5 3 4 9
7 5 8 6
7 0 0 x
```

(b)

Fig.4. A 4 by 4 DSOM label map during the training process. (a) is the label map after the first 30 training iterations. (b) is the label map after 6000 training iterations.

5. CONCLUDING REMARKS & FUTURE WORK

In this paper, we proposed an artificial neural network structure called DSOM for video sequence learning/recognition. The training approach of DSOM is a supervised learning algorithm. In this algorithm, we constructed a temporary label map to supervise the model learning process. The label map is updated through the training process. DSOM overcomes the limitations of traditional SOM and single dynamic neural network, and inherits the advantages from both of them. Simple video sequence classification results convinced us some basic ideas of the DSOM for processing large set of dynamic sequences.

In the video sequence learning/recognition experiment, we did not use any specific model to help the training. This supports some belief that this system has potential advantages over many model based recognition systems. This algorithm also has potential to overcome the difficulties of a time alignment process in most temporal sequence recognition systems, and speed up the temporal sequence recognition. The ordered DSOM is also more similar to the human brain cortex map than traditional SOM. It is used on video sequence data in this paper, but we believe that this method can also be generalized to some other temporal sequence learning and recognition tasks.

The future work of this research will include trying more elaborate dynamic neuron models in the DSOM construction, and trying more efficient visual feature as the inputs of the network. We also want to spend some time testing this method on traditional SOM formation and compare the speed and formation result of the new algorithm with the traditional algorithm.

6. REFERENCES

[1] B. Jahne, *Digital Image Processing (1995)*, Springer-Verlag.

[2] C. Tomasi, T. Kanade, "Shape and motion from Image Streams under Orthography: A Factorization Method", *International Journal of Computer Vision*, 9:2, 137-154 (1992).

[3] ISO/IEC, JTC1/SC29/WG11, *Description of core experiments on coding efficiency in MPEG-4 video*, Sept. 1996.

[4] T. Maurer, C. Malsburg, "Tracking and Learning Graphs on Image Sequences of Faces", *Int. Conf. on Artificial Neural Networks*, Bochum, July 16-19, 1996.

[5] T. Kohonen, "The Self-Organizing Map (SOM)", Web page available at http://www.cis.hut.fi/nnrc/som.html.

[6] T. Kohonen, *Self-Organizing Maps (1995)*, Springer.

[7] Simon Haykin, *Neural Networks, A Comprehensive Foundation, 2nd Edition*, 1999, Prentice Hall.

[8] J.L.Elman, "Finding Structure in Time", *Cognitive Science*, vol.14, pp.179-211, 1990.

[9] M.C.Mozer, "Neural Net Architectures for Temporal Sequence Processing", in A.S.Weigend and N.A.Gershenfeld, eds., *Time Series Prediction: Forecasting the Future and Understanding the Past*, pp. 243-264, Reading, MA: Addison-Wesley.

[10] J.J.Hopfield, "Neurons, Dynamics and Computation", *Physics Today*, vol. 47, pp.40-46, Februry, 1994.

[11] S.Haykin, B.Van Veen, *Signals and Systems*, New York : Wiley, 1998.

[12] M.Hagiwara, "Theoretical Derivation of momentum term in back-propagation", *International Joint Conference on Neural Networks*, vol. I, pp.682-686, Baltimore.

[13] S.J.Orphanidis, *Introduction to Signal Processing*, Prentice-Hall, 1996.