

# In-Place Learning for Positional and Scale Invariance

Juyang Weng, Hong Lu, Tianyu Luwang, and Xiangyang Xue

**Abstract**—In-place learning is a biologically inspired concept, meaning that the computational network is responsible for its own learning. With in-place learning, there is no need for a separate learning network. We present in this paper a multiple-layer in-place learning network (MILN) for learning positional and scale invariance. The network enables both unsupervised and supervised learning to occur concurrently. When supervision is available (e.g., from the environment during autonomous development), the network performs supervised learning through its multiple layers. When supervision is not available, the network practices while using its own practice motor signal as self-supervision (i.e., unsupervised per classical definition). We present principles based on which MILN automatically develops positional and scale invariant neurons in different layers. From sequentially sensed video streams, the proposed in-place learning algorithm develops a hierarchy of network representations. The global invariance was achieved through multi-layer quasi-invariances, with increasing invariance from early layers to the later layers. Experimental results are presented to show the effects of the principles.

## I. INTRODUCTION

Existing study in neural science has provided much knowledge about biological multi-layer networks. However, they have also raised several important open issues:

**1. Innate orientated edge features?** Orientation selective cells in cortical area V1 are well known [11], [29], [12], [6], but the underlying computational principles that guide their emergence (i.e., development) are still elusive. Are cells in V1 totally genetically wired to detect orientations of visual stimuli? This issue is important since the later layers of network, with their larger receptive fields, need to detect features that are more complex than oriented edges, such as edge groupings (T, X, Y, cross, etc), patterns, object parts, human faces, etc. A classical study by Blakemore & Cooper 1970 [4] reported that if kittens were raised in an environment with only vertical edges, only neurons that respond to vertical or nearly vertical edges were found in the primary visual cortex (also see, more recent studies, e.g., [28] and its review).

**2. Developmental mechanisms.** If the answer to the above is negative, what are the possible mechanisms that lead to the emergence of the orientation cells in V1? Since V1 takes input from the retina, LGN, and other cortical areas, the issue points to the developmental mechanisms for the early multi-layer pathway of visual processing. In this project, we propose intra-layer developmental mechanisms as well as inter-layer developmental mechanisms. The within-layer mechanisms are based on two well-known biological

mechanisms, *Hebbian learning* and *lateral inhibition*, but we further its optimality. The inter-layer developmental mechanisms specify how information between layers are passed and used, which enables both unsupervised and supervised learning to occur in the same network. Of course there are many other mechanisms that we do not examine in this work, such as internally generated spontaneous signals which assist the development of cortical networks before birth [26].

**3. In-place development.** By in-place development, we mean that the signal processing network itself deals with its own development through its own internal physiological mechanisms and interactions with other corrected networks and, thus, there is no need to have an extra network that accomplishes the leaning (adaptation). It is desirable that a developmental system uses an in-place developmental program, due to its simplicity and biological plausibility. We further propose that biological in-place learning mechanisms can facilitate our understanding of biological systems.

**4. Integration of unsupervised and supervised learning.** There exist many learning algorithms for self-organizing a multi-layer network that corresponds to a network of feature detectors. Well known unsupervised learning algorithms include Self-Organizing Map (SOM) by Kohonen [19], vector quantization [1], the Principal Component Analysis (PCA) [17], Independent Component Analysis (ICA) [14], [16], Isomap [30], and Non-negative Matrix Factorization (NMF) [21]. Only a few of these algorithms have been expressed by in-place versions (e.g., SOM and PCA [35]). Supervised learning networks include feed-forward networks with back-propagation learning, radial-basis functions with iterative model fitting (based on gradient or similar principles), Cascade-Correlation Learning Architecture [7], support vector machines (SVM) [5], and Hierarchical Discriminant Regression (HDR) [13]. However, it is not convincing that biological networks used two different types of networks for unsupervised and supervised learning, which may occur concurrently in the process of development. Currently there is a lack of biologically inspired networks that integrate these two different learning modes using a single learning algorithm.

**5. Invariant representation.** Hubel and Wiesel [11], [10] suggested early that a logic-OR type of learning for edges presented at different positions in the receptive field might partially explain how positional invariance may arise in V1. Studies have shown that a normally developed human visual system has some capabilities, but are not perfect, in translational invariance (see, e.g., [24]), scale invariance (see, e.g., [20]), or orientational invariance (see, e.g., [31]). Therefore, it seems biologically incorrect, even though convenient to engineering, to impose strict invariance in position, scale, or

Juyang Weng is with Michigan State University, East Lansing, MI 48824, USA (contact phone & fax: 517-353-4388; email: weng@cse.msu.edu).

Hong Lu, Tianyu Luwang, and Xiangyang Xue are with the Shanghai Key Laboratory of Intelligent Information Processing, Department of Computer Science and Engineering, Fudan University, Shanghai, China.

orientation. From the developmental point of view, such an imposition will significantly restrict the system’s ability to learn other perceptual skills. For example, when a square is rotated by 45 degrees, the shape is called a diamond; and the number 6 rotated by 180 degrees is called 9.

Some networks have built-in (programmed-in) invariance, either spatial, temporal or some other signal properties. Neocognitron by Fukushima 1980 [9] is a self-organizing multi-layered neural network of pattern recognition unaffected by *shift in position*. Cresceptron by Weng et al. 1997 [34] has an architecture framework similar to Fukushima’s Neocognitron while its neural layers are dynamically generated from sensing experience and, thus, the architecture is a *function of sensory signals*. The above two networks have built-in shift-invariance in that weights are copied across neurons centered at different retinal positions. However, other types of spatial invariance, such as size, style, and lighting, have to be learned object-by-object (no sub-part invariance sharing). Detection of objects using motion information belongs to built-in invariance based on temporal information. Many other built-in signal properties have been used to develop quasi-invariant feature detectors, such as color (e.g., human skin tones), saliency, slow in response (e.g., slow feature analysis [27]) and the independence between norms of projections on linear subspaces [15].

Other networks do not have built-in invariance. The required invariance is learned object-by-object. The self-organizing maps (SOM) proposed by Teuvo Kohonen and many variants [19] belong to this category. The Self-organizing Hierarchical Mapping by Zhang & Weng 2002 [36] was motivated by *representation completeness* using incremental principle component analysis and showed that the neural coding can reconstruct the original signal to a large degree. Miikkulainen et al. 2005 [23] have developed a multi-layer network with nearby *excitatory interactions* surrounded by *inhibitory interactions*. There have been many other studies on computational modeling of retinotopic networks<sup>1</sup> (e.g., von der Malsburg 1973 [32]; Field 1994 [8]; and Obermayer et al. 1990 [25]). While this type of network has a superior discrimination power, the number of samples needed to reach desired invariance is very large.

The work reported here proposes a new, general-purpose, multi-layer network, which learns invariance from experience. The network is biologically inspired (e.g., in-place learning), but is not necessarily biologically fully provable at the current stage of knowledge. The network has multiple layers, later layers take the response from early layers as their input. Although it is not necessarily that later layers only take input from its previous layer, we use this restriction in our experiment to facilitate understanding. The network enables supervision from two types of projections: (a) supervision from the next layer, which also receive supervision recursively from the output motor layer (the last layer for motor output); (b) supervision from other cortical regions (e.g.,

motor layer) as attention selection signals. The network is self-organized with unsupervised signals (input data) from bottom up and supervised signals (motor signals, attention selection, etc.) from top down. Every layer has the potential to develop invariance, with a generally increased invariance at later layers, with support of lesser invariance from early layers.

In what follows, we first present the network structure in Section III. Then, in Section III, we explain the in-place learning mechanism within each layer. Section IV explains how invariance is learned by the same in-place learning mechanism while supervision signals are incorporated. Experimental examples that demonstrate the effects of the discussed principles are presented in Section V. Section VI provides some concluding remarks.

## II. THE IN-PLACE LEARNING NETWORK

This section presents the architecture of the new Multi-layer In-place Learning Networks (MILN).

The network takes a vector as input (a set of receptors). For vision, the input vector corresponds to a retinal image. For example, the input at time  $t$  can be an image frame at time  $t$ , or the few last consecutive frames before time  $t$ , of images in a video stream. Biologically, each component in the input vector corresponds to the firing rate of the receptor (rod or cones) or the intensity of a pixel.

The output of the network corresponds to motor signals. For example, each node in the output layer corresponds to the control signal (e.g., voltage) of a motor. Mathematically, this kind of network that produces numerical vector outputs is called regressor. This network can also perform classification. For example, the number of nodes in the output layer is equal to the number of classes. Each neuron in the output layer corresponds to a different class. At each time instant  $t$ , the neuron in the output layer that has the highest output corresponds to the class that has been classified.

The architecture of the multi-layer network is shown in Fig. 1. For biological plausibility, assume that the signals through the lines are *non-negative* signals that indicate the firing rate. Two types of synaptic connections are possible, excitatory and inhibitory.

This is a recurrent network. The output from each layer is not only used as input for the next layer, but is also feedback into other neurons in the same layer through lateral inhibition (dashed lines in the figure). For each neuron  $i$ , at layer  $l$ , there are three types of weights:

- 1) bottom-up (excitatory) weight vector  $w_b$  that links input lines from the previous layer  $l - 1$  to this neuron;
- 2) lateral (inhibitory) weight  $w_h$  that links other neurons in the same layer to this neuron.
- 3) top-down (excitatory or inhibitory) weight  $w_t$ . It consists of two parts: (a) the part that links the output from the neurons in the next layer  $l + 1$  to this neuron. (b) The part that links the output of other layer processing areas (e.g., other sensing modality) or layers (e.g., the motor layer) to this neuron  $i$ . For notational simplicity,

<sup>1</sup>Each neuron in the network corresponds to a location in the receptor surface.

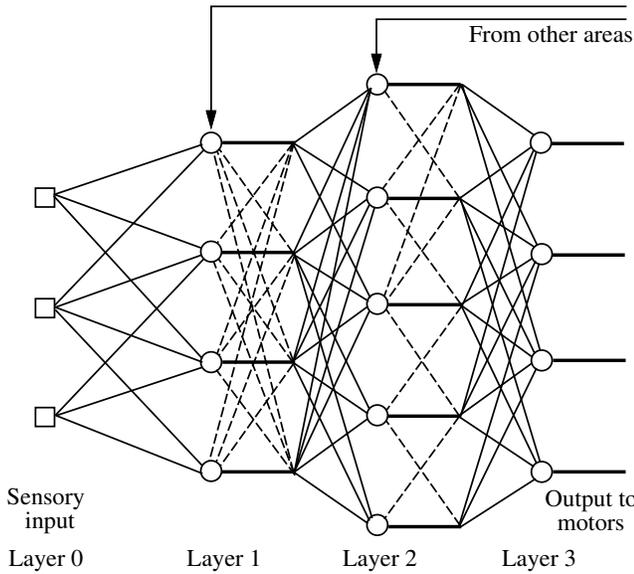


Fig. 1. The architecture of the Multi-layer In-place Learning Networks. A circle indicates a cell (neuron). The thick segment from each cell indicates its axon. The connection between a solid signal line and a cell indicates an excitatory connection. The connection between a dashed signal line and a cell indicates an inhibitory connection. Projection from other areas indicates excitatory or inhibitory supervision signals (e.g., excitatory attention selection).

we only consider excitatory top-down weight, which selects neurons selected to increase their potential values. Inhibitory top-down connection can be used if the primary purpose is inhibition (e.g., inhibition of neurons that have not been selected by attention selection signals).

Assume that this network computes in discrete times,  $t = 0, 1, 2, \dots$ , as a series of open-ended developmental experience after the birth at time  $t = 0$ . This network incorporates unsupervised learning and supervised learning. For unsupervised learning, the network produces an output vector at the output layer based on this recurrent computation. For supervised learning, the desired output at the output layer at time  $t$  is set (imposed) by the external teacher at time  $t$ . Because the adaptation of each neuron also uses top-down weights, the imposed motor signal at the output layer can affect the learning of every neuron through many passes of computations.

### III. IN-PLACE LEARNING

Let us consider a simple computational model of a neuron. Consider a neuron which takes  $n$  inputs,  $\mathbf{x} = (x_1, x_2, \dots, x_n)$ . The synaptic weight for  $x_i$  is  $w_i$ ,  $i = 1, 2, \dots, n$ . Denoting the weight vector as  $\mathbf{w} = (w_1, w_2, \dots, w_n)$ , the response of a neuron has been modeled by:

$$y = g(w_1x_1 + w_2x_2 + \dots + w_nx_n) = g(\mathbf{w} \cdot \mathbf{x}), \quad (1)$$

where  $g$  is its nonlinear sigmoidal function, taking into account under-saturation (noise suppression), transition, and over-saturation, and  $\cdot$  denotes the dot production. Next, we

discuss various types of algorithms that learn the weight vector  $w$  and the sigmoidal function  $g$ .

#### A. Learning types

To better understand the nature of learning algorithms, we define five types of learning algorithms:

**Type-1 batch:** A batch learning algorithm  $L_1$  computes  $g$  and  $\mathbf{w}$  using a batch of vector inputs  $B = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_b\}$ , where  $b$  is the batch size. That is,  $(g, \mathbf{w}) = L_1(B)$ , where the argument  $B$  is the input to the learner,  $L_1$ , and the right side is its output. This learning algorithm needs to store an entire batch of input vectors  $B$  before learning can take place.

**Type-2 block-incremental:** A type-2 learning algorithm,  $L_2$ , breaks a series of input vectors into blocks of certain size  $b$  ( $b > 1$ ) and computes updates incrementally between blocks. Within each block, the processing by  $L_2$  is in a batch fashion. The Type-3 algorithm below explains how to learn incrementally with block size  $b = 1$ .

**Type-3 incremental:** Each input vector must be used immediately for updating the learner's memory (which must not store all the input vectors) and then the input must be discarded before receiving the next input. Type-3 is the extreme case of Type-2 in the sense that block size  $b = 1$ . At time  $t$ ,  $t = 1, 2, \dots$ , taking the previous learner memory  $M^{(t-1)}$  and the current input  $\mathbf{x}_t$ , the learning algorithm  $L_3$  updates  $M^{(t-1)}$  into  $M^{(t)}$ :  $M^{(t)} = L_3(M^{(t-1)}, \mathbf{x}_t)$ . The learner memory  $M^{(t)}$  is needed to determine the neuron  $N$  at time  $t$ , such as its sigmoidal  $g^{(t)}$  and its synaptic weight  $\mathbf{w}^{(t)}$ . It is widely accepted that every neuron learns incrementally.

**Type-4 covariance-free incremental:** A Type-4 learning algorithm  $L_4$  is a Type-3 algorithm, but, it is not allowed to compute the 2nd or higher order statistics of the input  $\mathbf{x}$ . In other words, the learner's memory  $M^{(t)}$  cannot contain the second order (e.g., correlation or covariance) or higher order statistics of  $\mathbf{x}$ . The CCI PCA algorithm [35] is a covariance-free incremental learning algorithm for computing principal components as the weight vectors of neurons.

**Type-5 in-place neuron learning:** A Type-5 learning algorithm  $L_5$  is a Type-4 algorithm, but further, the learner  $L_5$  must be implemented by the signal processing neuron  $N$  itself.

The five types of algorithms have progressively more restrictive conditions, with batch (Type-1) being the most general and in-place (Type-5) being the most restrictive. In this work, we deal with Type-5 (in-place) learning.

#### B. Recurrent Network

Consider a more detailed computational model of a layer within a multi-layer network. Suppose the input to the cortical layer is  $\mathbf{y} \in \mathcal{Y}$ , the output from early neuronal processing. However, for a recurrent network,  $\mathbf{y}$  is not the only input. All the input to a neuron can be divided into three parts: bottom-up input from the previous layer  $\mathbf{y}$  which is weighted by the neuron's bottom-up weight vector  $\mathbf{w}_b$ , lateral inhibition  $\mathbf{h}$  from other neurons of the same layer corresponding which is weighted by the neuron's lateral weight vector  $\mathbf{w}_h$ , and the top-down input vector  $\mathbf{a}$  which is weighted by the neuron's

top-down weight vector  $\mathbf{w}_t$ . Therefore, the response  $z$  from this neuron can be written as

$$z = g(\mathbf{w}_b \cdot \mathbf{y} - \mathbf{w}_h \cdot \mathbf{h} + \mathbf{w}_t \cdot \mathbf{a}). \quad (2)$$

Since this is a recurrent network, the output from a neuron will be further processed by other neurons whose response will be fed back into the same neuron again. In real-time neural computation, such a process will be carried out continuously. If the input is stationary, an equilibrium can possibly be reached when the response of every neuron is no longer changed. However, this equilibrium is unlikely even if the input is static since neurons are band-pass filters: A temporally constant input to a neuron will quickly lead to a zero response.

For digital computer, we simulate this analogue network through discrete times  $t = 0, 1, 2, \dots$ . If the discrete sampling rate is much faster than the change of inputs, such a discrete simulation is expected to be a good approximation of the network behavior.

### C. Lateral inhibition

For simplicity, we will first consider the first two parts of input only: the input from the previous layer  $\mathbf{y}$ , and the lateral inhibition part from  $\mathbf{h}$ .

Lateral inhibition is a mechanism of competition among neurons in the same layer. The output of A is used to inhibit the output of neuron B which shares a part of the receptive field, totally or partially, with A.

Since each neuron needs the output of other neurons in the same layer and they also need the output from this neuron, a direct computation will require iterations, which is time consuming. We realize that the net effect of lateral inhibition is (a) for the neurons with strong outputs to effectively suppress weakly responding neurons, and (b) for the weak neurons to less effectively suppress strongly responding neurons. Therefore, we can use a computationally more effective scheme to simulate lateral inhibition without resorting to iterations: Sort all the responses. Keep top- $k$  responding neurons to have non-zero response. All other neurons have zero response (i.e., does not go beyond the under-saturation point).

Next, consider input part  $\mathbf{y}$ .

### D. Hebbian Learning

A neuron is updated using an input vector only when the (absolute) response of the neuron to the input is high. This is called Hebbian's rule.

The rule of Hebbian learning is to update weights when output is strong. And the rule of lateral inhibition is to suppress neighbors when the neuron output is high. Fig. 2 illustrates the lateral inhibition and Hebbian learning. In the figure, hollow triangles indicate excitatory connections, and the solid triangles indicate inhibitory connections.

However, this intuitive observation will not lead to optimal performance. We need a more careful discussion about the statistical properties of neural adaptation under two mechanisms: lateral inhibition and Hebbian learning. With our

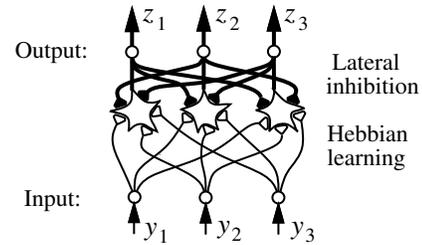


Fig. 2. Neurons adapt using lateral inhibitions and Hebbian Learning

further derivation, the lateral inhibition and Hebbian learning will take a detailed form that leads to a quasi-optimality.

### E. Lobe components

What is the main purpose of early sensory processing? Atick and coworkers [2], [3] proposed that early sensory processing decorrelates inputs. Weng et al. [35] proposed an in-place algorithm that develops a network that whitens the input. Therefore, we can assume that prior processing has been done so that its output vector  $\mathbf{y}$  is roughly white. By white, we mean its components have unit variance and are pairwise uncorrelated. In the visual pathway, the early cortical processing does not totally whiten the signals, but they have similar variance and are weakly correlated. The sample space of a  $k$ -dimensional white input random vector  $\mathbf{y}$  can be illustrated by a  $k$ -dimensional hypersphere.

A concentration of the probability density of the input space is called a lobe, which may have its own finer structure (e.g., sublobes). The shape of a lobe can be of any type, depending on the distribution. For non-negative input components, the lobe components lie in the section of the hypersphere where every component is non-negative (corresponding to the first octant in 3-D).

Given a limited cortical resource,  $c$  cells fully connected to input  $\mathbf{y}$ , the developing cells divide the sample space  $\mathcal{Y}$  into  $c$  mutually nonoverlapping regions, called *lobe regions*:

$$\mathcal{Y} = R_1 \cup R_2 \cup \dots \cup R_c, \quad (3)$$

(where  $\cup$  denotes the union of two spaces). Each region  $R_i$  is represented by a single unit feature vector  $\mathbf{v}_i$ , called the *lobe component*. Given an input  $\mathbf{y}$ , many cells, not only  $\mathbf{v}_i$ , will respond. The response pattern forms a new population representation of  $\mathbf{y}$ .

Suppose that a unit vector (neuron)  $\mathbf{v}_i$  represents a lobe region  $R_i$ . If  $\mathbf{y}$  belongs to  $R_i$ ,  $\mathbf{y}$  can be approximated by  $\mathbf{v}_i$  as the projection onto  $\mathbf{v}_i$ :  $\mathbf{y} \approx \hat{\mathbf{y}} = (\mathbf{y} \cdot \mathbf{v}_i)\mathbf{v}_i$ . Suppose the neuron  $\mathbf{v}_i$  minimizes the mean square error  $E\|\mathbf{y} - \hat{\mathbf{y}}\|^2$  of this representation when  $\mathbf{y}$  belongs to  $R_i$ .

According to the theory of Principal Component Analysis (PCA) (e.g., see [17]), we know that the best solution of column vector  $\mathbf{v}_i$  is the principal component of the conditional covariance matrix  $\Sigma_{\mathbf{y},i}$ , conditioned on  $\mathbf{y}$  belonging to  $R_i$ . That is  $\mathbf{v}_i$  satisfies  $\lambda_{i,1}\mathbf{v}_i = \Sigma_{\mathbf{y},i}\mathbf{v}_i$ .

Replacing  $\Sigma_{y,i}$  by the estimated sample covariance matrix of column vector  $y$ , we have

$$\lambda_{i,1} \mathbf{v}_i \approx \frac{1}{n} \sum_{t=1}^n \mathbf{y}(t) \mathbf{y}(t)^\top \mathbf{v}_i = \frac{1}{n} \sum_{t=1}^n (\mathbf{y}(t) \cdot \mathbf{v}_i) \mathbf{y}(t). \quad (4)$$

We can see that the best lobe component vector  $\mathbf{v}_i$ , scaled by “energy estimate” eigenvalue  $\lambda_{i,1}$ , can be estimated by the *average* of the input vector  $\mathbf{y}(t)$  weighted by the linearized (without  $g$ ) response  $\mathbf{y}(t) \cdot \mathbf{v}_i$  whenever  $\mathbf{y}(t)$  belongs to  $R_i$ . This average expression is crucial for the concept of optimal statistical efficiency discussed below.

The concept of statistical efficiency is very useful for minimizing the chance of false extrema and to optimize the speed to reach the desired feature solution in a nonlinear search problem. Suppose that there are two estimators  $\Gamma_1$  and  $\Gamma_2$ , for a vector parameter (i.e., synapses or a feature vector)  $\theta = (\theta_1, \dots, \theta_k)$ , which are based on the same set of observations  $S = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ . If the expected square error of  $\Gamma_1$  is smaller than that of  $\Gamma_2$  (i.e.,  $E\|\Gamma_1 - \theta\|^2 < E\|\Gamma_2 - \theta\|^2$ ), the estimator  $\Gamma_1$  is more statistically efficient than  $\Gamma_2$ . Given the same observations, among all possible estimators, the optimally efficient estimator has the smallest possible error. The challenge is how to convert a nonlinear search problem into an optimal estimation problem using the concept of statistical efficiency.

For in-place development, each neuron does not have extra space to store all the training samples  $\mathbf{y}(t)$ . Instead, it uses its physiological mechanisms to update synapses incrementally. If the  $i$ -th neuron  $\mathbf{v}_i(t-1)$  at time  $t-1$  has already been computed using previous  $t-1$  inputs  $\mathbf{y}(1), \mathbf{y}(2), \dots, \mathbf{y}(t-1)$ , the neuron can be updated into  $\mathbf{v}_i(t)$  using the current sample defined from  $\mathbf{y}(t)$  as:

$$\mathbf{x}_t = \frac{\mathbf{y}(t) \cdot \mathbf{v}_i(t-1)}{\|\mathbf{v}_i(t-1)\|} \mathbf{y}(t). \quad (5)$$

Then Eq. (4) states that the lobe component vector is estimated by the average:

$$\lambda_{i,1} \mathbf{v}_i \approx \frac{1}{n} \sum_{t=1}^n \mathbf{x}_t. \quad (6)$$

Statistical estimation theory reveals that for many distributions (e.g., Gaussian and exponential distributions), the sample mean is the most efficient estimator of the population mean (see, e.g., Theorem 4.1, p. 429-430 of Lehmann [22]). In other words, the estimator in Eq. (6) is nearly optimal given the observations.

#### F. To deal with nonstationary processes

For averaging to be the most efficient estimator, the conditions on the distribution of  $\mathbf{x}_t$  are mild. For example,  $\mathbf{x}_t$  is a stationary process with exponential type of distribution. However,  $\mathbf{x}_t$  depends on the currently estimated  $\mathbf{v}_i$ . That is, the observations  $\mathbf{x}_t$  are from a nonstationary process. In general, the environments of a developing system change over time. Therefore, the sensory input process is a nonstationary process too. We use the amnesic mean technique

below which gradually “forgets” old “observations” (which use bad  $\mathbf{x}_t$  when  $t$  is small) while keeping the estimator quasi-optimally efficient.

The mean in Eq. (6) is a batch method. For incremental estimation, we use what is called an amnesic mean [35].

$$\bar{x}^{(t)} = \frac{t-1-\mu(t)}{t} \bar{x}^{(t-1)} + \frac{1+\mu(t)}{t} x_t \quad (7)$$

where  $\mu(t)$  is the amnesic function depending on  $t$ . If  $\mu \equiv 0$ , the above gives the straight incremental mean. The way to compute a mean incrementally is not new but the way to use the amnesic function of  $n$  is new for computing a mean incrementally.

$$\mu(t) = \begin{cases} 0 & \text{if } t \leq t_1, \\ c(t-t_1)/(t_2-t_1) & \text{if } t_1 < t \leq t_2, \\ c + (t-t_2)/r & \text{if } t_2 < t, \end{cases} \quad (8)$$

in which, e.g.,  $c = 2, r = 10000$ . As can be seen above,  $\mu(t)$  has three intervals. When  $t$  is small, straight incremental average is computed. Then,  $\mu(t)$  changes from 0 to 2 linearly in the second interval. Finally,  $t$  enters the third section where  $\mu(t)$  increases at a rate of about  $1/r$ , meaning the second weight  $(1+\mu(t))/t$  in Eq. (7) approaches a constant  $1/r$ , to slowly trace the slowly changing distribution.

#### G. Single layer: In-place learning algorithm

We model the development (adaptation) of an area of cortical cells (e.g., a cortical column) connected by a common input column vector  $\mathbf{y}$  by the following Candid Covariance-free Incremental Lobe Component Analysis (CCI LCA) (Type-5) algorithm, which incrementally updates  $c$  such cells (neurons) represented by the column vectors  $\mathbf{v}_1^{(t)}, \mathbf{v}_2^{(t)}, \dots, \mathbf{v}_c^{(t)}$  from input samples  $\mathbf{y}(1), \mathbf{y}(2), \dots$  of dimension  $k$  without computing the  $k \times k$  covariance matrix of  $\mathbf{y}$ . The length of the estimated  $\mathbf{v}_i$ , its eigenvalue, is the variance of projections of the vectors  $\mathbf{y}(t)$  onto  $\mathbf{v}_i$ . The output of the layer is the response vector  $\mathbf{z} = (z_1, z_2, \dots, z_c)$ . The quasi-optimally efficient, in-place learning, single layer CCI LCA algorithm  $\mathbf{z} = \text{LCA}(\mathbf{y})$  is as follows:

- 1) Sequentially initialize  $c$  cells using first  $c$  observations:  $\mathbf{v}_t^{(c)} = \mathbf{y}(t)$  and set cell-update age  $n(t) = 1$ , for  $t = 1, 2, \dots, c$ .
- 2) For  $t = c + 1, c + 2, \dots$ , do
  - a) If the output is not given, compute output (response) for all neurons: For all  $i$  with  $1 \leq i \leq c$ , compute response:

$$z_i = g_i \left( \frac{\mathbf{y}(t) \cdot \mathbf{v}_i^{(t-1)}}{\|\mathbf{v}_i^{(t-1)}\|^{3/2}} \right), \quad (9)$$

The denominator  $\|\mathbf{v}_i^{(t-1)}\|^{3/2}$  in the above expression is such that the standard deviation of the first term is unit. Thus,  $g_i(x)$  can take the same form for all the neurons:

$$g(x) = \frac{1}{1 + e^{-4(x-1)}}. \quad (10)$$

Some useful values of this functions are:  $g(0) = 0.018$ ,  $g(0.5) = 0.12$ ,  $g(1) = 0.5$ ,  $g(1.5) = 0.88$ ,  $g(2) = 0.982$  and  $g'(1) = 1$ .

- b) Simulating lateral inhibition, decide the winner:  $j = \arg \max_{1 \leq i \leq c} \{z_i\}$ , using  $z_i$  as the belongingness of  $y(t)$  to  $R_i$ .
- c) Update only the winner neuron  $v_j$  using its temporally scheduled plasticity:

$$\mathbf{v}_j^{(t)} = w_1 \mathbf{v}_j^{(t-1)} + w_2 z_j \mathbf{y}(t),$$

where the scheduled plasticity is determined by its two age-dependent weights:

$$w_1 = \frac{n(j) - 1 - \mu(n(j))}{n(j)}, w_2 = \frac{1 + \mu(n(j))}{n(j)},$$

with  $w_1 + w_2 \equiv 1$ . Update the number of hits (cell age)  $n(j)$  only for the winner:  $n(j) \leftarrow n(j) + 1$ .

- d) All other neurons keep their ages and weight unchanged: For all  $1 \leq i \leq c$ ,  $i \neq j$ ,  $\mathbf{v}_i^{(t)} = \mathbf{v}_i^{(t-1)}$ .

The neuron winning mechanism corresponds to the well known mechanism called lateral inhibition (see, e.g., Kandel et al. [18] p. 4623). The winner updating rule is a computer simulation of the Hebbian rule (see, e.g., Kandel et al. [18] p.1262). Assuming the plasticity scheduling by  $w_1$  and  $w_2$  are realized by the genetic and physiologic mechanisms of the cell, this algorithm is in-place.

Instead of updating only top-1 neuron, using biologically more accurate “soft winners” where multiple top (e.g., top- $k$ ) winners update, we have obtained similar results.

#### IV. MULTI-LAYER NETWORK

Next, we discuss multi-layer networks. Given each input  $\mathbf{y}(t)$ , the single-layer CCI LCA algorithm automatically generates a representation through its output vector,

As we discussed earlier, object-by-object learning is effective to learn invariance in a global way. However, it does not provide a mechanism that enables the sharing of local invariance or parts.

##### A. Supervision

Without the top-down part, the algorithm discussed above conducts unsupervised learning. However, invariance is effective under supervision. For example, different positions, scales of an object part can be learned by the same set of neurons at a certain layer if some “soft” supervision signal is available at that layer. Then, at the output layer, the global invariance can be more effectively achieved and using fewer examples, by taking advantages of local invariance developed by neurons in the earlier layer. The key is how to provide such “soft” supervision signals at early layers.

Revisit the the top-down part in the expression in Eq. (2), which requires the response from later processing. However, for digital networks, the output from later layers for the current input is not available until after the current layer has computed. Assuming that the network samples the world at a fast pace so that the response of each layer does not

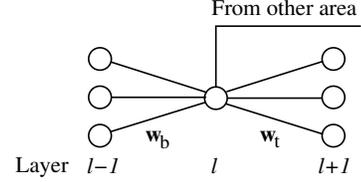


Fig. 3. The bottom-up projection and top-down projection of a neuron at layer  $l$ .

change too fast compared to the temporal sampling rate of the network, we can then use the output from the later layers at  $t - 1$  as the supervision signal. Denoting the response vector at layer  $l$  at time  $t$  to be vector  $\mathbf{y}^{(l)}(t)$  and its  $i$ -th component to be  $y_i^{(l)}(t)$ , we have the model for computation of this supervised and unsupervised network, for each neuron  $i$  at layer  $l$ :

$$y_i^{(l)}(t) = g_i^{(l)}(\mathbf{w}_b \cdot \mathbf{y}^{(l-1)}(t) - \mathbf{w}_h \cdot \mathbf{y}^{(l)}(t) + \mathbf{w}_t \cdot \mathbf{a}(t-1)).$$

Note that the  $i$ -th component in  $\mathbf{w}_h$  is zero, meaning that neuron  $i$  does not use the response of itself to inhibit itself.

In other words, the output from the later layers  $\mathbf{a}$  can take the previous one  $\mathbf{a}(t-1)$ . Using this simple model, the belongingness can take into account not only unsupervised belongingness of  $\mathbf{y}^{(l-1)}(t)$  but also the supervised belongingness enabled by  $\mathbf{a}(t-1)$ . For example, when the firing neurons at a layer fall into the receptive field selected by the attention selection signal represented by  $\mathbf{a}(t-1)$ , its belongingness is modified dynamically by the supervised signal  $\mathbf{a}$ . Therefore, the region that belongs to a neuron can be a very complex nonlinear manifold determined by the dynamic supervision signal (e.g., attention), achieving quasi-invariance locally within every neuron of early layers and global invariance at the output layer.

The next key issue is whether the single-layer in-place learning algorithm can be directly used when the top-down part  $\mathbf{w}_t \cdot \mathbf{a}(t-1)$  is incorporated into the input of every neuron. Let us discuss two cases for the top-down factor  $\mathbf{a}$  and its weight vector  $w_t$ , as shown in Fig. 3: (a) The part that corresponds to the output from the next layer. Since the fan-out vector  $\mathbf{w}_t$  from neuron  $i$  corresponds to the weight vector linking to  $\mathbf{a}$ , the inner product  $\mathbf{w}_t \cdot \mathbf{a}$  directly indicates how much the current top-down part supervises the current neuron  $i$ . (b) The part that corresponds to the output from other areas. Again the corresponding part in the top-down factor  $\mathbf{w}_t$  indicates how the supervision should be weighted in  $\mathbf{w}_t \cdot \mathbf{a}$ . In summary, by including the supervision part  $\mathbf{a}$ , the single-layer in-place learning algorithm can be directly used.

With the top-down projection  $\mathbf{a}_i^{(t-1)}$ , the Eq. (9) should be changed to:

$$z_i = g_i \left( \frac{\mathbf{y}(t) \cdot \mathbf{v}_i^{(t-1)}}{\|\mathbf{v}_i^{(t-1)}\|^{3/2}} + \mathbf{w}_i^{(t-1)} \cdot \mathbf{a}_i^{(t-1)} \right) \quad (11)$$

where  $\mathbf{w}_i^{(t-1)}$  is the fan-out vector of neuron  $i$  and  $\mathbf{a}_i^{(t-1)}$  is the top-down projection vector. When we compute the fan-

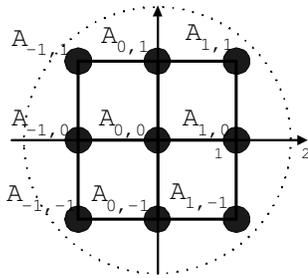


Fig. 4. The  $3 \times 3$  neighbor of the winner  $A_{0,0}$  for updating.

out vector from the fan-in vector of the next layer, use the normalized versions of all fan-in vectors so that their lengths are all equal to one.

### B. Multiple layers

The following is the multi-layer in-place learning algorithm  $\mathbf{z}(t) = \text{MILN}(\mathbf{x}(t))$ . Suppose that the network has  $l$  layers.

**MILN Learning:** Initialize the time  $t = 0$ . Do the following forever (development), until power is off.

- 1) Grab the current input frame  $\mathbf{x}(t)$ . Let  $\mathbf{y}_0 = \mathbf{x}(t)$ .
- 2) If the current desired output frame is given, set the output at layer  $l$ ,  $\mathbf{y}_l \leftarrow \mathbf{z}(t)$ , as given.
- 3) For  $j = 1, 2, \dots, l$ , run the LCA algorithm on layer  $j$ ,  $\mathbf{y}_j = \text{LCA}(\mathbf{y}_{j-1})$ , where layer  $j$  is also updated.
- 4) Produce output  $\mathbf{z}(t) = \mathbf{y}_l$ ;  $t \leftarrow t + 1$ .

### C. Topographic map

In the above discussion, the neurons can reside at any locations and their locations are independent of their weights. The concept of topographic map is that neurons that detect similar features are nearby. This property is important for cortical **modularization** in the sense that the responses for similar inputs appear nearby in cortical area, naturally forming what is called “modules” found in the brain: a certain cortical area is responsible for detecting a class of objects (e.g., faces). Topographic map facilitates generalization because similarity of inputs is translated (mapped) into proximity in cortical locations.

To form a topographic cortical map, we assume that neurons lie in a 2-D “cortical” surface  $\mathcal{S}$ . The topographic map can be realized by updating not only the winner  $A_{0,0}$  at location  $(0, 0)$  in  $\mathcal{S}$  but also its  $3 \times 3$  neighbors in  $\mathcal{S}$ , as shown in Fig. 4. The weights for updating neuron  $A(i, j)$  was computed as follows:

$$W_{i,j} = e^{-d_{i,j}}, \text{ where } d_{i,j} = i^2 + j^2. \quad (12)$$

This weight modifies  $w_2$  in the CCI LCA algorithm through a multiplication, but the modified  $w_1$  and  $w_2$  still sum to one. This is called  $3 \times 3$  update.

### D. Sparse coding

Sparse-coding [8] is a result of lateral inhibition. It allows relatively few winning neurons to fire in order to disregard

less relevant feature detectors. Therefore, the resulting representation is local (small response region) which in turn requires only local connections with the help of a topographic map.

In the algorithm, we only allow  $k$  neurons in each layer to have non-zero response. The output of the top  $i$ -th ranked neuron has a response  $z' = \frac{k-i+1}{k}z$ , if  $i < k$ , where  $z$  is the original response. All other neurons with  $i \geq k$  are suppressed to give zero responses. This is called top- $k$  response.

## V. EXPERIMENTAL EXAMPLES

MILN presented above is an engine for incremental cortical development. For sensory processing, it requires an attention selection mechanism that restricts the connection range of each neuron so that the network can perform local feature detection from the retinal image, as was done by Zhang & Weng [36]. The addition of attention mechanism is our planned future work. Without local analysis, MILN performs global feature-based matching, which should not be expected to outperform other classifiers with local feature analysis in, e.g., character recognition. However, handwritten characters are good for visualizing the invariance properties of the network because the lobe components indicate the corresponding class clearly.

### A. A simple three-class example

The main purpose of this example is to facilitate understanding. For this example, the input images are classified into 3 classes, X, Y, Z. For clarity, we collected two small sets of training samples, containing 27 samples in each. The shift set and scale set mainly show variations in shifting and in scaling, respectively, as shown in Fig. 5 (a) and (b). MILN deals with other types of variations as well, as indicated in these two data sets, because MILN does not model any particular type of variation. This is a very important property of a task-nonspecific developmental program [33].

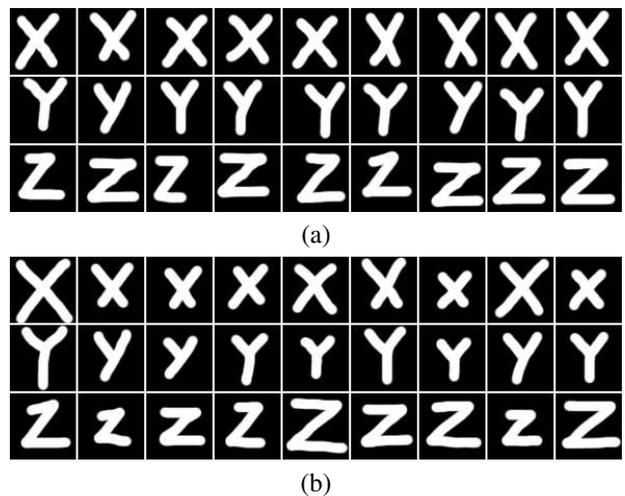


Fig. 5. Two sets of inputs with mainly (a) position variation and (b) size variation.

It is important to note that a 2-layer network with a sufficient number of neurons in both layers is logically sufficient to classify all the training samples, according to the nearest neighbor rule realized by the CCI LCA algorithm using the top-1 winner. However, this is not an interesting case. In practice, the number of neurons is much smaller than the number of sensed sensory inputs.

Then, what are the major purposes of a hidden layer? With attention selection and different receptive fields in the early layers, a later hidden layer is able to recognize an object from its parts. If the earlier layers have developed invariant features, a later hidden layer is able to recognize an object from its invariant parts detection.

Therefore, in the experiment conducted here, our goal is not to perform perfect classification, but rather, to visualize the effects of the introduced mechanisms. Only when these effects are well understood, can we understand the pros and cons of the introduced developmental mechanisms.

In this example, we first decided the network has two layers. The numbers of neurons from layer 1 through layer 2 are  $3 \times 3 = 9$  and 3, respectively. The supervision signals for the output layer are  $(1, 0, 0)$ ,  $(0, 1, 0)$  and  $(0, 0, 1)$ , respectively. The number  $k$  is set to 3 at layer-1, so that the first layer has 3 non-zero firing neurons. The 27 samples in each set were reused 1000 times to form a long input stream.

Fig. 6 gives the developed weights of the first level neurons on these two different training sets. Specifically, Fig. 6 (a) gives the results on position variation and Fig. 6(b) gives the results on size variation. In the development of neurons, unsupervised learning is used in layer-1 and the supervised method is used in layer-2. Furthermore, all methods use the top-1 update.

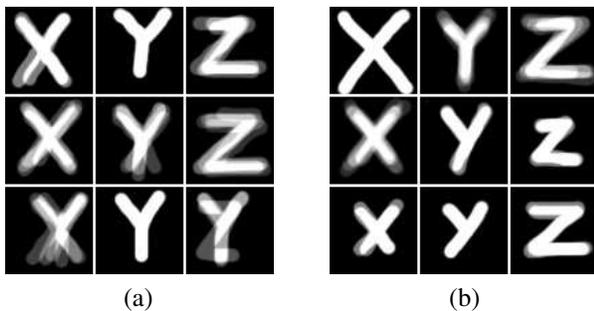


Fig. 6. The weight vectors (lobe components) of the layer-1 neurons, with unsupervised layer-1. All use the top-1 update. (a) Position variation. (b) Size variation.

It can be observed from Fig. 6 that each neuron may average a few nearest samples (in the inner-product space), based on top-1 competition. From Fig. 6, we can see that a lobe component might average samples from different classes. In other words, the region a lobe component belongs to may contain samples from different classes. The invariance

is achieved from case-based mapping: the responding lobe component from layer-1 is mapped to the node representing the corresponding class in layer-2.

Fig. 7 gives the same information as Fig. 6, except that supervised learning is also used in layer-1. We can see that with supervision, the lobe components in layer-1 are more likely to be “pure,” averaging only samples from the same class. Of course, the “purity” is not guaranteed, depending on many factors including the number of neurons. In other words, manifold of the regions tends to cut along the boundary of output class, a concept of discriminant features. Thus, the lobe components take into account both unsupervised learning (to represent the input space well for information completeness) as well as supervised learning (to give higher resource to discriminant features).

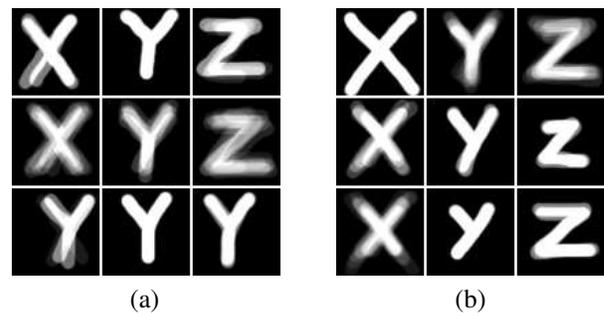


Fig. 7. The weight vectors (lobe components) of the first level neurons, with supervised layer-1. All use the top-1 update. (a) Position variation. (b) Size variation.

Fig. 8 shows the effect of  $3 \times 3$  updating. All the other settings are the same as before. First, compared to  $1 \times 1$  updating,  $3 \times 3$  updating has a tendency to smooth across samples, as we expected. However, the near features look similar. In this experiment, the order of the samples in the training set help to self-organize. Furthermore, it can be observed that by incorporating supervision in layer-1, the lobe components in layer-1 tend to be more “pure.”

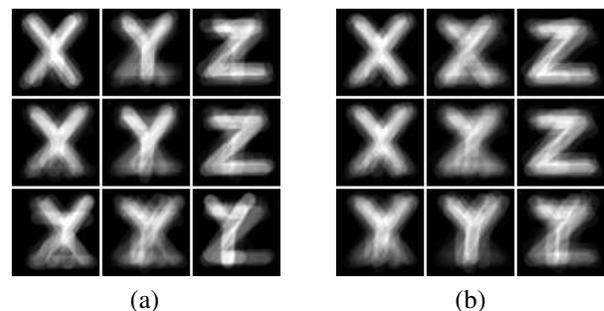


Fig. 8. The weight vectors (lobe components) of the first level neurons, with  $3 \times 3$  updating, using the training set that mainly shows position variation. (a) Unsupervised learning in layer-1. (b) Supervised learning in layer-1.

Fig. 9 shows the effect of a 3-layer network. Layer-1 and layer-2 have 9 neurons each, and layer-3 has 3 neurons. All layers use supervision. Specifically, layer-3 receives supervision from the desired (1,0,0), (0,1,0), and (0,0,1) output, layer-2 receives supervision from layer-3, and layer-1 receives supervision from layer-2. Top-1 update was used for all of the layers. Layer-1 uses top-3 response and layer-2 uses top-1 response. Fig. 9(c) shows multi-layer responses for a given input “Y.”

As we can see, each layer-2 feature is a weighted sum of layer-1 neurons, which gives a mechanism of composition of objects by parts, provided that layer-1 features are localized (which is not true in this example as explained earlier). However, there is no compelling reason to believe that 3-layer MILN will lead to a better recognition rate for global matching (which is not the case of composition by parts).

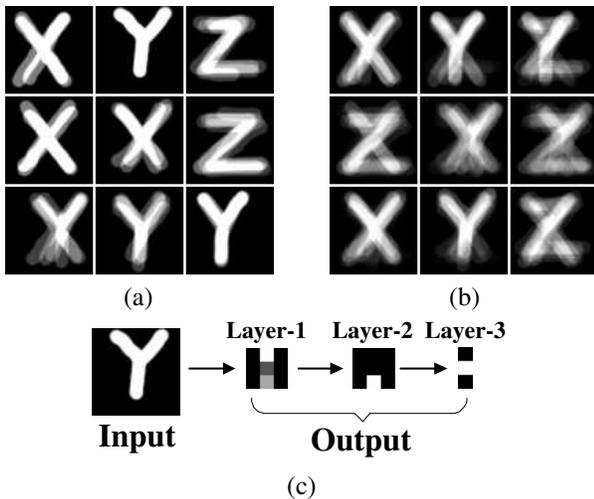


Fig. 9. 3-layer MILN. (a) Lobe components at layer-1. (b) Layer-2 features shown as the sum of lobe components at layer-1 weighted by the connections from layer-1 to layer-2. (c) An example of multi-layer responses for a given input.

From the images of the neuronal synapses, we can see that the classification, and thus invariance, requires cooperation between neurons at different retinal positions and layers.

### B. Topographic maps

We used the MNIST database of handwritten digits available from <http://yann.lecun.com/exdb/mnist/> to show the effect of the topographic map. The data set has 10 classes, from “0” to “9”. The MNIST database was collected among the Census Bureau employees and high-school students with 60000 training samples. The size of images in this database is  $28 \times 28$  pixels. All images have already been translation-normalized, so that each digit resides in the center of the image. Fig. 10 shows some examples.

The MILN network adopted has three layers. The first layer has  $20 \times 20 = 400$  neurons and its  $k$  is set to 40. The

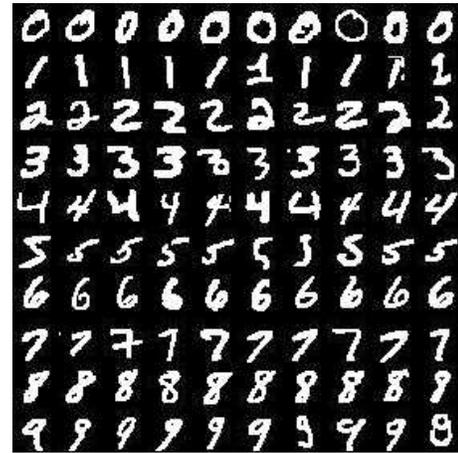


Fig. 10. Some examples of the MNIST training data.

second layer has  $10 \times 10 = 100$  neurons with  $k = 10$ . The third layer has 10 neurons. In the training stage, the signal of the third output layer was imposed by the given class information. In the testing stage, the neuron of the third layer with largest output gives the recognized class identification. Fig. 11 shows neurons of layer-1 trained by the training set. The figure demonstrates that the lobe components represent features that run smoothly across the 2-D “cortical” surface  $\mathcal{S}$ , and is somewhat grouped together excepting “4” grouped into two main sets. That is, within-class variation is naturally organized across a small neighborhood in one or few regions in the 2-D space  $\mathcal{S}$  and different classes are imbedded seamlessly into this 2-D “cortical” space.

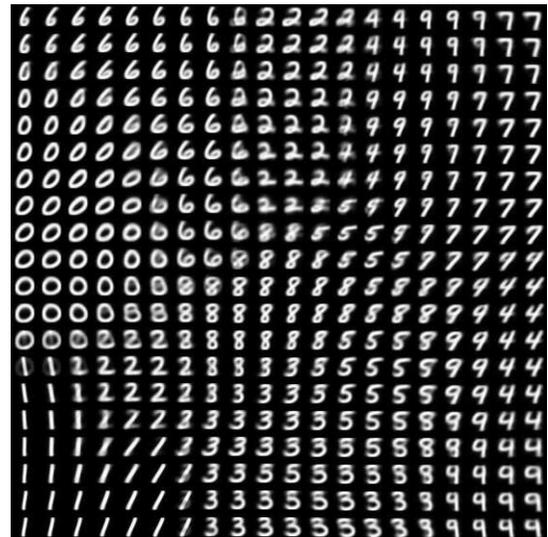


Fig. 11. The topographic map trained from hand-written numerals.

This experiment also shows something very important for development in the natural world: the effect of a restricted environment. If the environment only shows a restricted

set of patterns (e.g., vertical or horizontal edges, numerals, etc.), neurons developed in the earlier layers are significantly biased toward such limited environments. Only sensed space is represented in the representation so the limited resource is dedicated to developmental experience.

## VI. CONCLUSIONS

The new Multi-layer In-place Learning Network (MILN) is a simple, unified computational model that incorporates the Hebbian rule, lateral inhibition, as well as supervised learning for, theoretically, any type of invariance features throughout the network. A hierarchy of representation in the network is developed with increasing invariance from early to later layers. In this work, we concentrate on shift and scale invariance using easy-to-understand numerals as inputs. However, the network does not model any type of variation and, thus, potentially can deal with any type of invariance as long as the input displays such variation. In future work, we will use natural images as input to study what kinds of invariant features, as well as receptive fields, are developed through different layers of the network. The near-optimal efficiency and in-place learning are its two major advantages. Potentially, MILN can be used as a core technology for robots with multiple sensors and multiple effectors to develop perceptual and cognitive skills through real-time experience.

## ACKNOWLEDGMENTS

This work was supported in part by Shanghai Municipal R&D Foundation under contract 045115020, Natural Science Foundation of China under contracts 60533100, 60373020, and 60402007, and LG Electronics. Weng's work was supported by a strategic partnership grant from MSU.

## REFERENCES

- [1] H. Abut, editor. *Vector Quantization*. IEEE Press, Piscataway, NJ, 1990.
- [2] J. J. Atick and A. N. Redlich. Towards a theory of early visual processing. *Neural Computation*, 2:308–320, 1990.
- [3] J. J. Atick and A. N. Redlich. Convergent algorithm for sensory receptive field development. *Neural Computation*, 5:45–60, 1993.
- [4] C. Blakemore and G. F. Cooper. Development of the brain depends on the visual environment. *Nature*, 228:477–478, Oct. 1970.
- [5] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.
- [6] V. Dragoi and M. Sur. Plasticity of orientation processing in adult visual cortex. In L. M. Chalupa and J. S. Werner, editors, *Visual Neurosciences*, pages 1654–1664. MIT Press, Cambridge, Massachusetts, 2004.
- [7] S. E. Fahlman and C. Lebiere. The cascade-correlation learning architecture. Technical Report CMU-CS-90-100, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, Feb. 1990.
- [8] D. Field. What is the goal of sensory coding? *Neural Computation*, 6:559–601, 1994.
- [9] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980.
- [10] D. H. Hubel. *Eye, Brain, and Vision*. Scientific American Library, Distributed by Freeman, New York, 1988.
- [11] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology*, 160(1):107–155, 1962.
- [12] M. Hubener, D. Shoham, A. Grinvald, and T. Bonhoeffer. Spatial relationships among three columnar systems in cat area 17. *J. of Neuroscience*, 17:9270–9284, 1997.
- [13] W. S. Hwang and J. Weng. Hierarchical discriminant regression. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(11):1277–1293, 2000.
- [14] A. Hyvarinen. Survey on independent component analysis. *Neural Computing Surveys*, 2:94–128, 1999.
- [15] A. Hyvarinen and P. Hoyer. Emergence of phase- and shift-invariant features by decomposition of natural images into independent feature subspaces. *Neural Computation*, 12:1705–1720, 2000.
- [16] A. Hyvarinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. Wiley, New York, 2001.
- [17] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.
- [18] E. R. Kandel, J. H. Schwartz, and T. M. Jessell, editors. *Principles of Neural Science*. McGraw-Hill, New York, 4th edition, 2000.
- [19] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, Berlin, 3rd edition, 2001.
- [20] P. A. Kollers, R. L. Duchnick, and G. Sundstroem. Size in visual processing of faces and words. *J. Exp. Psychol. Human Percept. Perform.*, 11:726–751, 1985.
- [21] D. Lee and S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.
- [22] E. L. Lehmann. *Theory of Point Estimation*. John Wiley and Sons, Inc., New York, 1983.
- [23] R. Miikkulainen, J. A. Bednar, Y. Choe, and J. Sirosh. *Computational Maps in the Visual Cortex*. Springer, Berlin, 2005.
- [24] T. A. Nazir and J. K. O'Regan. Some results on translation invariance in the human visual system. *Spatial Vision*, 5(2):81–100, 1990.
- [25] K. Obermayer, H. Ritter, and K. J. Schulten. A neural network model for the formation of topographic maps in the CNS: Development of receptive fields. In *Proc. International Joint Conference on Neural Networks, vol. II*, pages 423–429, San Diego, CA, 1990.
- [26] M. J. O'Donovan. The origin of spontaneous activity in developing networks of the vertebrate nervous systems. *Current Opinion in Neurobiology*, 9:94–104, 1999.
- [27] W. L. Sejnowski and T. Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural Computation*, 14(4):715–770, 2002.
- [28] F. Sengpiel, P. Stawinski, and T. Bonhoeffer. Influence of experience on orientation maps in cat visual cortex. *Nature Neuroscience*, 2(8):727–732, 1999.
- [29] D. C. Somers, S. B. Nelson, and M. Sur. An emergent model of orientation selectivity in cat visual cortical simple cells. *Journal of Neuroscience*, 15(8):5448–5465, 1995.
- [30] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, December 22 2000.
- [31] P. Thompson. Margaret Thatcher: a new illusion. *Perception*, 9:483–484, 1980.
- [32] C. von der Malsburg. Self-organization of orientation-sensitive cells in the striate cortex. *Kybernetik*, 15:85–100, 1973.
- [33] J. Weng. Developmental robotics: Theory and experiments. *International Journal of Humanoid Robotics*, 1(2):199–235, 2004.
- [34] J. Weng, N. Ahuja, and T. S. Huang. Learning recognition and segmentation using the Cresceptron. *International Journal of Computer Vision*, 25(2):109–143, Nov. 1997.
- [35] J. Weng, Y. Zhang, and W. Hwang. Candid covariance-free incremental principal component analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(8):1034–1040, 2003.
- [36] N. Zhang and J. Weng. A developing sensory mapping for robots. In *Proc. IEEE 2nd International Conf. on Development and Learning (ICDL 2002)*, pages 13–20, MIT, Cambridge, Massachusetts, June 12–15 2002.