# USING DUALLY OPTIMAL LCA FEATURES IN SENSORY AND ACTION SPACES FOR CLASSIFICATION

By

Nikita Nitin Wagle

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Computer Science and Engineering

2012

## ABSTRACT

## USING DUALLY OPTIMAL LCA FEATURES IN SENSORY AND ACTION SPACES FOR CLASSIFICATION

**By**

**Nikita Nitin Wagle**

Appearance based methods have utilized a variety of techniques to extract training-data dependent features, such as Linear Disciriminant Analysis (LDA), Support Vector Machine (SVM), k-means clustering, and sparse auto-encoders. The Developmental Networks (DN), which use Lobe Component Analysis (LCA) features developed not only from the image space $X$ but also the effector (action) space $Z$. Since the effector space $Z$ can be taught to represent a set of trainer specified meanings (e.g., type and location in two ports), a DN treats these meanings in a unified way for both detection and recognition for objects in dynamic cluttered backgrounds. However, the DN method has not been applied to publicly available data sets and compared with well-known major techniques. In this work, we fill this void. We describe how the $Z$ information enables the features to be more sensitive to trainer specified output meanings (e.g., type and location). The reported experiments fall into two extensively studied categories — global template based object recognition and local template based scene classification. For the data sets used, the performance of the DN method is better or comparable to some global template based methods and comparable to some major local template based methods while the DNs also provide statistics-based location information about the object in a cluttered scene.

To my family and friends.

## ACKNOWLEDGMENT

The authors would like to thank Matthew Luciw and Yuekai Wang for providing some of their programs for the work reported here.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# Chapter 1

# Introduction

Techniques for appearance-based pattern recognition can be categorized into two types — global template based methods and local template based methods. A global template based method typically assumes that the object of interest has already been detected and cropped and then shifted, scaled, and rotated in a standard way. A local template based method does not use such assumptions. It uses multiple local templates at arbitrary image locations on scene images where features of scene of interest (or objects of interest) can arise anywhere in each input image. Problems of this latter type have been called scene classification (i.e., features indicate a scene type) and sometimes also object recognition (i.e., features indicate an object).

## 1.1 Existing Major Methods

The LDA and SVM algorithms have been widely applied to global template based image matching. The LDA algorithm [22, 23, 24, 4] finds a linear combination of features that separate two or more classes of objects; the resulting combination is used for dimensionality

reduction before further classification. The SVM algorithm [25, 26, 5] constructs a hyper-plane in high-dimensional space, to gain a good separation or a functional margin that has the largest distance to the nearest data points of any class, since, larger the distance, lower is the classification error.

The local template based methods use features derived from local patches of images rather than from the entire images. These local feature representations can be obtained by applying well-known unsupervised feature learning algorithm such as sparse auto-encoders [28, 29], sparse RBM [30], k-means clustering, etc. [10, 31, 3].

We will discuss the global and local template based techniques in more detail later.

## 1.2 Characteristics of DN

The DN (Developmental Network) framework [8] has used global templates, as in [1], derived from image datasets, as well as local templates derived from image patches of cluttered scenes, as in [2].

The novel characteristics of a DN over some well-known pattern recognition methods are discussed below and a summary is provided in Table 1.1.

| Characteristics | LDA | SVM | Sparse Methods | DN |
|---|---|---|---|---|
| Environmental openness | Possible | No | No | Yes |
| High dimensional sensors | Yes | Possible | Yes | Yes |
| Completeness | Yes | No | No | Yes |
| Real-valued actions | No | No | No | Yes |
| Real-time training | No | No | No | Yes |
| Incremental learning | No | Yes | No | Yes |
| Perform while learning | No | No | No | Yes |
| Input having background | No | No | Yes | Yes |

Table 1.1: Comparison of characteristics of global and local template based methods. Sparse methods include sparse auto-encoders, sparse RBMs, k-means clustering, etc. in this work.

1) Environmental openness: The DN is meant to learn and improve from realistic scenes and its (human taught with effector-supervised) experience of actions through such scenes, even though its performance is imperfect due to its limited computational resource and limited learning experience. It is not assumed that only certain objects are of interest (e.g., face), so the environment is open.

2) Low and high dimensional sensors: The DN is applicable to low- and high-resolution video. Lower resolution video speeds up computational time and saves storage space, but should work reasonably well regardless of the resolution, since the internal operations of DN are based on normalized inner product.

3) Completeness in representation for different amounts of teaching: All the features in DN emerge as optimal representations learned from the experience. The DN does not use, and not limited by, handcrafted features (e.g., SIFT, or oriented edges which may fail if edges are sparse or absent in an input).

4) Instead of learning features from image space $X$ only, DN finds optimal clusters in the space of $X \times Z$, which are sensorimotor features, where $Z$ is the spaces of effectors (including labels represented as vectors). By optimal, we mean maximal likelihood (ML) in the representation of the space of $X \times Z$ based on limited computational resource in DN (e.g., the number of neurons) and the limited amount of teaching. The analytical results in this work show the advantages of using both $X$ and $Z$ for learning features (i.e., discriminant).

5) Real-valued actions: The DN always accepts and processes real-valued sensory and effector (motor) information, instead of human supplied discrete class labels. Discrete class labels are special cases of real-valued actions (e.g., each neuron in $Z$ represents a class), the set of all possible robotic actions is more general than a set of all class labels, since actions

can be taught or created for the physical world without a human predefined class.

6) Real-time training: During training, the sensory and memory refreshing rate must be high enough so that each physical event (e.g., motion of a person or the motion of the head of self) can be temporally sampled and processed in real-time (e.g., hopefully about 15Hz to 30Hz when the computers are fast). Although we used class labels in our experiments here for comparisons, class labels of objects appearing in the real scene require a human to supply which restrict the possibility of real-time training. The DN can take actions directly, regardless of whether they represent a real-valued action or a class label.

7) Incremental learning: Acquired skills must be used to assist in the acquisition of new skills, as a form of "scaffolding." This requires incremental processing. Batch processing is used by DN. By incremental processing, we mean that each new observation $\mathbf{x} \in X$ and $\mathbf{z} \in Z$ must be used to update the current DN and the current $(\mathbf{x}, \mathbf{z})$ must be discarded before the next $(\mathbf{x}, \mathbf{z})$ can be acquired. Incremental learning is necessary for learning in real time, as the amount of data in the sensory and motor stream is virtually unbounded.

8) Perform while learning: At any time, the human teacher can teach the DN by imposing an action on its effector port $Z$ (motor-supervised learning). As soon as the human lets $Z$ free, the DN generates $\mathbf{z} \in Z$ as its best prediction or best action at this time.

9) Input having background: A typical scene has objects of interest in a cluttered background. The method must be able deal with object in any position in unknown cluttered backgrounds. A global template method and a local template method are different in the sense that the former attempts to match the entire scene but the latter attempts only to match some local templates. A local template method can deal with cluttered background. Local patches in DN add"votes" to a single supervised location in the Location Motor (LM)

of $Z$, although different local patches are at different locations in the input image.

## 1.3 Novelty

This work has two major novelties — First, the DN algorithm is originally meant for a more general problem of spatiotemporal event detection and recognition in complex background, and is not restricted to the global and local template based image matching problem here. However, it is desirable that we apply it to the space-only problems here so that it can be compared with major pattern recognition techniques. The ways to use the effector space $Z$ are very different between DN and other compared methods.

Second, this work is the first to use the DN for problems where multiple local features contribute to the classification (type) and localization (location). Multiple firing feature detectors ($Y$ neurons) vote for the corresponding single neuron in the Type Motor (TM) area and the corresponding single neuron in the Location Motor (LM) area. The area $Z$ consists of two subareas TM and LM.

DN can be used for shallow learning (single level of features), deep learning (multiple levels of features), and a mixed shallow and deep learning. Here, we concentrate on shallow learning. Experimentally, we show that the shallow learning (DN) is at least comparable to the deep learning methods (sparse auto-encoders, k-means clustering, etc.). In a local feature based DN, the "where" information from LM is applied as input to the DN in a supervised manner during the training phase along with the "what" information or type of object from TM applied as input to the DN. While $Z$ is free during a test session, through multiple updates of DN, the location information and type information feed from $Y$ to $Z$ and then from $Z$ back to $Y$ reinforce each other and suppress inconsistent features, like relaxation.

The object location must be consistent with type and the type must be consistent with location. Such an relaxation between $Y$ and $Z$ is faster than in deep learning methods.

The remainder of the paper is organized as follows. In chapter 2, we review previous work. In chapter 3, we explain DN. The experimental results are presented in chapter 4, and chapter 5 gives concluding remarks.

# Chapter 2

# Previous Work

In this chapter, we briefly discuss the most widely applied algorithms for global and local template based image matching.

## 2.1 Global Template Based Methods

There are many methods for global template matching such as Wavelets, Neural Networks, Correlation, PCA, LDA, SVM, etc.. Of these, we discuss two types of methods — LDA and SVM in this work.

### 2.1.1 LDA Algorithm

The LDA algorithm [22, 23, 24] finds a linear combination of features which characterize or separate two or more classes of data instances, so that dimension of the data instances is reduced prior to being projected onto the linear space. The LDA algorithm, previously devised by Daniel Swets and Juyang Weng [4], is based on optimal subspace generation; the subspace is generated using two projections - a Karhunen Loéve projection to produce a set

of MEF (most expressive features) features followed by a discriminant analysis projection to produce a set of MDF (most discriminant features) features; together they are called the DKL (discriminant Karhunen Loéve) projection. The MEF projection discriminates a dataset based on lighting variations between two classes, and chooses a reduced set of most expressive features to be projected onto the MDF space; whereas the MDF projection defines an optimal discrimination between two classes. A set of MEF and MDF features is generated for each image in training dataset; an image from the testing dataset is projected in the same subspace. In the subspace, a Euclidean distance is computed between the two feature spaces to find a set of k-nearest neighbors to recognize the class to which the image in the testing set belongs. The MEF features forming the MEF vector $\mathbf{Y}$ are the unit eigenvectors associated with the $m$ largest eigenvalues of the covariance matrix of the vector $\mathbf{X}$ of training image instances, where $m$ is chosen such that sum of the unused eigenvalues is less than some fixed percentage $P$ ($P = 5\%$) of the entire training dataset, and $m < n$, n being the original number of features. The MDF generation finds a projection matrix $W$ that maximizes the ratio $\frac{dets_b}{dets_w}$, i.e. maximize between-class scatter while minimizing within-class scatter. The between-class scatter matrix is defined as $S_w = \sum_{i=1}^{c} \sum_{j=1}^{n_i} (\mathbf{Y}_j - \mathbf{M}_i)(\mathbf{Y}_j - \mathbf{M}_i)^t$ , for $i = 1, 2, ..., c$ classes with class mean $\mathbf{M}_i$ for $n_i$ samples from class $i$. and the within-class scatter matrix is defined as $S_b = \sum_{i=1}^{c} (\mathbf{M}_i - \mathbf{M})(\mathbf{M}_i - \mathbf{M})^t$, for mean vector, $\mathbf{M}$ for all data instances from all classes.

### 2.1.2 SVM Algorithm

The SVM algorithm [24, 25, 26], on the other hand, maps a set of n-dimensional data points from a finite-dimensional space to a high- dimensional space, constructing a (n-1) dimensional

maximum- margin hyperplane or set of hyperplanes in a high- dimensional space, so that a good gap called the functional margin, that has the largest distance to the nearest training data points of any class, is gained. The testing data points are then mapped to the same space and are predicted to belong to a class based on which side of the gap they fall on; and fits a non-linear kernel function to the maximum- margin hyperplane when separation is non-linear in finite- dimensional space. The support vectors are a set of data points that lie closest to the decision boundary; these points have direct bearing on the position of the decision boundary.

If $(x_i, y_i)$ are a set of data instance-class label pairs, $i$ = 1, 2, ..., $l$ where $x_i \in R^n$ and $y_i \in 1, -1^l$, the SVM algorithm finds an optimum solution to the following problem:

$$\begin{aligned} \min_{\mathrm{w},b,\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\phi(x_i + b)) \geq 1 - \xi_i, \\ & \xi_i \geq 0. \end{aligned}$$

Here, w is a normal vector to the hyperplane, and parameter $\frac{b}{\|\mathbf{w}\|}$ determines the offset of the hyperplane from the origin along the normal vector. The data instances $x_i$ are mapped from a finite- dimensional space to a high-dimensional space by the transformation function $\phi$. If no hyperplane can separate data into 'yes' or 'no' classes, the hyperplane that can split the data instances in as clean manner as possible, while maximizing the distance to the nearest well split data instances is chosen, and the slack variable, $\xi_i$, measures the degree of misclassification of the data instances $x_i$ such that C > 0 is the penalty parameter of $\xi_i$. The kernel function of the transform is expressed in terms of the transformation function $\phi(x_i)$ as $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$. The three kernel functions, that are extensively used are

the linear kernel, Gaussian RBF (radial basis function) kernel and polynomial kernel [26]-

linear kernel - $K(x_i, x_j) = x_i^T x_j$

RBF kernel - $K(x_i, x_j) = exp(-\gamma \|x_i - x_j\|^2), \gamma > 0$

polynomial kernel - $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0$

The most obvious choice of kernel function is a Gaussian RBF kernel, since, unlike the linear kernel, it can separate data points into classes when the relation between data points and class labels is non-linear. Also, unlike the polynomial kernel, the RBF kernel has fewer hyperparameters that influence the complexity of kernel model selection. However, if the number of features of the data instances is extensively large, then a linear kernel provides a better separation as compared to an RBF kernel. The RBF kernel function is dependent on the Euclidean distance of $x_i$ (training data instance or support vector) from $x_j$ (testing data point). The support vector is placed at the center of the RBF kernel and $\sigma$ determines the area of influence the support vector has over the data space. The efficiency of an RBF kernel depend on a good choice of two parameters C and $\gamma$, where $\gamma = \frac{1}{2\sigma}$, and $\sigma$ determines the area of influence of the support vector over the new data instance. The best combination of C and $\gamma$ is gained through a grid-search with exponentially growing sequences of C and $\gamma$ ($C \in 2^{-5}, 2^{-3}, ..., 2^{13}, 2^{15}$) and $\gamma \in 2^{-15}, 2^{-13}, ..., 2^{1}, 2^{3}$). Each combination of C and $\gamma$ parameters is tested for using cross validation, and the parameters with best cross-validation accuracy are chosen. The SVM training model, which is used for testing new data instances, is trained on the complete training dataset using the selected values of C and $\gamma$ [5].

Figure 2.1 shows the two-class clustering of data points by the LDA, SVM and DN methods. The LDA boundary is linear, whereas the SVM boundary is characterized by the non-linear kernel function chosen. The DN algorithm separates data points into voronoi

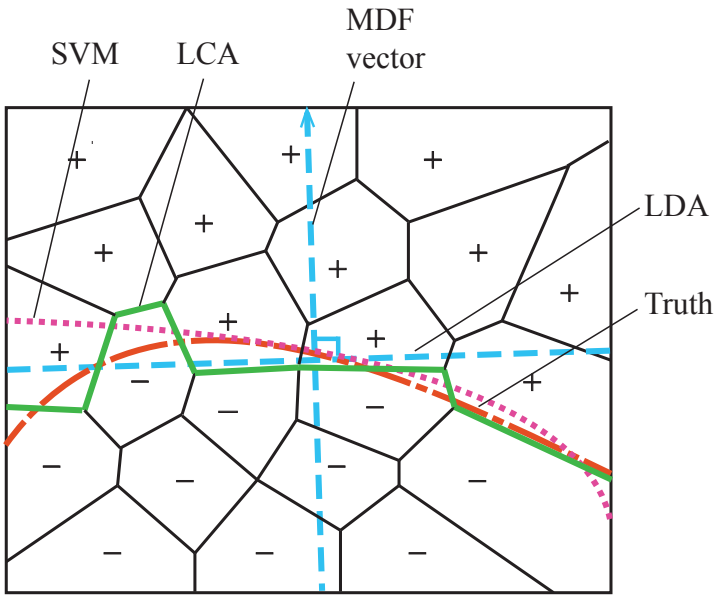



Figure 2.1: The two-class clustering of data points when subject to linear separator in MDF space in LDA algorithm, RBF kernel based non-linear SVM classification versus the separation of data points into voronoi regions in DN algorithm (For interpretations of the references to color in this and all other figures, the reader is referred to the electronic version of this thesis.).

regions, such that the boundary separating the two regions is at an equal distance from both points.

## 2.2 Local Template Based Methods

A simple feature learning framework that incorporates a local feature-learning algorithm like sparse auto-encoders [28, 29], sparse RBMs [30], K-means clustering, and Gaussian mixture models [10, 31], as a "black box" module [3] within has been studied to discover local feature representations from unlabeled data instances. A set of random patches is extracted from unlabeled training data instances, such that each patch has dimension $w \times w$ and has d-channels, where $w$ is the size of the receptive field; each $w \times w$ patch can be represented as

a vector in $\mathbb{R}^N$ of pixel intensity values, such that $N = w \cdot w \cdot d$. A dataset of randomly sampled patches $X = x^{(1)}, ..., x^m$ is then constructed, where $x^i \in \mathbb{R}^N$. Then, each patch $x^{(i)}$ is normalized by subtracting the mean and dividing by the standard deviation of its elements, after which the dataset is optionally whitened. When the data is pre-processed, an unsupervised learning algorithm viewed as a "black box" module takes the dataset $X$ and outputs a function $f : \mathbb{R}^N \to \mathbb{R}^K$ and maps input vector $x^{(i)}$ to a new feature vector $y = f(x) \in \mathbb{R}^K$ of $K$ features, where $K$ is a parameter of the unsupervised learning algorithm used and the $k$th feature is $f_k$. A $(n - w + 1) \times (n - w + 1)$ representation with $K$ channels is defined for each $w \times w$ "subpatch" of the training data; $y^{(i,j)}$ is referred to as the $K$-dimensional feature representation extracted from location $i, j$ of the training data. The feature representation, $y^{(i,j)}$ is split into four equally sized quadrants and the sum in each quadrant is computed to yield a reduced $K$-dimensional representation of each quadrant for a total of $4K$ features on which a linear classification algorithm is applied to identify new data instances [3].

# Chapter 3

# The Developmental Network Model

In this work, we consider the DN model to have a general purpose brain area $Y$, which is connected with the sensory area $X$ and the motor area $Z$, as illustrated in Figure 3.1, in which the order of areas from low to high is $X, Y, Z$. Much of the material in this section in extracted from Weng and Luciw 2011 [9].

## 3.1 Area Function

At the first $c$ time instances, during the "prenatal" learning phase, the first $c$ neurons of $A$ in $\{X, Y, Z\}$ initialize their synaptic vectors $V = (\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_c)$, where each synaptic vector $\mathbf{v}_i$ is initialized using the input pair $\mathbf{p}_i = (\mathbf{b}_i, \mathbf{t}_i)$, where the bottom up input is $\mathbf{b}_i$ and the top-down input is $\mathbf{t}_i$, $i = 1, 2, ..., c$, and initialize the firing ages $A = (a_1, a_2, ..., a_c)$, such that each firing age $a_i$ is initialized to be zero, $i = 1, 2, ..., c$.

After the "birth" phase, at each time instant, each area $A$ computes its response $\mathbf{r'}$ from its input pair $\mathbf{p} = (\mathbf{b}, \mathbf{t})$ based on its adaptive part $N = (V, A)$ and its current response $\mathbf{r}$. The current response $\mathbf{r}$ is regulated by the attention vector $\mathbf{b}_a$. Each area $A$ updates its

adaptive part $N$ to $N'$ as follows:

$$(\mathbf{r}', N') = f(\mathbf{b}, \mathbf{r}, \mathbf{t}, \mathbf{r}_a, N)$$

where $f$ is the area function. The attention supervision vector $\mathbf{r}_a$ is used to softly avoid the area $A$ from excessively learning background. The attention supervision vector $\mathbf{r}_a$ has the same dimension as $\mathbf{r}$. The attention supervision vector $\mathbf{r}_a$, in this work, suppresses all the $A$ neurons to zeros except $3 \times 3 = 9$ ones centered at the correct object location. Since, the vector $\mathbf{r}_a$ is biologically driven by other connected brain areas, it is not very accurate during early development phases, which may result in learning some irrelevant information (background information). Thus, there arises a need for the soft internal attention vector $\mathbf{r}_a$ to be removed in future for construction of a more powerful model of brain-like development.

The area $A$, whether $X, Y, or Z$ compute and update in a unified way as described above. But $X$ area does not have bottom-up input and $Z$ area does not have top-down input. The $X$ area and $Z$ area are nerve terminals.

## 3.2 Area Computation

We introduce that the receptive field (RF) of a neuron should contain three parts — sensory RF (SRF), motor RF (MRF) and lateral RF (LRF), respectively. The effective field (EF) of each neuron should also include three parts: sensory EF (SEF), motor EF (MEF), and lateral EF (LEF), respectively. The SRF, MRF, LRF, SEF, MEF, LEF together form the hextuple fields of a neuron as shown in Figure 3.3(a). The lateral connections — connections with neurons in the same area that are strongly correlated or anticorrelated, whether inhibitory or

excitatory, are used by each cortical area $(X, Y, Z)$ so that the neurons in each area can find their roles. The highly correlated cells are connected by excitatory connections to generate a smooth map that globally covers a rough "terrain" but gradually becomes selective and local to fit the details of the "terrain". Whereas, highly anti-correlated cells are connected by inhibitory connections, so that the neurons in the same cortical area can respond to different features, and allow only top ranked neurons to fire and update, ensuring that other weakly responding neurons do not fire so that they do not learn irrelevant information and also keep their long-term memory intact.

The lateral connections within $Y$ area of the DN model, in this work, are simulated by the top-$k$ competition among neurons for fast-speed identification of top-k winners within each network instead of actual inhibitory connections. The top-k mechanism is used for quick sorting of the winner neuron in real-time (every 30ms), when the software or hardware cannot run fast enough i.e. cannot update the entire network at 1kHz or above. In our work, we hand-pick the value of $k$, assuming that the value is largely gene pre-dispositioned. According to the sparse coding theory explained in [1, 10], if only top-k neurons in each area fire and update, then those that do not update act as long term memory for current context.

Consider an area $A$ in $\{X, Y, Z\}$.; if both bottom-up input $\mathbf{b}$ and top-down input $\mathbf{t}$ are associated with each area, each neuron in $A$ has a weight vector $\mathbf{v} = (\mathbf{v}_b, \mathbf{v}_t)$, corresponding to the two area inputs $(\mathbf{b}, \mathbf{t})$, else the part of input that is not associated with the area is not included in the notation. The sum of two normalized inner products gives the pre-action of the neuron:

$$r(\mathbf{v}_b, \mathbf{b}, \mathbf{v}_t, \mathbf{t}) = \dot{\mathbf{v}} \cdot \dot{\mathbf{p}},$$

where $\dot{\mathbf{v}}$ is the unit vector of the normalized synaptic vector, $\mathbf{v} = (\dot{\mathbf{v}}_b, \dot{\mathbf{v}}_t)$, and $\dot{\mathbf{p}}$ is a unit

vector of the normalized input vector, $\mathbf{p} = (\dot{\mathbf{b}}, \dot{\mathbf{t}})$. The inner product measures the degree of match between the two directions — $\dot{\mathbf{v}}$ and $\dot{\mathbf{p}}$, because $r(\mathbf{v}_b, \mathbf{b}, \mathbf{v}_t, \mathbf{t}) = \cos(\theta)$ where $\theta$ is the angle between two unit vectors $\dot{\mathbf{v}}$ and $\dot{\mathbf{p}}$. Let the area $A$ under consideration be $Y$, connecting with bottom-up input area $X$ and top-down input area $Z$. The area $Y$ with many neurons has a set of cluster centers:

$$\{(\mathbf{v}_x, \mathbf{v}_z) \mid \mathbf{v}_x \in X, \mathbf{v}_z \in Z\}.$$

Each center $(\mathbf{v}_x, \mathbf{v}_z)$ is the center of the corresponding Voronoi tile in the area's input space $X \times Z$, and is an instance of co-occurrence, which is linguistically impure in human language or does not have exact description as a concept of the external environment in natural language. Thus, we assume that the linguistic impurity must be true for all neurons inside the brain which are not under direct supervision of the external environment. In our DN based LCA algorithm [1], we use top-k mechanism to show that lateral connections within neurons in each area enable them to sort top winner neurons within each time step $t_n$, $n = 1, 2, 3, ....$ Consider the weight vector of neuron $i$ is $\mathbf{v}_i = (\mathbf{v}_{bi}, \mathbf{v}_{ti})$, $j = 1, 2, ..., c$, and if $k = 1$, the single winner neuron $j$ is identified as follows:

$$j = \arg \max_{1 \le i \le c} r(\mathbf{v}_{bi}, \mathbf{b}, \mathbf{v}_{ti}, \mathbf{t}).$$

If $c$ is sufficiently large and the set of $c$ synaptic vectors distributes well, i.e. the density of the $c$ points well approximates the observed probability density in parallel input space, then

there is a high probability that both parts of the winner neuron $j$ match well:

$$\mathbf{v}_{bj} \approx \mathbf{b} \text{ and } \mathbf{v}_{tj} \approx \mathbf{t}$$

not counting the length of the vectors because of the length normalization in $r(\mathbf{v}_b, \mathbf{b}, \mathbf{v}_t, \mathbf{t})$. Consider area $A$ to be $Y$-area; we want the response value $y_j$ to approximate the probability for $(\mathbf{x}, \mathbf{z})$ to have $\mathbf{v}_j = (\mathbf{v}_{xj}, \mathbf{v}_{zj})$ as the nearest neighbor. If $k = 1$, only one winner neuron fires with a response value $y_j = 1$ and the rest of the neurons in the area do not fire, so $y_i = 0$ for $i \neq j$. Thus, in general, for $k > 1$, a dynamic scaling function shifts and scales the pre-action potential of each neuron $r_i$ in a dynamic manner, so that the winner has a response value $y = 1$ and the $(k+1)$-th and weaker neurons have a response value zero. Without a need to explicitly solve $c$ simultaneous equations, each dynamic function[1] $g$ depends on values in $\mathbf{y}$ at time $t_n$ to give the updated response $y_i$ but for the next time instant $t_{n+1}$.

## 3.3 Dual Optimality of Lobe Component Analysis

The Lobe Component Analysis (LCA) [1] is a model for long-term memory retention, and uses a dually optimal (optimal in space and time) framework that casts long-term memory and short-term memory together by the optimal distribution of the limited number of neurons of each area in the input space $X \times Z$ — optimal Hebbian learning, spatially and temporally, as shown in Figure 3.2.

[1] Consider $r_1 \geq r_2, ..., \geq r_{k+1}$ and $r_{k+1} \geq r_i$ for all $i > k+1$. Then $y_i = g(r_i)$, $i = 1, 2, ..., c$, where $g(r)$ depends on the ranked values $r_1$ and $r_{k+1}$ — $g(r_i) = (r_i - r_{k+1})/(r_1 - r_{k+1})$, if $i \leq k$. The response value $r_{k+1}$ is subtracted from the input $r$ and the difference is divided by $r_1 - r_{k+1}$ so that the response vector $\mathbf{y}$ of the area has $k$ positive components with the maximum value is at most 1. The remaining $c - k$ neurons in $\mathbf{y}$ have response value zero — $g(r_i) = 0$ for all $i > k$.

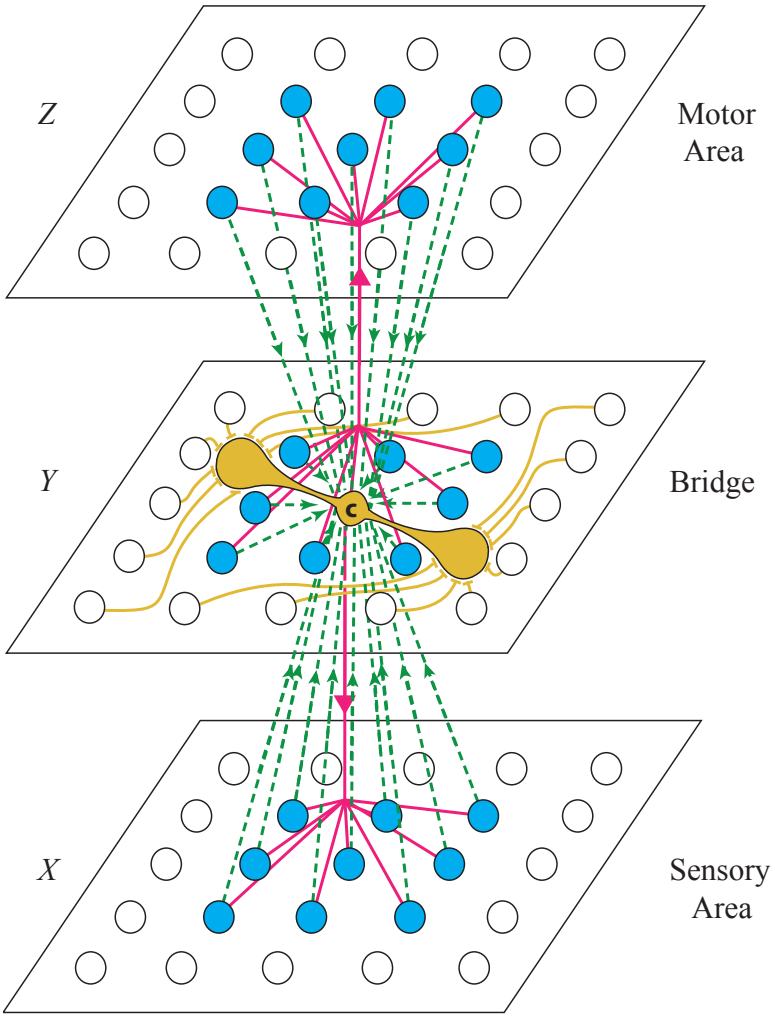



Figure 3.1: An illustration of LCA network model.

In LCA framework, the connection pattern of each neuron in $Y$ area is shown in 3.1. The $Y$ area acts as a bridge between the sensory area $X$ and the motor area $Z$. The two-way local connections in green represent neuronal input and in pink represent neuronal output. In the same area, near neurons are connected by excitatory connections for smoothness of representation, and far neurons are connected by inhibitory connections, hence, competition between neurons result in detection of different features by different neurons.

The meaning of the dual (spatiotemporal) optimality of LCA is shown in Figure 3.2. The upper area is a 2-D representation of the positions for the neurons in several stacked

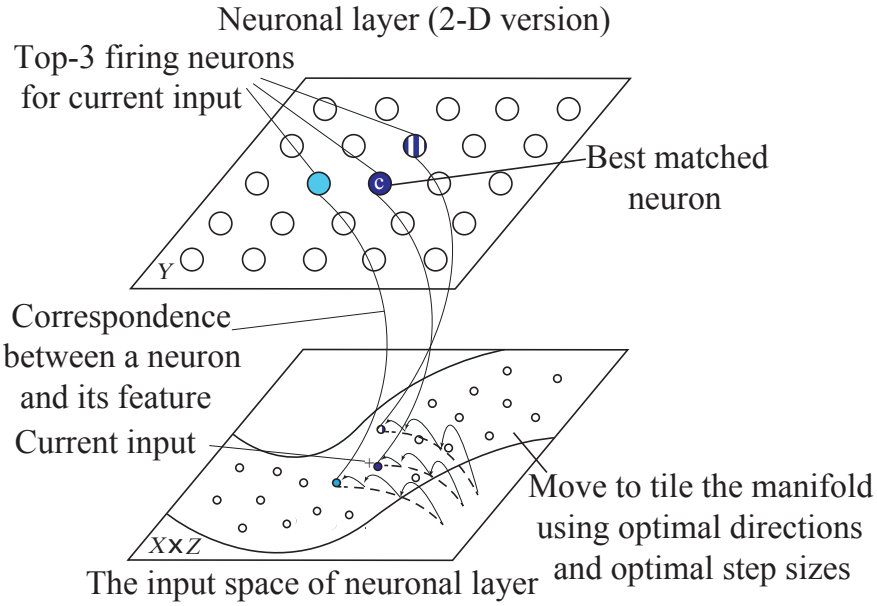



Figure 3.2: The meaning of dual optimality in LCA network.

layers within $Y$ area. The top-3 firing neurons in the $Y$ area, shown in different shades and patterns of blue (top-1 neuron in dark blue shade at the center, top-2 in light blue shade, and top-3 in vertical blue n white stripes), are context-dependent working memory for the current context and the ones that do not fire are context dependent long-term memory for the current context. A 2-D representation of a very high dimensional input space $P = X \times Z$ (a manifold of the distribution of input data, which is very sparse in $P$ and of a much lower dimension than the apparent dimension of $P$) of the cortical area $Y$ is shown by the lower area, in which each neuron in $Y$ plane is linked with its synaptic weight vector by a curved arc; the synaptic weight vectors of $Y$ represented in $P$ as small dots define a Voronoi diagram in $P$.

LCA is dually optimal, its spatial optimality means that the target tiling by the Voronoi diagram in the manifold is optimal to minimize the representation error for $P = X \times Z$, whereas its temporal optimality means that the neuronal weight of firing neurons must move

toward their unknown best target the quickest throughout the development process. Here, the updating trajectory of every neuron is a highly nonlinear trajectory. The statistical efficiency theory for neuronal weight update (amnesic average) ensures minimum error in each age-dependent update. This means that not only the direction but also every step size of each neuronal update is nearly optimal, is autonomously determined.

## 3.4 Where-What Network

The "Where-What" Network (WWN), a DN embodiment, has a motor zone (type or "what" motor area TM), which is used to teach a type concept and another motor zone (location or "where" motor area LM), which is used to teach a location concept.

The Figure 3.3(b) shows the three areas of a simple WWN — retina $X$, simple brain $Y$, and motor $Z$. The $Z$ area has two concept areas LM and TM. The connecting wires indicate that the pre-synaptic and post-synaptic neurons have co-fired; a two-way arrow indicates two one-way connections whose two sysnapses are generally not same. The weight is the frequency of pre-synaptic co-firing when the post-synaptic neuron fires. Within each cortical area, each neuron connects with highly correlated neurons using excitatory connections but connect with highly anti-correlated neurons using inhibitory connections. Every $Y$ neuron is *location-specific* and *type-specific*, corresponding to an object type (marked by its color pattern) and to a location block ($2 \times 2$ each). Each LM neuron is location-specific and type-invariant, and each TM neuron is type-specific and location-invariant. Each $Z$ neuron pulls all applicable cases from $Y$ area neurons as well as boosts all applicable cases in $Y$ as top-down context. A WWN does not treat features in $Y$ as a "bag-of-features" to reduce the number of training samples, because of the inner-product-based neuronal response for

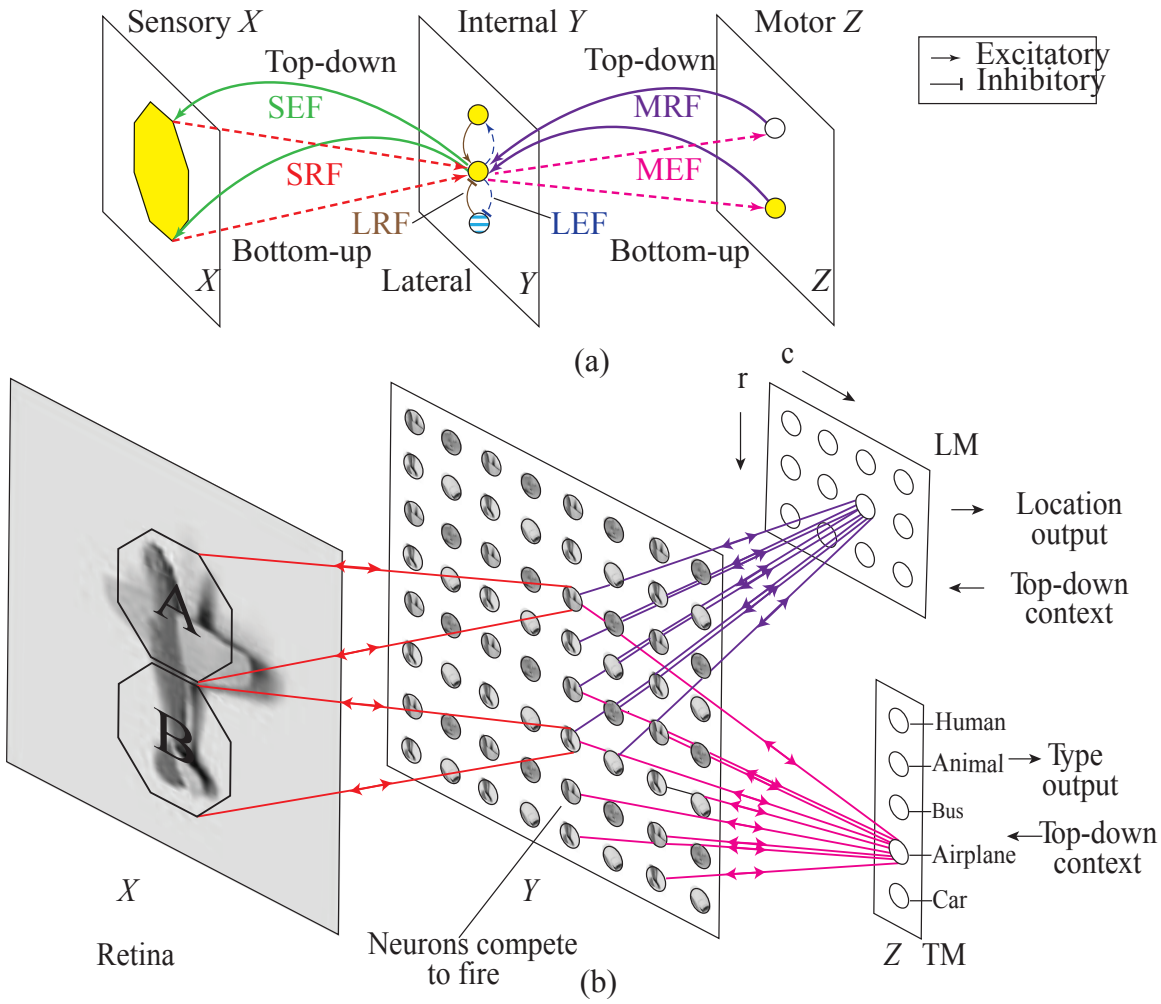



Figure 3.3: A simple WWN.

$Z$. The location of each element in a vector $\mathbf{x}$ affect the outcome of the inner product.

An object contour from a foreground, by default, is approximated by an octagon; however, refinement of the contour needs synapse maintenance, discussed in [2], which automatically cuts off synapses that have bad matches or whose pre-synaptic input is from the background. In our experiments, we do not use the synapse maintenance. In case of shortage of $Y$ neurons, the $Y$ area optimally uses the limited resource by implicitly balancing the trade off between type or "what" motor concept representation and location or "where" concept representation. Hence, each $Y$ neuron must deal with the misalignment between an object and its receptive

field, simulating a more realistic resource situation.

This shows that, a WWN does not require the human programmer to model each concept but instead enables un-modeled concepts, in this case, location and type, to be learned interactively and incrementally as actions; it enables such concepts to serve as goals, whether supervised or self-generated, but in general serve as attended spatiotemporal equivalent top-down contexts; and it enables such goals to direct perception, recognition and behavior emergence [9].

## 3.5 Receptive Fields Are Selective and Dynamic

The hextuple field concept means that the receptive field of a neuron is *attention-selective* and *temporally dynamic*, which means that a different subpart is active at a different time [11], depending on top-down attention and the competition in early areas.

Conventionally, a sensory neuron has a receptive field, but a motor neuron does not. However, the SRF of each motor neuron is global, but selective and dynamic, since only winner $Y$ neurons fire at a time instant. The dynamic and selective property of SRF gives clear explanation of why each TM neuron is *locationally invariant* and *type specific*, an why each LM neuron is *type invariant* and *location specific*. Similarly, an MRF is also selective and dynamic in the sense that different motor actions boost a V1 neuron at different contexts. Each $Y$ neuron connects one neuron in LM and TM, respectively, thus, an MRF is typically disconnected.

We, hence, state that the motor area $Z$ can be taught to represent any human communicable concept that are produced by muscle contractions. Verbal concepts such as written, verbal, sign languages, etc. can be communicated through muscle languages, whereas non-

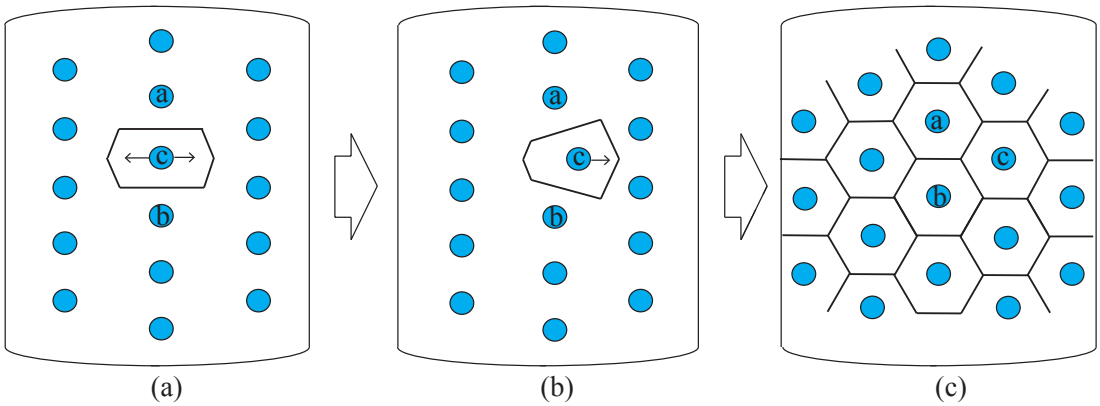



Figure 3.4: The square-like tiling property of the self-organization in a cortical area.

verbal concepts such as reaching, grasping and manipulation can be produced through muscle procedures.

## 3.6 Properties of DN

Here, we discuss two important properties of DN — distance-sensitive property and top-down representational property relevant to our work.

First, the expression for neuronal learning can be rewritten as $\mathbf{v}_j \leftarrow \mathbf{v}_j + w(n_j)(y\mathbf{p} - \mathbf{v}_j)$. Thus, the amount of vector change $w(n_j)(y\mathbf{p} - \mathbf{v}_j)$ is proportional to the vector difference $y\mathbf{p} - \mathbf{v}_j = \mathbf{p} - \mathbf{v}_j$ when $y = 1$. We call it *the distance-sensitive property*. With this property, we have the *square-like tiling* theorem:

**Theorem 1 (Square-like tiling)** *Suppose that the learning rule in a self-organization scheme has the distance-sensitive property. Then the neurons in the area move toward a uniformly distribution (tiling) in the space of areal input* $\mathbf{p}$ *if its probability density is uniform.*

The proof is available in [9]

The square-like tiling property of DN is illustrated in Figure 3.4. In a uniform input space,

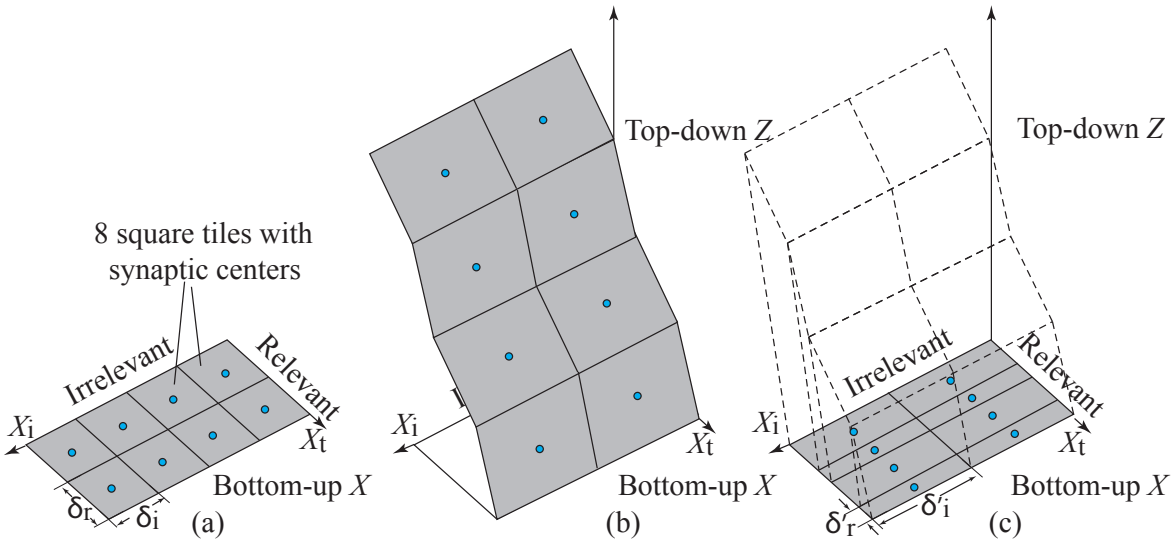



Figure 3.5: The top-down representational effect.

neurons in an layer self-organize until their Voronoi regions are nearly isotropic (square-like to nearly hexagons in 2-D). The Voronoi region of neuron $c$ is very anisotropic — elongated horizontally) — resulting horizontal pulling is statistically stronger as shown in Figure 3.4(a). A horizontal perturbation leads to continued expected pulling in the same direction (rightward in this case) as shown in Figure 3.4(b). Through many updates, the Voronoi regions are nearly isotropic, ideally regular hexagons but generally square-like as illustrated in Figure 3.4(c).

Note that "move toward" does not state how fast. The speed of self-organization depends on the optimality of the step sizes; the temporal optimality of DN deals with the speed.

Second, as shown in Figure 3.5, learning using top-down inputs sensitizes neurons to action-relevant bottom-up input components (e.g., foreground pixels) and desensitize to irrelevant components (e.g., leaked-in background pixels). This is true during operation, when top-down input is unavailable during free-viewing. This is called the *top-down representational effect*. We have the top-down effect theorem:

**Theorem 2 (top-down effect)** *Given a fixed number of neurons in a self-organization scheme that satisfies the distance sensitivity property, adding top-down input from motor $Z$ in addition to bottom-up input $X$ enables the quantization errors for action-relevant subspace $X_r$ to be smaller than the action-irrelevant subspace $X_i$, where $X = X_r \times X_i$.*

The proof is available in [9].

The top-down inputs sensitize the response for relevant bottom components although which are relevant is unknown. Without top-down input, square Voronoi tiles in the bottom-up space give the same quantization width for irrelevant component $X_i$ and the relevant component $X_r$: $\delta_i = \delta_r$. All samples in each tile is quantized as the point (synaptic vector) at the center as illustrated in Figure 3.4(a). With top-down inputs during learning, square tiles cover the observed "pink" manifolds, indicating the local relationships between $X_r$ and $Z$ as shown in Figure 3.4(b). When top-down $Z$ is not available during free-viewing, each tile is narrower along direction $X_r$ than along $X_i$: $\delta'_r < \delta'_i$, meaning that the average quantization error for relevant $X_r$ is smaller than that for irrelevant $X_i$ as shown in Figure 3.4(c).

The above theorem gives two consequences (due to $\delta'_i > \delta'_r$): First, action-relevant bottom-up inputs are salient (e.g., toys and other Gestalt effects). Thus, we need to re-consider the conventional thinking that bottom-up saliency is static and probably totally innate. Second, relatively higher variation through a synapse gives information for cellular synaptic pruning in all neurons, to delete their links to irrelevant components. This was used in synapse maintenance [2], but this capability has been turned off because the image data available from the public datasets, that we use here, do not present variation of background pixels like a dynamic physical world would.

# Chapter 4

# Experimental Results

In this chapter, we conduct two experiments, the DN algorithm is applied to: (a) A global template, in which the patterns to be recognized have been shifted, rotated, and scaled so that the entire input image contains mainly the pattern of interest. (b) A set of local templates, in which each input image contains a cluttered background, which means that the detection and pre-normalization in (a) of object of interest has not been done. The experimental results, so obtained, are compared to other widely-used major algorithms (LDA, SVM, sparse coding or k-means clustering, etc.) in the pattern recognition community.

## 4.1 Global Template Based Methods

In the first experiment, a set of well-framed (a dataset is termed 'well-framed' if only a small variation in the size, position and orientation of objects within the image is allowed) "feature images" (global template) from two datasets — Weizmann and FERET are trained on LDA, SVM and LCA-net algorithms. The Weizmann face dataset (A courtesy of Weizmann Institute) is a set of images, of size $88 \times 64$, from 28 humans, each having 30 images with all

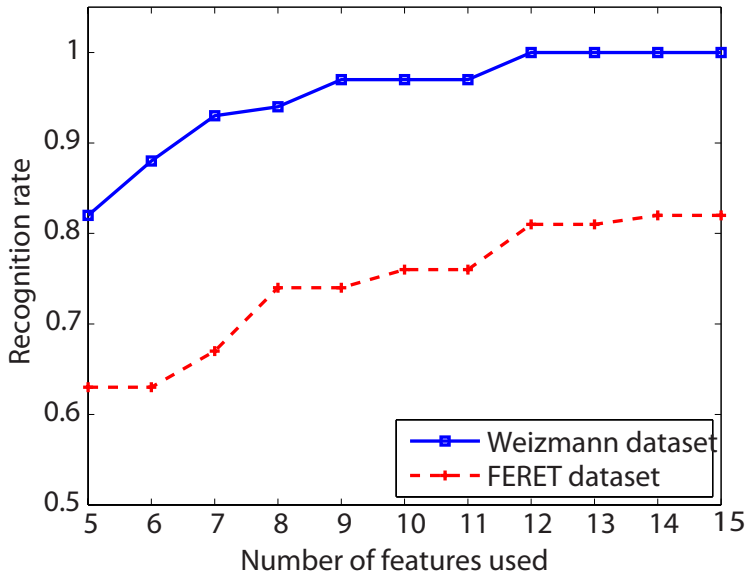



Figure 4.1: Number of features is varied in MDF subspace to gain maximum recognition rate when trained on LDA classifier; Weizmann dataset need 12 features, whereas FERET needed 14 features.

possible combinations of two different expressions under three different lighting conditions with five different orientations, of which 812 images are used as training set and 28 images are used as testing set (one from each class). The FERET face dataset [27] is a set of 1762 images, of size $88 \times 64$, from 1010 humans, of which 1624 images are used as training set and 138 images are used as testing set.

| C\$\gamma$ | 0.002 | 0.0078 | 0.0313 | 0.125 | 0.5 |
|---|---|---|---|---|---|
| 0.0313 | 4.3% | 4.3% | 4.3% | 4.3% | 4.3% |
| 0.125 | 4.3% | 4.3% | 7.2% | 4.3% | 4.3% |
| 0.5 | 4.3% | 7.2% | 7.2% | 79.7% | 79.7% |
| 2 | 7.2% | 79.7% | 84.7% | 79.7% | 84.7% |
| 8 | 80.4% | 80.4% | 84.7% | 84.7% | 84.7% |

Table 4.1: Disjoint test recognition rate from C and $\gamma$ parameter variation on FERET dataset

The Weizmann as well as the FERET data instances fall in the case, where number of feature vectors are more than the number of data instances, causing the within class scatter matrix $S_W$ to degenerate. In such a case, if the MEF and MDF projections are done on

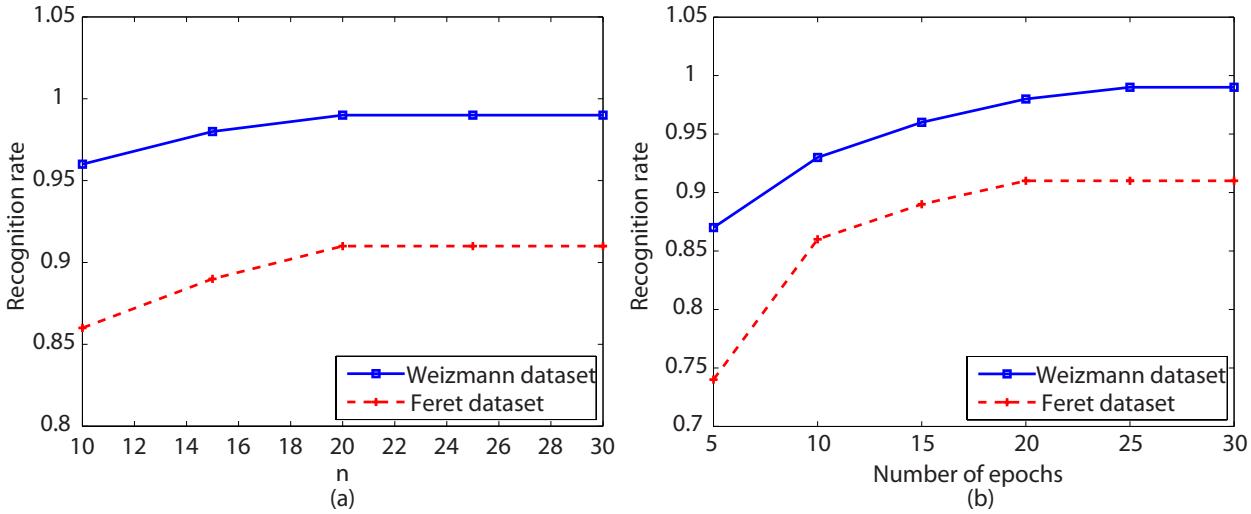



Figure 4.2: (a) The variation number of Y-area neurons to gain maximum attainable recognition rate in LCA algorithm on Weizmann and FERET datasets (b) The variation number of epochs in training phase to gain maximum attainable recognition rate in LCA algorithm on Weizmann and FERET datasets

the original data instances, it will render the $S_w$ non-invertible due to the high-dimensional input data relative to the number of data instances. The DKL projection on the datasets finds a set of MEF and MDF features by using the LDA algorithm [22, 23, 24, 4] to deal with the $S_w$ degeneracy issue, since the within-class scatter matrix non-invertibility issue does not arise in the MEF space; followed by a k-nearest neighbor search to label the class to which the image from the testing data belongs. The $n$-dimensional original image space is projected onto an $m$-dimensional MEF space; the $m$ is chosen such that for $s$ data samples from $c$ classes, $m + c \leq s$. Thus, $m$ is constrained to be less than the rank of $S_w$ to make $S_w$ non-degenerate. Also, since $S_w^{-1} S_b$ can have at most $c$-1 non- zero eigen values, $k$-dimensional MDF space is constrained to $k \leq c - 1$; so that, $k + 1 \leq c \leq m \leq s - c$. The number of MDF features used to linearly classify the testing data were varied and recorded for the Weizmann and FERET datasets as shown in Figure 4.1. The LDA classifier gains 0.6% error on the Weizmann dataset after 30 cross-validation tests (a different image from

each class is used for testing in each test) and 17.4% error on FERET data as shown in Table 4.2.

On the other hand, prior to application of SVM algorithm [25, 26, 5], the images in both, the training and testing data were linearly scaled to the range [0 1], to avoid features in numeric range dominate those in lower ones as well as to avoid numerical complications during calculation of inner product of feature vectors. The support vectors for FERET dataset needed an exhaustive parameter search to find the best (C,$\gamma$), which are not priory known for a dataset, so that the testing data can be predicted with higher accuracy. The grid search for the best (C,$\gamma$) parameters is done using cross-validation for C within the range of $2^{-5}$ to $2^{15}$ and for $\gamma$ within the range of $2^{-15}$ to $2^{3}$, and the prediction with the best cross-validation accuracy is chosen as shown in Table 4.1(b). A coarse grid-search (C = $2^{-5}$, $2^{-3}$, ... $2^{15}$; $\gamma = 2^{-15}$, $2^{-13}$, ... $2^{3}$) finds the best value of parameters C and $\gamma$ as 2 and 0.0313 respectively which cause 3.6% error on training and 15.3% error on testing data. The Weizmann dataset did not gain a good recognition rate for any combination of C and $\gamma$, a 15.9% error on the training set and a 3.6% error on the testing set occurred by applying linear Gaussian kernel as shown in Table 4.2.

In the training phase of the LCA algorithm [1], a sequence of images are presented to $X$ by inserting a background image between every consecutive image from the training set - background, image one, background, image two, background, image three... and so on, such that each stimulus in sensory area has a 2 virtual time unit lasting. For each dataset, the number of $Y$ area neurons is varied from $n$ = 10, 15, ..., 25 and the training phase is run through variation in number of epochs from 5, 10, 15, ...., 30 and the recognition rate on the testing data with the best combination of the parameters is recorded as shown in

Figure 4.2. The response value of top-k neurons is ranked so that they replace repeated iterations that take place among a large number of two-way connected neurons in the same layer; in the experiment, the k-value is chosen as 1. The internal representation of Y-area and Z-area after the training phase are shown in Figure 4.3. Each Y neuron detects a type as in Figure 4.3(b); 28 different RGB color intensities represent 28 different types in TM. Due to limited neuronal resources in Y area, some neurons deal with multiple object types at multiple pixel locations. The bottom-up weights (TM) of two of Z neurons as in Figure 4.3(c) are normalized to the range 0 to 255 such that the pixel value indicates the strength of the connection between the corresponding Y neuron (the same (row,col) location as the pixel) and the Z neuron.The Weizmann and FERET datasets had to be trained on 625 $Y$ area neurons ($n = 25$) after which 0% error was gained on both, Weizmann and FERET dataset in resubstitution test (data in training phase is the same as that from testing phase), and 0% error was gained on Weizmann dataset and 8.7% error was gained on FERET dataset in disjoint test (data in training phase is not the same as that from testing phase), both in 20 epochs as shown in Table 4.2.

| Dataset | Test | LDA | SVM | LCA |
|---|---|---|---|---|
| Weizmann | resubstitution | 100% | 100% | 100% |
| | disjoint | 99.4% | 90.3% | 100% |
| | per image training time | 17.7s | 3.9s | 4.1s |
| | per image testing time | 4.3s | 1.2s | 1.4s |
| FERET | resubstitution | 100% | 96.4% | 100% |
| | disjoint | 82.6% | 84.7% | 91.3% |
| | per image training time | 12.2s | 7.0s | 4.6s |
| | per image testing time | 5.7s | 4.1s | 1.7s |

Table 4.2: Global template based method comparison on Weizmann and FERET datasets

## 4.2 Local Template Based Methods

In the second experiment, a set of local templates is derived from two object recognition datasets, NORB and CIFAR-10 using the idea of local patch extraction from foreground in [2] and the recognition rate so gained is compared to some major local-feature learning algorithms [28, 29, 30, 31, 3].

| Patch size\k | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $11 \times 11$ | 35.7% | 42.3% | 70.0% | 51.5% | 42.3% |
| $12 \times 12$ | 39.4% | 42.3% | 73.1% | 54.7% | 43.5% |
| $13 \times 13$ | 39.4% | 45.9% | 79.3% | 59.9% | 43.5% |
| $14 \times 14$ | 39.4% | 53.2% | 81.8% | 59.5% | 49.7% |
| $15 \times 15$ | 48.0% | 53.2% | 84.2% | 61.3% | 54.1% |
| $16 \times 16$ | 48.0% | 59.8% | 85.1% | 68.0% | 54.1% |
| $17 \times 17$ | 53.1% | 60.1% | 93.8% | 72.9% | 63.5% |
| $18 \times 18$ | 53.1% | 63.5% | 93.8% | 76.0% | 63.5% |
| $19 \times 19$ | 53.1% | 63.5% | 94.0% | 81.6% | 71.9% |
| $20 \times 20$ | 63.2% | 72.6% | 94.2% | 83.2% | 71.9% |
| $21 \times 21$ | 63.2% | 72.9% | 94.2% | 83.7% | 74.5% |
| $22 \times 22$ | 69.9% | 74.5% | 95.0% | 85.2% | 74.5% |
| $23 \times 23$ | 67.1% | 71.3% | 94.2% | 81.6% | 74.2% |
| $24 \times 24$ | 61.3% | 69.8% | 94.2% | 81.6% | 66.3% |
| $25 \times 25$ | 56.2% | 68.8% | 94.0% | 79.2% | 45.0% |
| $26 \times 26$ | 43.7% | 65.2% | 93.5% | 77.5% | 32.1% |
| $27 \times 27$ | 26.4% | 65.2% | 81.2% | 51.3% | 24.7% |
| $28 \times 28$ | - | 49.1% | 74.9% | 34.7% | - |
| $29 \times 29$ | - | 49.1% | 74.9% | 27.8% | - |

Table 4.3: Recognition rate gained from variation of top-k firing neurons and patch size on NORB dataset

The Norb dataset with elimination of complex background (small- NORB dataset) [6] is a set of images of 50 toys, of size $96 \times 96$, belonging to 5 classes namely, four-legged animals, human figures, airplanes, trucks and cars, imaged by two cameras under 6 lighting conditions, 9 elevations and 18 azimuths, of which 24300 images were used for training and testing each. The CIFAR-10 dataset [7] is a set of 60000 color images, of size $32 \times 32$, belonging to 10

| Patch size\k | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $5 \times 5$ | - | - | 25.2% | - | - |
| $6 \times 6$ | - | 22.2% | 39.4% | - | - |
| $7 \times 7$ | - | 30.3% | 51.5% | 23.2% | - |
| $8 \times 8$ | 25.2% | 30.3% | 64.6% | 38.3% | - |
| $9 \times 9$ | 36.3% | 44.4% | 72.7% | 52.5% | 29.3% |
| $10 \times 10$ | 36.3% | 44.4% | 80.8% | 63.6% | 43.4% |
| $11 \times 11$ | 35.3% | 42.4% | 79.8% | 61.6% | 42.4% |
| $12 \times 12$ | 36.3% | 44.4% | 76.7% | 59.6% | 43.4% |
| $13 \times 13$ | 36.3% | 44.4% | 71.7% | 56.5% | 43.4% |
| $14 \times 14$ | 32.3% | 43.4% | 68.6% | 53.5% | 40.4% |
| $15 \times 15$ | 32.3% | 43.4% | 67.6% | 51.5% | 40.4% |
| $16 \times 16$ | 29.3% | 40.4% | 64.6% | 49.5% | 40.4% |
| $17 \times 17$ | 29.3% | 40.4% | 62.6% | 48.4% | 40.4% |
| $18 \times 18$ | 29.3% | 40.4% | 59.6% | 45.4% | 40.4% |
| $19 \times 19$ | 29.3% | 40.4% | 57.5% | 44.4% | 39.4% |
| $20 \times 20$ | 25.2% | 38.3% | 56.5% | 42.4% | 38.3% |
| $21 \times 21$ | 25.2% | 38.3% | 54.5% | 42.4% | 38.3% |
| $22 \times 22$ | 25.2% | 38.3% | 51.5% | 40.4% | 38.3% |

Table 4.4: Recognition rate gained from variation of top-k firing neurons and patch size on CIFAR-10 dataset

classes, with 6000 images per class, of which 50000 images are used for training 10000 images are used for testing.

Each image in the training set of NORB images is rotated at an angle of $20^\circ$, and the process is looped until the original image is obtained, so that 18 rotated instances of each image in training dataset are obtained. Each $20^\circ$ rotated image instance is added to the original training set of NORB images so that a test image belonging to the same class is placed into the correct class in spite of the elevation angle to which it is raised. In the CIFAR-10 dataset, however, each patch extracted from the training image is trained at each location within the training image (each image is circularly shifted row-wise and then column-wise, and the process is looped until the original image is gained back), for instance, a patch of size 10 is trained at $(32-10+1) \times (32-10+1)$ i.e. $23 \times 23$ different locations within the image.

The circular shift of training set of images is to make sure that the object to be recognized is trained in each location possible within the image, so that a test image containing an object from the same class is placed into the correct class, in spite of the position at which it is located within the image.

| Algorithm | Recognition rate |
|---|---|
| Conv. Neural Network | 93.4% |
| Deep Boltzmann Machine | 92.8% |
| Deep Belief Network | 95.0% |
| Deep Neural Network | 97.1% |
| Sparse Auto-encoder | 96.9% |
| Sparse RBM | 96.2% |
| K-means (Hard) | 96.9% |
| K-means (Triangle) | 97.0% |
| K-means (Triangle, 4000 features) | 97.2% |
| WWN | 95.0% |
| per frame training time | 1.3s |
| per image testing time | 0.8s |

Table 4.5: Local template based method comparison on NORB dataset [3]

| Algorithm | Rec rate | Loc Error (in pixels) |
|---|---|---|
| Raw pixels | 37.3% | - |
| 3-Way Factored RBM (3 layers) | 65.3% | - |
| Mean-covariance RBM (3 layers) | 71.0% | - |
| Improved Local Coord. Coding | 74.5% | - |
| Conv. Deep Belief Net (2 layers) | 78.9% | - |
| Sparse Auto-encoder | 73.4% | - |
| Sparse RBM | 72.4% | - |
| K-means (Hard) | 68.6% | - |
| K-means (Triangle) | 77.9% | - |
| K-means (Triangle, 4000 features) | 79.6% | - |
| WWN | 80.8% | 1.7 |
| per frame training time | 1.5s | |
| per image testing time | 1.2s | |

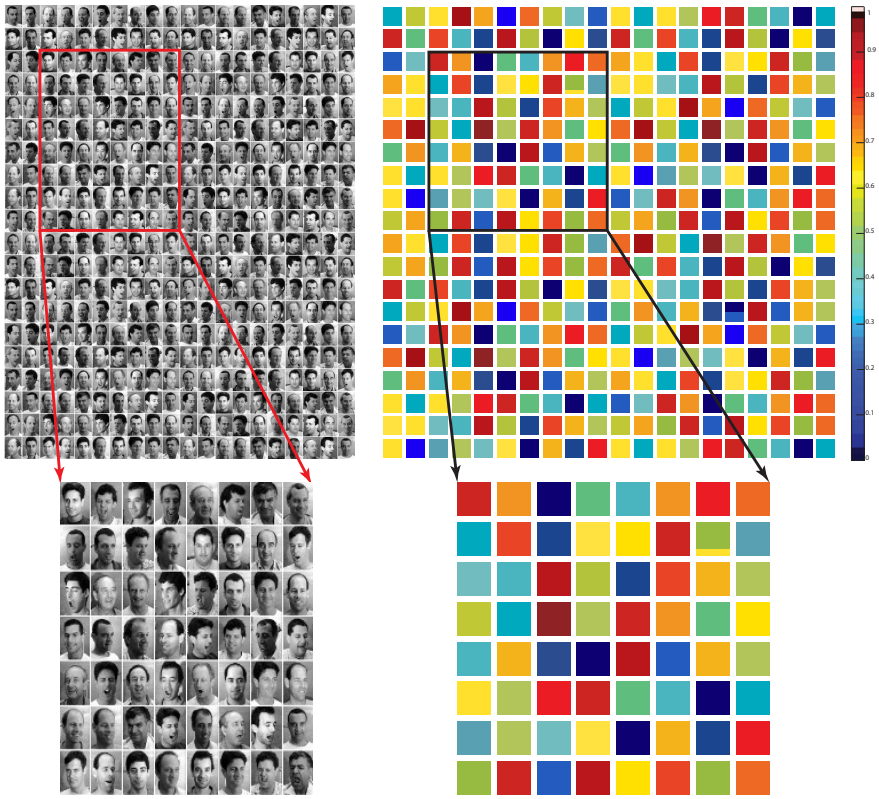
Table 4.6: Local template based method comparison on CIFAR-10 dataset [3]

The local feature templates derived from NORB and CIFAR-10 images are trained on the WWN algorithm [2] varying training parameters like, k-value in the number of top-k

neurons firing, size of input patch of local features, thickness of the Y-area neurons, etc.. A recognition rate of 95.0% was obtained from NORB dataset for a patch of size $22 \times 22$ (the size of the original image being $96 \times 96$, the thickness of the network being 10, when top-3 neurons fire as shown in Table 4.3; whereas a recognition rate of 80.8% was obtained from CIFAR-10 images for a patch of size $10 \times 10$ (the size of the original image being $32 \times 32$), the thickness of the network being 30, when top-3 neurons fire as shown in Table 4.4. The number of training epochs are varied from 0 to 15 and the recognition rate at each epoch is plotted as shown in Figure 4.4. Each epoch performs 3 iterations for reinforcement learning of LM and TM input by WWN. The internal representation of Y area and Z area after the training phase of the WWN algorithm are visualized in Figure 4.5 and Figure 4.6 respectively. Each Y neuron (in all depths) detects a type as in Figure 4.5(b) in a specific location as in Figure 4.5 (c). Due to limited neuronal resources in Y area, some neurons deal with multiple object types at multiple pixel locations. The bottom-up weights (TM and LM) of two of Z neurons as in Figure 4.6 are normalized to the range 0 to 255 such that the pixel value indicates the strength of the connection between the corresponding Y neuron (the same (row,col) location as the pixel) and the Z neuron.The Figure 4.7 shows the location error (in pixels) over 20 epochs for CIFAR-10; the location error remains constant after 10 epochs. The recognition rate, thus obtained, is compared to the recognition rate obtained by some major local-feature learning algorithms as shown in Table 4.5 and Table 4.6 to show that a WWN performs comparable to them.

In NORB dataset, the object of interest is already centered, thus, "where" information in DN gets no room for improvement. However, in the CIFAR-10 dataset, the object of interest appears at different locations within a scene. Thus, the "where" information from

LM ensures that the object is present in the scene as a configuration (not necessarily rigid) that is consistent with the training experience, not as scattered broken parts, for instance, the head and the tail of an airplane is present with an observed configuration and not as separate dis-assembled parts, which might be a reason for the recognition rate to be slightly better than other methods.



(a) Bottom- up $Y$ (b) Top- down $Y$ (TM)

"Amir-1" neuron "Amir-2" neuron

(c) Bottom-up $Z$ (TM)

Figure 4.3: Visualization of weights in $Y$ area and $Z$ area of Weizmann dataset (a) bottom-up weights for $Y$ ($20 \times 20$ $Y$ neurons in which each cell has dimension $88 \times 64$) (b) top-down weights (TM) for $Y$ ($20 \times 20$ $Y$ (TM) neurons in which each cell has dimension $10 \times 10$) (c) bottom-up weights of two type neurons in $Z$ area ($1 \times 28$ $Z$ (TM) neurons in which each cell has dimension $20 \times 20$).

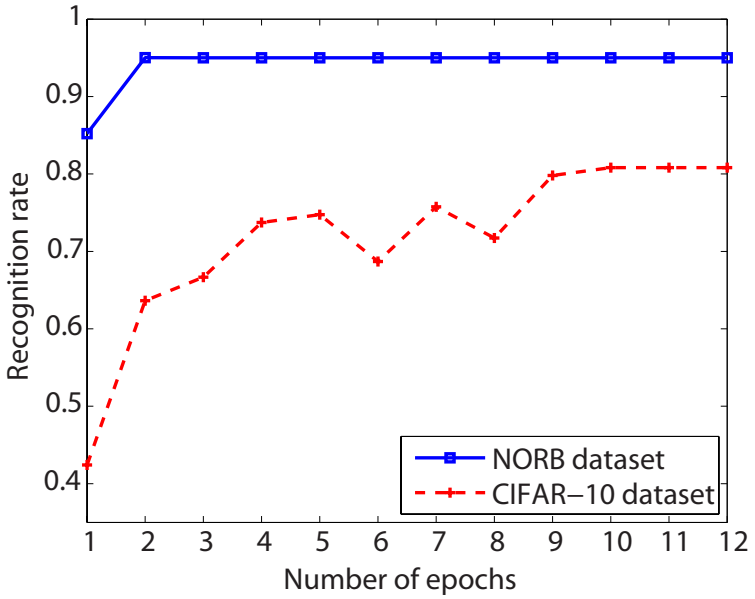
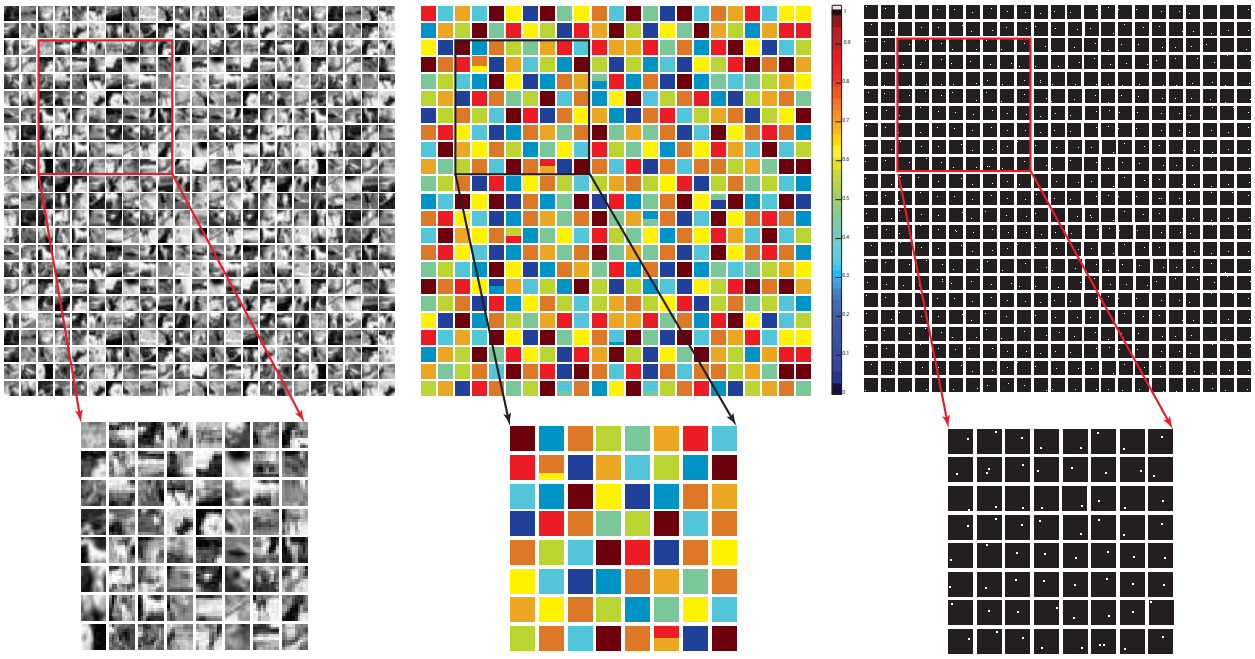



Figure 4.4: The variation number of epochs in training phase to gain maximum attainable recognition rate on NORB and CIFAR-10 datasets



(a) Bottom- up $Y$ (b) Top- down $Y$ (TM) (c) Top-down $Y$ (LM)

Figure 4.5: Visualization of weights in one depth (of 30 depths) in $Y$ area of CIFAR-10 (a) bottom-up weights ($23 \times 23$ $Y$ neurons in which each cell has dimension $10 \times 10$) (b) top-down weights (TM) ($23 \times 23$ TM neurons in which each cell has dimension $10 \times 10$) (c) top-down weights (LM) ($23 \times 23$ LM neurons in which each cell has dimension $23 \times 23$).

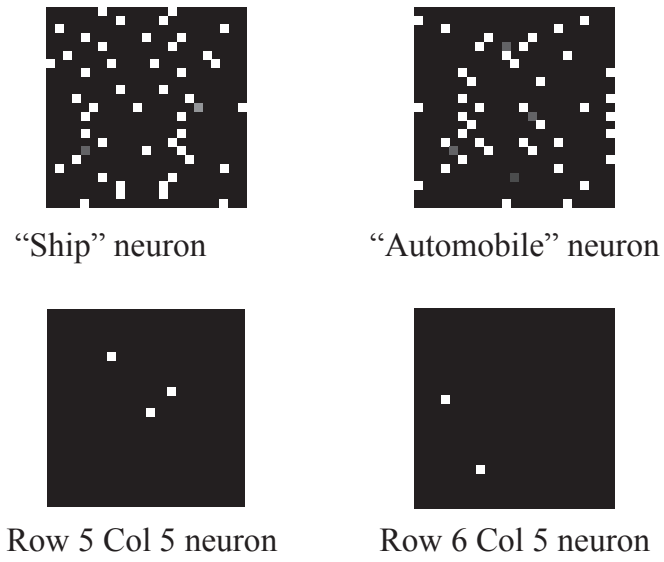



Figure 4.6: Visualization of the bottom-up weights for $Z$ area - two type neurons in one depth (of 30 depths) in TM ($1 \times 10$ TM neurons in which each cell has dimension $23 \times 23$) and two location neurons in LM ($23 \times 23$ LM neurons in which each cell has dimension $23 \times 23$).

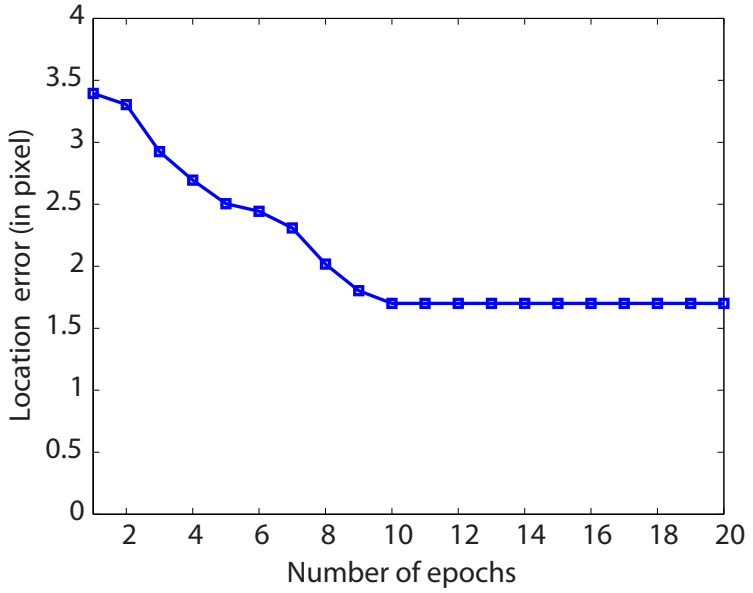



Figure 4.7: The number of epochs in training phase versus the location error (in pixels) in CIFAR-10 dataset.

# Chapter 5

# Conclusions

In this work, the DN algorithm has been compared with major well-known pattern recognition algorithms in global and local template based image matching problem. It showed that the performance of the DN method is better than, or comparable to, those global template based methods and the performance is comparable with those local template based methods.

In the local template based problems, DN allows different firing feature neurons to vote for a single neuron in TM and LM, which represent the type and location respectively. This experience-based voting has reached pixel-level location accuracy for CIFAR-10 dataset.

The work, here, indicates that the pay-off of finding the better spatial features seems to have diminished. The DN method tested in this work for space-only problem was meant for a tight integration of space and time information for visual events in natural cluttered backgrounds. In the work of [15, 16], it has been shown that using temporal information for spatial recognition in DN has considerably improved its recognition rate without imposing rigid constraints on object appearance through time which is typical with model-based object tracking.

A possible future direction of research is the detection of object contours from richly textured background — using a technique called synapse maintenance, as reported in [2]. The function of synapse maintenance in DN was not used for experiments here because the image datasets available in the public repositories like the ones we use here consist of snapshots of static scenes (static object fixed with a static background) which are very different from what human-eye sees from the dynamic physical world where the contours of unknown objects manifest themselves when an object moves relative with its cluttered backgrounds. This perspective indicates that learning directly from dynamic scenes can take into account information that has been overlooked by many publicly available computer vision data sets.

# BIBLIOGRAPHY

# BIBLIOGRAPHY


[1] J. Weng, and M. Luciw, Student Member, IEEE [2009] "Dually optimal neuronal layers: Lobe component analysis," IEEE Transactions On Autonomous Mental Development, Vol. 1, No. 1.

[2] Y. Wang, X. Wu, and J. Weng [2011] "Synapse maintenance in the 'where-what' networks," Proceedings of International Joint Conference on Neural Networks, San Jose, California, USA.

[3] A. Coates, H. Lee, and A. Ng [2011] "An analysis of single-layer networks in unsupervised feature learning," Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS), Fort Lauderdale, FL, USA, Volume 15 of JMLR: W&CP 15.

[4] D. Swets, and J. Weng [1996] "Using discriminant eigenfeatures for image retrieval," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 18.

[5] C. -W. Hsu, C. -C. Chang, and C. -J. Lin [2010] "A practical guide to support vector classication," Technical Report, National Taiwan University.

[6] Y. LeCun, F. Huang, and L. Bottou [2004] "Learning methods for generic object recognition with invariance to pose and lighting," IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR).

[7] A. Krizhevsky [2009] "Learning multiple layers of features from tiny images," Master's Thesis, Department of Computer Science, University of Toronto.

[8] J. Weng [2010] "A 5-chunk developmental brain-mind network model for multiple events in complex backgrounds," In Proc. Intl Joint Conf. Neural Networks, pages 18, Barcelona, Spain.

[9] J. Weng, and M. Luciw [2011] "Brain-like emergent spatial processing," IEEE Transactions on Autonomous Mental Development, Submitted and accepted.

[10] B. Olshaushen, and D. Field [1996] "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," Nature, Vol. 381, Pages 607-609.

[11] D. Fitzpatrick [2000] "Seeing beyond the receptive field in primary visual cortex," Current Opinion in Neurobiology, Vol. 10, No. 4, Pages 438-443.

[12] J. Weng, and M. Luciw [2006] "Optimal in-place self-organization for cortical development: Limited cells, sparse coding and cortical topography," Proc. 5th Int'l Conference on Development and Learning (ICDL'06), Bloomington, IN, Pages +1-7.

[13] M. Luciw, and J. Weng and S. Zeng [2008] "Motor initiated expectation through top-down connections as abstract context in a physical world," IEEE Int'l Conference on Development and Learning, Monterey, CA, Pages +1-6.

[14] J. Weng [2011] "Three theorems: brain-like networks logically reason and optimally generalize," Proc. Int'l Joint Conference on Neural Networks, San Jose, CA, Pages 2983-2990.

[15] M. Luciw, and J. Weng [2010] "Where what network 3: developmental top-down attention with multiple meaningful foregrounds" in Proc. International Joint Conference on Neural Networks, Barcelona, Spain, Pages 4233-4240.

[16] J. Weng [2011] "Why have we passed 'neural networks do not abstract well'?," Natural Intelligence: the INNS Magazine, Vol. 1, No.1, Pages 13-22.

[17] M. Luciw, and J. Weng [2009] "Laterally connected lobe component analysis: Precision and topography," Proc. IEEE 8th Int'l Conference on Development and Learning, Shanghai, China, Pages +1-8.

[18] M. Jordan, and C. Bishop [1997] "Neural networks," CRC Handbook of Computer Science, CRC Press, Boca Raton, FL, Pages 536-556.

[19] Z. Tu, X. Chen, A. Yuille, and S. -C. Zhu [2005] "Image parsing: Unifying segmentation, detection, and recognition," Int'l J. of Computer Vision, Vol. 3, No. 2, Pages 113-140.

[20] B. Yao, and L. Fei-Fei [2010] "Modeling mutual context of object and human pose in human-object interaction activities," Proc. Computer Vision and Pattern Recognition, San Francisco, CA, Pages +1-8.

[21] A. Gupta, and A. Kembhavi and L. S. Davis [2009] "Observing human-object interactions: Using spatial and functional compatibility for recognition," PAMI, Vol. 31, No. 10, Pages 1775-1789.

[22] R. Fisher [1936]. "The use of multiple measurements in taxonomic problems," Annals of Eugenics, Vol. 7, Pages179188.

[23] S. Mika, et al. [1999] "Fisher discriminant analysis with kernels," IEEE Conference on Neural Networks for Signal Processing IX, Pages 4148.

[24] R. Duda, P. Hart, and D. Stork [2000] "Pattern classification (second edition)," Wiley Interscience, New York.

[25] C. Cortes, and V. Vapnik [1995] "Support-vector networks," Machine Learning, Vol. 20, No. 3, Pages 273-297.

[26] T. Poggio, and G. Federico [1990] "Networks for approximation and learning," Proceedings of the IEEE, Vol. 78, No. 9, Pages14811497.

[27] P. Phillips, H. Moon, P. Rauss, and S. Rizvi [2000] "The FERET evaluation methodology for face recognition algorithms," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, No. 10.

[28] I. Goodfellow, Q. Le, A. Saxe, H. Lee, and A. Ng [2009] "Measuring invariances in deep networks," NIPS.

[29] H. Lee, A. Battle, R. Raina, and A. Ng [2007] "Efficient sparse coding algorithms," NIPS.

[30] H. Lee, C. Ekanadham, and A. Ng [2008] "Sparse deep belief net model for visual area V2," NIPS.

[31] J. Yang, K. Yu, Y. Gong, and T. Huang [2009] "Linear spatial pyramid matching using sparse coding for image classification," Computer Vision and Pattern Recognition.

[32] Y. Wang, X. Wu, and J. Weng [2011] "Brain-Like Learning Directly from Dynamic Cluttered Natural Video," Proceedings of the 18th international conference on Neural Information Processing.