11-27-2013

Hierarchical Feature Learning

Jayanta Kumar Dutta

HIERARCHICAL FEATURE LEARNING

by

Jayanta Kumar Dutta

A Thesis

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

Major: Electrical and Computer Engineering

The University of Memphis

December 2013

ACKNOWLEDGMENTS

Most of all, I would like to thank my advisor Dr. Bonny Banerjee for his continuous guidance from the very beginning. It has been a great privilege to work with and learn from him. He has been an amazing mentor and advisor, not only in research, but also other aspects of academic life.

I am grateful to the members of my thesis committee for their time and invaluable advice and constructive feedback to improve this thesis.

I am thankful to my colleagues at the CIL lab for their suggestions and discussions. I would also like to thank all of my friends and roommates at the UofM who made my stay here such a memorable one.

I gratefully acknowledge the funding support for this work from the Institute for Intelligent Systems in the form of a graduate assistantship and the Herff College of Engineering in the form of a fellowship.

Finally this thesis is dedicated to my parents and sister. Without their support, this thesis would not have been possible.

ABSTRACT

Dutta, Jayanta Kumar. MS. The University of Memphis. December 2013. Hierarchical feature learning. Major Professor: Dr. Bonny Banerjee.

The success of many tasks depends on good feature representation which is often domain-specific and hand-crafted requiring substantial human effort. Such feature representation is not general, i.e. unsuitable for even the same task across multiple domains, let alone different tasks.

To address these issues, a multilayered convergent neural architecture is presented for learning from repeating spatially and temporally coincident patterns in data at multiple levels of abstraction. The bottom-up weights in each layer are learned to encode a hierarchy of overcomplete and sparse feature dictionaries from space- and time-varying sensory data. Two algorithms are investigated: recursive layer-by-layer spherical clustering and sparse coding to learn feature hierarchies. The model scales to full-sized high-dimensional input data and to an arbitrary number of layers thereby having the capability to capture features at any level of abstraction. The model learns features that correspond to objects in higher layers and object-parts in lower layers.

Learning features invariant to arbitrary transformations in the data is a requirement for any effective and efficient representation system, biological or artificial. Each layer in the proposed network is composed of simple and complex sublayers motivated by the layered organization of the primary visual cortex. When exposed to natural videos, the model develops simple and complex cell-like receptive field properties. The model can predict by learning lateral connections among the simple sublayer neurons. A topographic map to their spatial features emerges by minimizing the wiring length simultaneously with feature learning.

The model is general-purpose, unsupervised and online. Operations in each layer of the model can be implemented in parallelized hardware, making it very efficient for real world applications.

TABLE OF CONTENTS

LIST OF FIGURES

Figures **Pages**

Chapter 1

Introduction

1.1 Representation and Learning Strategies in Multilayered Architectures

In recent years, there has been a surge of interest in learning feature hierarchies from data using deep architectures largely motivated by the layered organization of certain parts of the brain, particularly the neocortex. This interest is also fueled by a strong hypothesis – that the learning algorithms operating in the different perceptual cortices are very similar (Bach-y-Rita, 2004; Bach-y-Rita and Kercel, 2003; Constantine-Paton and Law, 1978; Metin and Frost, 1989; Mountcastle, 1978; Roe et al., 1992; von Melchner et al., 2000). George (George, 2008) observes that the common cortical algorithm hypothesis in conjunction with the No Free Lunch theorem (Ho and Pepyne, 2002) points toward a basic set of assumptions that are specific enough to make learning efficient while being general enough to be applicable to a large class of problems – the essence of intelligence.

Theoretical and empirical evidence shows that, unlike deep architectures, kernel methods (e.g., Support Vector Machine (Vapnik, 1998)) and other "shallow" architectures (e.g., neural networks with one hidden layer) are inefficient at representing complex functions involved in perception (Bengio, 2009). Deep architectures, such as convolutional neural networks (Farabet et al., 2011; LeCun and Bengio, 1995), HMAX (Riesenhuber and Poggio, 1999; Serre et al., 2007b), and deep belief networks (Hinton, 2007), recognize objects and actions with better accuracy than shallow architectures. In the rest of this section, we briefly review a few important issues related to representation and learning strategies in deep architectures.

1.1.1 Open-loop vs. Closed-loop Learning

Formulating an objective function helps to understand a model's global behavior. Two approaches to feature learning using deep models are prevalent – closed-loop (i.e., with feedback) and open-loop (i.e., without feedback). In the former, an objective or

energy function is minimized by iteratively updating connection weights with respect to an error signal that is propagated backwards. In case of supervised learning, this signal is often the gradient of the classification error which results in learning discriminative features while in the unsupervised case, it is the gradient of the reconstruction error which results in learning generative features. An appropriate regularization term is often used to avoid overfitting and induce sparsity. Variants of deep belief networks (Hinton et al., 2006), convolutional neural networks (LeCun et al., 1998; Lee et al., 2009), and sparse/denoising autoencoders (Vincent et al., 2008) are trained using this approach (also see (Larochelle et al., 2009; Ranzato et al., 2007)). One useful objective function for unsupervised learning is the l_2 norm of the reconstruction error,

$$\mathcal{E}(\mathtt{X}, \mathtt{D}) \equiv \frac{1}{2}\|\mathtt{X} - \mathtt{D}\alpha\|_2^2 \tag{1.1}$$

where $\mathtt{X}$ are the signals, $\mathtt{D}$ is an overcomplete dictionary of non-orthogonal bases or features, and α are the coefficients. Minimization of $\mathcal{E}$ subject to $\|\alpha\|_0 < n$, where $\|.\|_0$ is the l_0 norm and n is an integer (small relative to the number of features), allows learning $\mathtt{D}$ using which a sparse representation of the input is possible (Aharon et al., 2006). Non-orthogonality and overcompleteness of features leading to sparse coding have been claimed to explain certain observations in the response properties of cortical cells (Olshausen and Field, 1996).

In the open-loop approach, unsupervised learning can be conceptualized as capturing the distribution of recurring coincident patterns in the data by a feedforward mechanism, i.e. an explicit feedback of the error signal is absent. Clustering is an example of this approach though not the only one. Variants of Neocognitron (Fukushima, 1980; Fukushima, 1988; Fukushima, 2003), HMAX (Riesenhuber and Poggio, 1999; Serre, 2006; Serre et al., 2007a; Serre et al., 2007b) and Hierarchical Temporal Memory (HTM (George and Hawkins, 2005; George, 2008)) are trained using this approach.

1.1.2 Linear vs. Part-based Representation

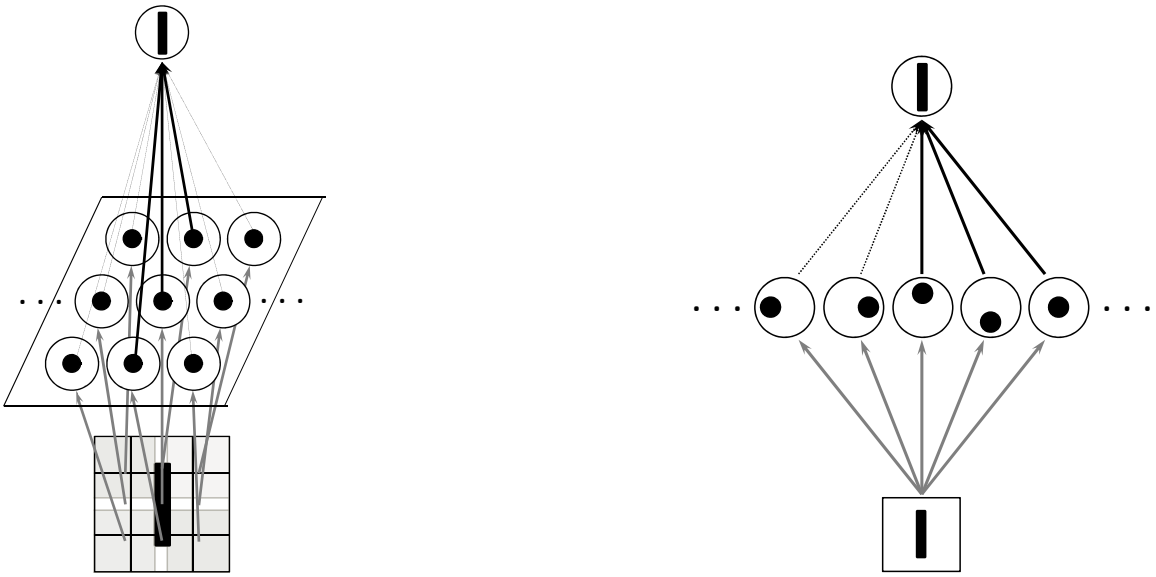
In hierarchical networks, complex RF structures or features in higher layer neurons can be learned from simpler features in lower layer neurons in at least two apparently different ways – by the principle of spatial organization that follows from the seminal work of Hubel and Wiesel (Hubel and Wiesel, 1962; Hubel and Wiesel, 1965; Hubel and Wiesel, 1968), and by the principle of linear superposition that is utilized widely in machine learning applications with impressive results (Hinton et al., 2006; Vincent et al., 2008). See (Martinez and Alonso, 2003) for a review on this issue.

In the case of spatial organization, neurons are arranged in a 2D grid. Each neuron receives input from a unique region in space. Two or more neurons might have some overlap (less than 100%) in their inputs. The physical size of RFs increases as we ascend up the hierarchy. A higher layer feature is learned by generating strong connections with a subset of neurons in the lower layer, the subset is determined by the input data.

In the case of linear superposition, a signal $\mathtt{x}$ is modeled as a linear combination of features in a dictionary $\mathtt{D}$, i.e. $\mathtt{x} = \mathtt{D}\alpha$, where each feature in $\mathtt{D}$ has the same dimensions as $\mathtt{x}$. Since a feature is encoded in the RF of a neuron, all neurons receive input from the same region in space. Therefore, all neurons always have 100% overlap in their inputs. The physical size of RFs remain the same throughout the hierarchy. As in the case of spatial organization, a higher layer feature is learned by generating strong connections with a subset of neurons in the lower layer, the subset is determined by the input data. Fig. 1.1 illustrates the two principles using a caricature of center-surround RFs in the lower layer and a simple RF in the higher layer.

We observe that these two feature representations are functionally similar. In the case of spatial organization, arrangement of neurons in a 2D grid along with lateral connections allow the layer of neurons to encode the relative spatial location of a feature. The same information is encoded within the RF of a neuron in the case of linear superposition, hence they are not required to be arranged in a 2D grid. Also, they do not

require lateral connections to encode their relative locations. However, they are more rigid in the sense that if the parts remain the same but their relative locations change, new features will have to be learned. In the case of spatial organization, only the weight of lateral connections need to be altered.



(a) (b)

Fig. 1.1: Two ways of learning features by combining lower level features. (a) The center surround features in the lower layer have the same location within the RF. The higher layer feature, a vertical bar, is learned by selectively superimposing with some degree of overlap (less than 100%) the lower layer features. The selected neurons have stronger connections to the higher layer neuron. RF size increases as we ascend up the layers. (b) The center surround features in the lower layer have different locations within the RF. The higher layer feature is learned by selectively superimposing with 100% overlap the lower layer features. The selected neurons have stronger connections to the higher layer neuron. RF size remains constant across layers.

In the design of our architecture, we take a hybrid approach. A node, consisting of a set of neurons with different RFs, receives input from a unique location in space. Different nodes receive inputs from different spatial locations with some degree of overlap (less than 100%). This is discussed in Section 2.1. In any layer in our architecture, nodes are arranged in a 2D grid and connected via lateral connections (spatial organization) while neurons in a node are arranged in a topographic map which minimizes the wiring

length (Dutta and Banerjee, 2013; Hyvarinen and Hoyer, 2001; Kavukcuoglu et al., 2009) and their RFs are learned using linear superposition.

1.1.3 Discriminative vs. Generative Models

Given an input data x for classification, the goal for discriminative hierarchical neural models is to infer its class or label y encoded in the highest layer representation. In probabilistic terms, a discriminative model learns the posterior $p(y|x)$ directly which is a mapping from the data to the class labels. In contrast, the goal for generative models is to generate or reconstruct the data at the output as faithfully as possible. A generative model learns the joint probability $p(x, y)$ of the inputs x and the label y, computes $p(y|x)$ using Bayes rule, and selects the most likely label y for classification. In the task of classification, a number of studies have been shown that both models have their own regimes of performance in which each of them does better (Long et al., 2007; Ng and Jordan, 2001). When the training set is small, classifiers based on generative models outperform discriminative classifiers (Schmah et al., 2009).

1.2 Invariant Representation Learning

Learning features invariant to arbitrary transformations in data is a requirement for any biological or artificial recognition system. A number of computational models have been proposed that can learn transformation-invariant features for state-of-the-art recognition in images, audio and videos using alternating simple and complex layers. The simplest way is to put built-in invariances directly like translation invariance in the architecture. Convolutional neural network (LeCun et al., 1989), SIFT descriptors (Lowe, 2004) use this approach. But it only works for known invariances and can not represent unknown invariances. Another way to represent invariance is to learn topographic filter maps in the simple cell layer (Hyvarinen and Hoyer, 2001; Kavukcuoglu et al., 2009; Mairal et al., 2011). In this approach, similar features are closed by in the filter map and invariance can be achieved by pooling units close in space together. It can be also done by group sparse coding (Garrigues and Olshausen, 2010).

Another approach is to use the temporal coherence in the data to learn transformation invariance. It is believed that the inputs are more likely a consequence of the same cause if they follow one another in time (Gregor and LeCun, 2011). Invariance can be achieved by discovering the cause that is same for all those inputs. There are several methods, which use this idea. In (Földiák, 1991) a trace rule has been used which will give some benefit to fire the same complex cell in close temporal proximity. (Einhäuser et al., 2002) learns the correlation between strong presynaptic activity precedes by strong postsynaptic activity and (Masquelier et al., 2007) learns the correlation between currently most activated simple unit and previously most activated complex unit to learn invariances. Slow feature analysis (Bergstra and Bengio, 2009; Berkes and Wiskott, 2005; Wiskott and Sejnowski, 2002) forces the representation to change slowly. Temporal product network (Gregor and LeCun, 2010) breaks the input into two representations, one that is common to all frames and one that is complementary. (Cadieu and Olshausen, 2008) learns higher-order structure among the time-varying phase variables. HTM (George and Hawkins, 2009) forms groups based on transition matrix between states. (Gregor and LeCun, 2011) adds up the coefficients of sparse coding units over a fixed amount of time and derives the cause for that cumulative vector.

1.3 Overview of Contributions

The main contributions of this thesis can be given as follows:

- A hierarchy of features are learned using a multilayered neural network architecture. Two algorithms are investigated for the purpose – spherical clustering with an adaptive threshold and sparse coding. The threshold, unique for each neuron, does not allow outliers to affect the cluster centers. Experimental results show that reconstruction capability is better for sparse coding than clustering, classification accuracy is comparable for both algorithms, reconstruction/denoising of input from a higher layer is better than that from a lower layer, and classification accuracy obtained from a lower layer is better than a higher layer.

- Arbitrary transformations from spatiotemporal data are learned using a two layered neural network architecture. Two algorithms are investigated – temporal spherical clustering with variable receptive field size and non-negative matrix factorization. The former learns transformations of arbitrary lengths while the lengths are fixed for the latter. A topographic structure in the feature layer is learned by exploiting temporal coherence in the data without assuming any predefined pooling range.

1.4 Outline

This thesis will proceed as follows:

Chapter 2 will cover the architecture of the hierarchical neural network that we will follow throughout the thesis. We will briefly describe the use of different types of connections (e.g. feedforward, lateral, feedback) as well as the notations.

Chapter 3 will present hierarchical feature learning from sensory data. Two different algorithms for learning feature hierarchies will be investigated. In the first one, the bottom-up weights in each layer are learned to encode a hierarchy of overcomplete and sparse feature dictionaries from space- and time-varying sensory data by recursive layer-by-layer spherical clustering. This density-based clustering algorithm ignores outliers by the use of a unique adaptive threshold for each neuron. The model is fully-learnable with only two manually tunable parameters. The second one learns to encode features using sparse coding. The model scales to full-sized high-dimensional input data and also to an arbitrary number of layers thereby having the capability to capture features at any level of abstraction. Some differences between spherical clustering and sparse coding will be shown.

Chapter 4 will show how to learn invariance to arbitrary transformations using temporal spherical clustering and non-negative matrix factorization. It will be shown that the model develops complex cell-like receptive field properties in primary visual cortex.

1.5 First Published Appearances

Most of the work presented in this thesis have first appeared as various publications. The following list describes the representative publications roughly corresponding to each chapter in this thesis:

- Chapter 2: (Banerjee and Dutta, 2013c; Dutta et al., 2012)
- Chapter 3: (Banerjee and Dutta, 2013a; Banerjee and Dutta, 2013b; Banerjee and Dutta, 2013c)
- Chapter 4: (Dutta and Banerjee, 2013)

Chapter 2

Network Architecture

2.1 Nodes as Canonical Computational Units

Our network architecture (Banerjee, 2012; Dutta et al., 2012) consists of a hierarchy of layers of nodes (see Fig. 2.1). Each layer is composed of a simple and a complex sublayers. A node is a canonical computational unit consisting of a lamina of densely connected integrate-and-fire neurons (see Fig. 2.2) that is replicated throughout the architecture. Research in sensory systems gives strong indications that the brain applies similar computations to different problems, and has thus identified a number of these canonical computations which have proven capable of accounting for a wide variety of observed neurophysiological measurements. See, for example, (Cadieu et al., 2007; David et al., 2009; Douglas and Martin, 2010; Kouh and Poggio, 2008; Rust et al., 2005; Rust and DiCarlo, 2008). Our architecture consists of alternating simple and complex sublayers, starting with a simple sublayer, a strategy employed by a number of hierarchical neural models, such as the Neocognitron (Fukushima, 1980; Fukushima, 1988; Fukushima, 2003), HMAX (Riesenhuber and Poggio, 1999; Serre, 2006; Serre et al., 2007a; Serre et al., 2007b) and convolutional neural networks (Farabet et al., 2011; LeCun and Bengio, 1995).

2.2 Receptive Fields

A neuron in a node is connected to the neurons in the neighboring nodes in the same layer, one layer above, and one layer below by lateral, feedforward, and feedback connections, respectively. The first (or lowest) layer in the hierarchy receives input from external data (e.g., images, videos, audios). The RF of each neuron is spatiotemporal; that is, the RF has a spatial and temporal extent within which the neuron integrates information. The size of a RF is the same for all neurons in a layer and increases as we ascend up the hierarchy. If a neuron is excited enough by integrating activations over its

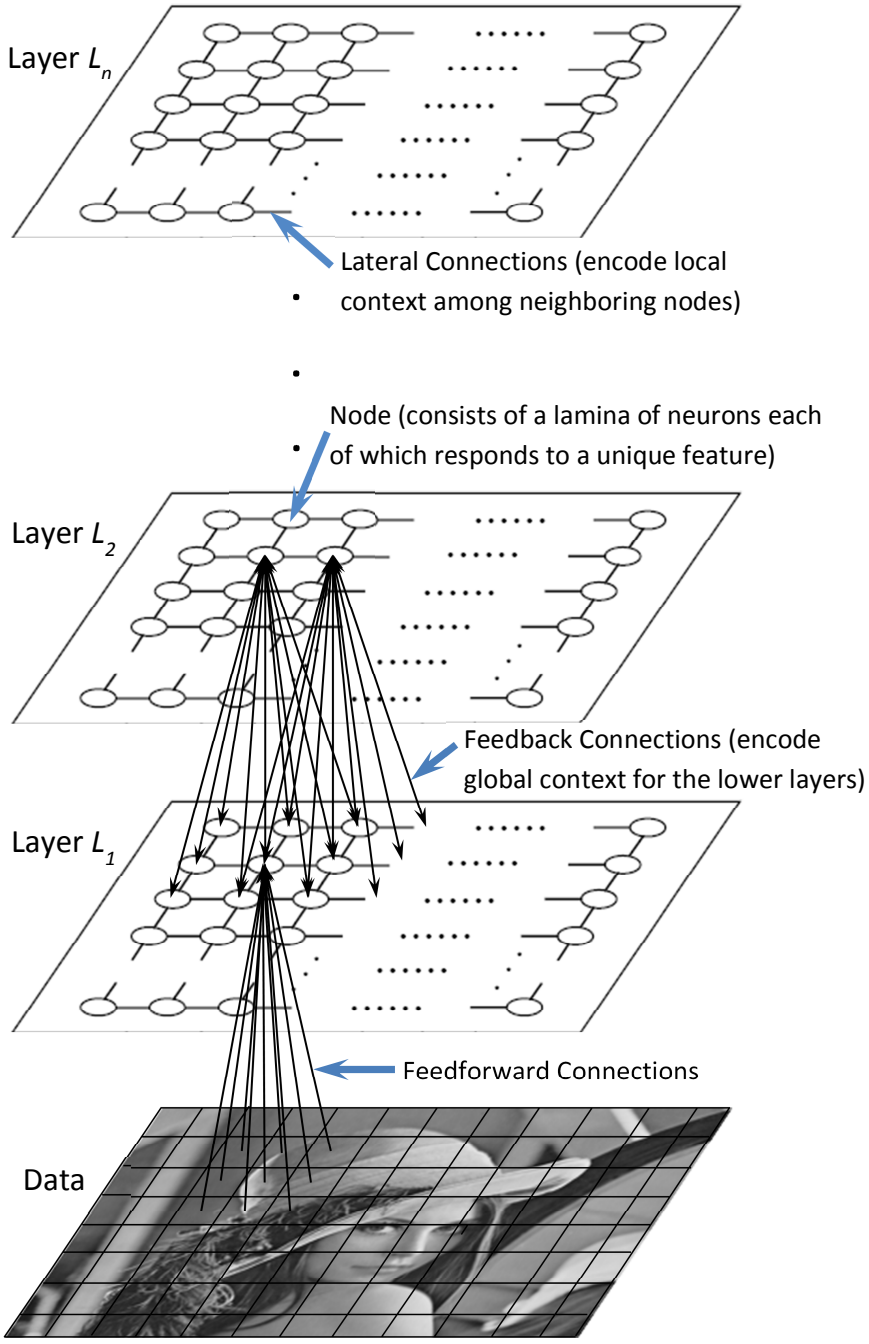


Fig. 2.1: The neural network architecture used to implement our model. Each layer L_i is a pair of simple and complex sublayers. Circles denote nodes. Inter-node lateral connections encode spatial correlations.

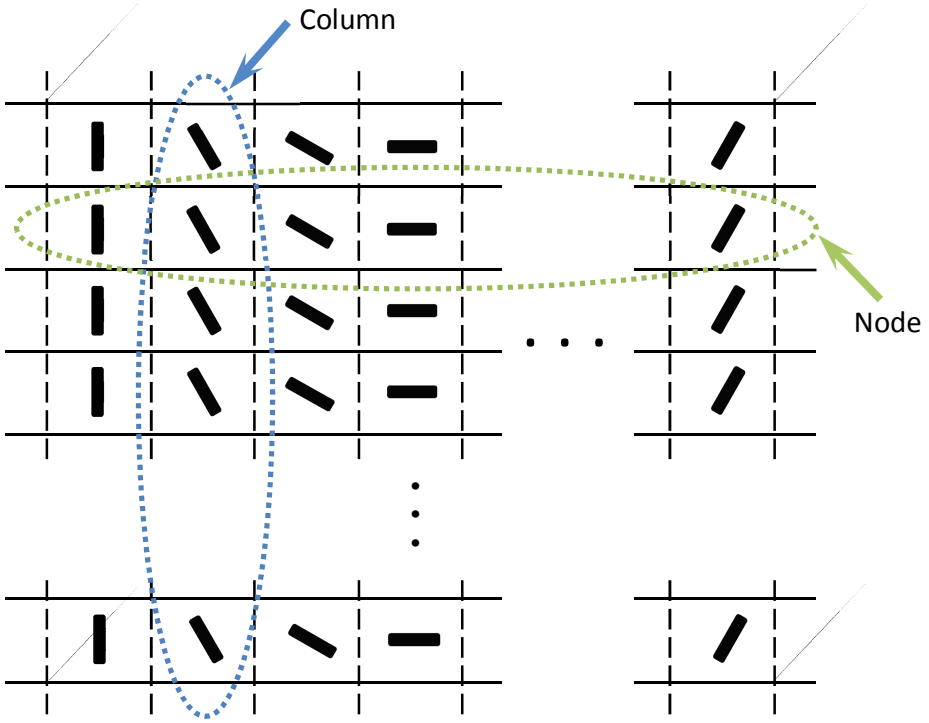
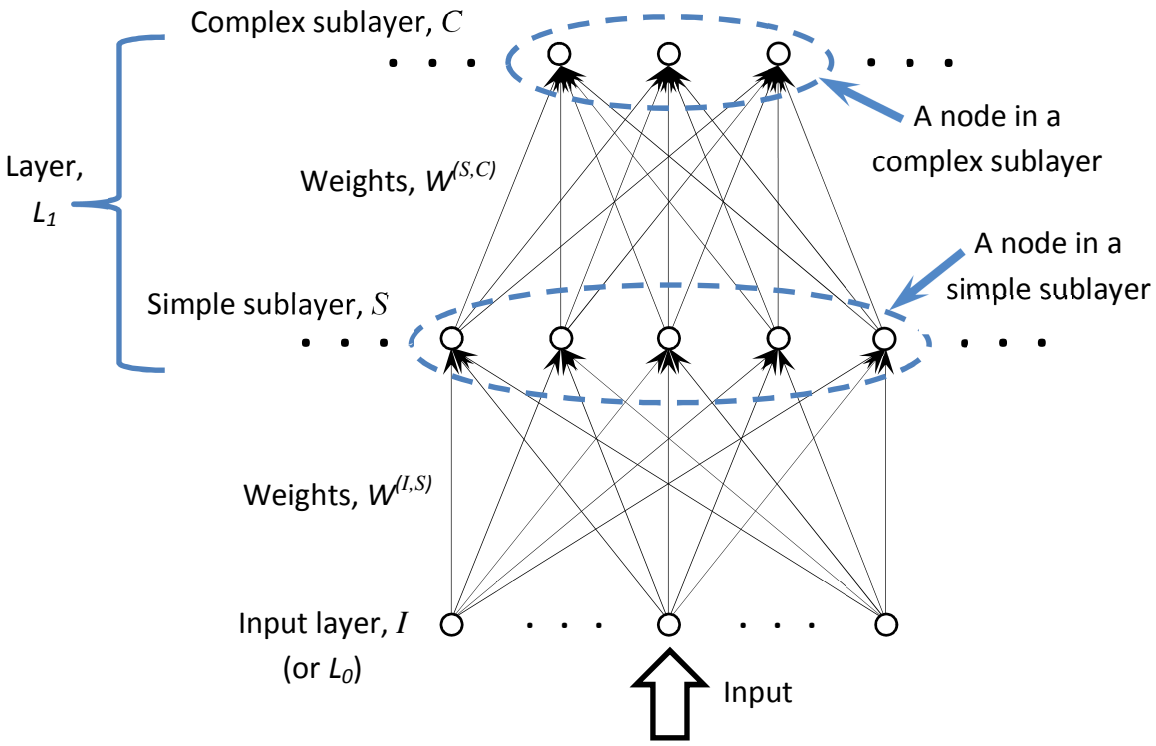
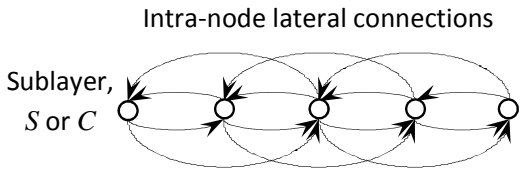


Fig. 2.2: Nodes and columns in our architecture. A node is a lamina of neurons (shown here in one-dimension) each of which responds to a unique feature. A column consists of neurons all of which respond to the same feature. This is conceptually similar to the ice-cube model of primary visual cortex (Hubel and Wiesel, 1977). Such a cube of nodes and columns forms a simple sublayer in our architecture as shown in Fig. 2.3.



(a)



(b)

Fig. 2.3: One layer in our architecture. (a) Feedforward connections from a simple to a complex sublayer node. Circles denote neurons. $W^{(I,S)}$ are learned to encode spatial sets or features in simple sublayer S. $W^{(S,C)}$ are learned to encode temporal sets or transformations in complex sublayer C. Feedback connections are not shown. (b) Lateral connections in S within a node. Intra-node lateral connections encode temporal correlations. These lateral weights $W^{(S,S)}$ in conjunction with $W^{(S,C)}$ are modeled to learn sequences.

spatiotemporal RF, it spikes.[1] It functions as an integrate-and-fire neuron (Abbott, 1999) and a suspicious coincidence detector (Földiák, 1990).

At each sampling instant, our model accepts spatial data as input through the first layer, which is passed on to higher layers in the form of activations. The goal of computations in each node is to explain or reconstruct the input. Over time, each neuron in a node gets tuned to a unique feature; such a sparse set of features can reconstruct many different inputs. Functionally, a node is an overcomplete set of filters, all of which are applied to each patch of the input data. These filters might be spatial (e.g., edges in different orientations) or spatiotemporal (e.g., vertical edge moving in a particular direction) depending on the temporal RF size of the corresponding neurons. For learning spatial features with no temporal component, the temporal RF size of the neurons may be set to unity. Such neurons are referred to as *simple neurons*. *Complex neurons* learn spatiotemporal features with a finite temporal RF size greater than unity and the same spatial RF size as its lower layer simple neurons. As a result of learning multiple layers of features, where each layer treats the activations of the lower layer neurons as data, strong connections are formed from the first layer neurons to the top layer neurons through the intermediate layers such that rapid categorization of the input signal may be achieved (Serre et al., 2007a). The connections that are used more often are strengthened while the rest are weakened.

In the most general case, the neurons in layer $\ell + 1$ in our architecture have larger RFs, both in space and time, than those in layer ℓ. In accordance with the structure of the visual pathway, researchers have opted to design multilayered neural architectures with alternating layers of simple and complex cell-like neurons where the simple neurons respond to spatial features (e.g., edges in different orientations) while the complex neurons induce scale and translation invariance to those features. Examples of such models include the Neocognitron, convolutional neural networks, HMAX and HTM. In

[1] In this article, we account for every spike of a neuron as opposed to the spiking/firing rate or any function of that (e.g., mean) or their distribution of a single or population of neurons.

these models, connections from a simple to complex layer are hardwired for pooling and only the connections to the simple layer are learned. In our model, all connections are learned from the data and thus can afford the flexibility to learn spatiotemporal features of sizes driven by the data. Further, our architecture may be used with minimal modification to learn spatial features (e.g., from images), temporal features (e.g., from audio), or spatiotemporal features (e.g., from videos).

2.3 Connections: Feedforward, Lateral and Feedback

Connections across layers are of two types – feedforward and feedback. Feedforward connections help higher layer neurons to abstract more stable spatiotemporal patterns by pooling from a number of lower layer neurons. This strategy has been used in many multilayered networks, such as HMAX (Riesenhuber and Poggio, 1999; Serre, 2006; Serre et al., 2007a; Serre et al., 2007b), HTM (George and Hawkins, 2005; George, 2008; Hawkins et al., 2011), convolutional neural networks (Farabet et al., 2011; LeCun and Bengio, 1995), and deep belief networks (Hinton et al., 2006; Hinton and Salakhutdinov, 2006; Hinton, 2007). The pooling mechanism has been shown to capture invariances to arbitrary transformations implicit in the data (Dutta and Banerjee, 2013). Top-down feedback connections predict global spatiotemporal patterns, that is, over a larger space and time. The strength of connections encode the recurring local correlations (or lack thereof) in neural spikes.

Lateral connections within a layer are of two types: those that connect neurons within a node (intra-node) encode temporal correlations while those that connect neurons across neighboring nodes (inter-node) encode spatial correlations. Spatial correlations have to be stored in inter-node lateral connections as each node looks at a particular region in space. Temporal correlations have to be stored in intra-node lateral connections as activations of neurons within a node over time depict how a feature changes in a particular region in space over time.

In our mdoel, a simple neuron strongly connects to a set of complex neurons in the lower layer, encoding a set of features in space. The relative spatial locations of these features, arranged in a 2D grid, are encoded as spatial correlations by inter-node lateral connections in the complex layer. These connections are undirected. A complex neuron strongly connects to a set of simple neurons in the lower layer, encoding a set of features in time. The sequence of occurrence of these features are encoded as temporal correlations by intra-node lateral connections in the simple layer. The direction of such a connection signifies the direction of transition in time. Thus, lateral connections provide spatial and temporal structure to the sets encoded by feedforward connections. Without these lateral connections, detection of features in the input would be possible but not their relative locations in space or time.

Notation. $\mathcal{N}^{(\ell)}(i)$ is the set of neurons in layer ℓ that connect to the i^{th} neuron in some layer. This is also referred to as the *neighborhood* in layer ℓ of the i^{th} neuron. $W_{ji}^{(k,\ell)}(t)$ is the weight or strength of connection from the j^{th} neuron in layer k to the i^{th} neuron in layer ℓ at time t. $A_i^{(\ell)}(t)$ and $S_i^{(\ell)}(t)$ are respectively the activation and state of the i^{th} neuron in layer ℓ at time t. Finally, τ_i^{ℓ} is the temporal RF size of the i^{th} neuron in layer ℓ. $\mathtt{M}^{\mathtt{T}}$ denotes transpose of matrix $\mathtt{M}$.

In the next two chapters, we will describe the learning procedure for simple and complex neurons. We will also show the use of different types of connections (e.g., feedforward, lateral and feedback) in learning those simple and complex neurons.

Chapter 3

Hierarchical Feature Learning from Sensory Data

3.1 Feature Learning using Spherical Clustering

3.1.1 Objective Function

The model described in this section learns feature hierarchies from recurring coincidences in the data in an unsupervised and online manner, minimizing the following objective function on convergence:

$$\ell(\mathcal{X}, \mathcal{W}) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j \in \mathcal{N}(i)} \|x_j - w_i\|^2 \tag{3.1}$$

where $\mathcal{X} = \{x_1, x_2, ...x_N\}$ and $\mathcal{W} = \{w_1, w_2, ...w_n\}$ are the set of d-dimensional data points and features respectively, $\mathcal{N}(i)$ is the set of data points in the neighborhood of w_i, $|\bigcup_{i=1}^{n} \mathcal{N}(i)| < N$, $|.|$ denotes the cardinality of a set. Each data point and feature is normalized to have unit norm. Each layer in our model learns a set of non-orthogonal features that soft-partitions a subset of the normalized input space; this subset, given by $\bigcup_{i=1}^{n} \mathcal{N}(i)$, does not contain outliers. Such a formulation may be construed as soft-clustering on the surface of a hypersphere of unit radius (a.k.a. *spherical clustering* (Dhillon and Modha, 2001)) where the outliers are not allowed to influence the cluster centers.

3.1.2 Architecture

In this section, we will concentrate on learning feature hierarchies using the feedforward connections and simple sublayers only.

3.1.3 Operation

At each sampling instant, our model accepts spatial data as input through the first layer which is passed on to higher layers in the form of activations. The goal of computations in each node is to selectively cluster the data into groups (Dutta and Banerjee, 2013). Over time, each neuron in a node gets tuned to a unique feature which

represents a cluster center. Functionally, a node is a bag of filters all of which are applied to each patch of the input data.

The output (or state) of a layer is the input to the next higher layer. The same operation is executed in each node in any layer. Thus, the feedforward weights in this hierarchical model are learned by recursive layer-by-layer spherical clustering.

3.1.4 Neuron

In our model, the activation of a neuron is given by

$$A_i^{(l)}(t) = \sum_{j \in \mathcal{N}^{(l-1)}(i)} W_{ji}^{(l-1,l)}(t) \times A_j^{(l-1)}(t) \tag{3.2}$$

In matrix form, $A^{(l)} = A^{(l-1)} \times W^{(l-1,l)}$ where the neighborhood information is implicit. Since each feature in $W^{(l-1,l)}$ and $A^{(l-1)}$ are normalized, $A^{(l)}$ is the normalized dot product of the input with each feature. This allows a neuron to act as a suspicious coincidence detector (Földiák, 1990), responding with high activation if the input pattern matches the feature encoded in its receptive field.

For a given input, all neurons in a node receive activations. The maximally activated neuron in a node is the "winner". While we compute the winner using a max operation, it is more biologically plausible to consider lateral connections within a node using which neurons inhibit each other at a faster time scale eventually settling at some stable state. Lateral inhibition has been used for similar purposes in many models, such as, in (Einhäuser et al., 2002), in the form of V-cells in Neocognitron (Fukushima, 2003) and in the LISSOM model (Sirosh and Miikkulainen, 1997).

The state of a neuron is binary and is given by

$$S_i^{(l)}(t) = \begin{cases} 1, & \text{if } A_i^{(l)}(t) > A_j^{(l)}(t), \forall j \neq i, \text{ and} \\ & \quad A_i^{(l)}(t) > \theta_i^{(l)}(t) \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

The threshold θ is adaptive and unique for each neuron. Only the winner in a node is

assigned the state 1 if its threshold is crossed. This is how our model implements the winner-take-all mechanism which allows only the neuron of highest activity to learn. We say a neuron has *fired* if its state reaches unity.

Thus, a neuron integrates all inputs over its RF until it reaches its threshold when it fires if it is the winner. As soon as it fires or if it fails to fire, it discharges and then starts integrating again. The discharge from a neuron inhibits neighboring neurons in its own layer. As in (Einhäuser et al., 2002), it may be assumed that this lateral inhibition is proportional to a neuron's total accumulated charge (or activation) and operates at a faster time scale. The inhibition is required to ensure that neurons in a layer do not get tuned to the same feature. The inhibition influences a neuron's activation which in turn influences its inhibition. This cycle ensues until a stable state is reached. In most practical cases, this inhibition is observed to be strong enough to drive all neurons close to their baseline activation. In our implementation, we assume this baseline to be zero which does not affect our features qualitatively.

3.1.5 Learning: Updating Weights and Thresholds

Feedforward weights to neuron j in layer l with $S_j^{(l)}(t) = 1$ are updated following Hebbian rule.

$$W_{ij}^{(l-1,l)}(t+1) = (1-\alpha) \times W_{ij}^{(l-1,l)}(t) + \alpha \times S_i^{(l-1)}(t) \tag{3.4}$$

where α is the learning rate that decreases with time for finer convergence, $0 < \alpha < 1$, $S^{(0)} = A^{(0)}$. This weight update rule is obtained by applying gradient descent on the objective function in eq. 3.1 in an online setting. Feedforward weights leading to each neuron are initialized to ones and normalized to have unit norm, which allows all neurons in a layer to compete on an equal footing. A new neuron is not recruited unless the incoming pattern is more similar to the initialized feature than to any of the learned features. After each update, weights to each neuron are normalized to have unit norm. Thus, feedforward connection from a presynaptic neuron (i) to a postsynaptic one (j) that

fire together are strengthened while the rest (to j) are weakened. The weakening of connections is crucial for robustness as it helps remove infrequent coincident patterns from memory which are probably noise.

The threshold is updated as follows:

$$\theta_i^{(l)}(t+1) = \begin{cases} A_i^{(l)}(t), & \text{if } S_i^{(l)}(t) = 1 \\ (1-\eta) \times \theta_i^{(l)}(t), & \text{otherwise} \end{cases} \tag{3.5}$$

where η is the threshold decay parameter, a constant, $0 < \eta < 1$. Due to the threshold, only a subset of stimuli can trigger learning. If $\eta = 1$, all stimuli are used in learning as in traditional clustering algorithms. If $\eta = 0$, no stimulus can cross the threshold, hence learning does not occur. Size of the set of effective stimuli reduces with reduction in the value of η. The threshold decay mechanism ensures that the size of the effective subset remains fixed throughout the learning process, thereby maintaining the plasticity of the network. The winner-take-all mechanism along with the threshold favor neurons with sparsely distributed activity.

In the proposed model, a neuron always passes on its activations to its neighboring neurons in all layers irrespective of whether it fires or not. This is crucial for online operation where learning and inferencing proceed simultaneously and not in distinct phases. If a pattern has been learned and a part of it is shown, a partial pattern of activations will stimulate the remaining neurons of the pattern to become active thereby completing the whole pattern. However, the strength of connections will not be altered unless enough of the pattern has been seen (as determined by θ) and the RFs of the presynaptic neurons are the best match to the incoming pattern in their respective nodes to fire the postsynaptic neuron in the higher layer.

3.1.6 Experimental Results

The proposed model in this section was deployed for learning feature hierarchies from data in different modalities in an unsupervised and online manner with the learning

rule derived from the same objective function as in eq. 3.1. The feedforward weights were learned layer by layer with $\alpha(t) = \alpha(t-1)/(1+t/10^6)$, $\alpha(0) = 0.1$. θ were initialized to ones. Overlap between patches for adjacent nodes was 75% and 25% in the first and second layers respectively. The top layer had only one node. The number of nodes in each layer is a function of the $\%$ overlaps and the RF sizes of neurons.

Features for the second and third layers were reconstructed as follows. For a neuron in the second layer, a neuron in each first layer node that most strongly connected to it was chosen. The features represented by these neurons were weighed by the connection strengths and spatially organized taking into consideration the % overlaps among nodes. Once the second layer features were constructed, the same procedure was carried out for each third layer neuron to construct their features. To reconstruct unknown data, a winner neuron was computed in each node in the highest layer. A neuron in each node in the lower layers was chosen based on strongest connection to the winner. The chosen lowest layer features, each multiplied by the norm of the corresponding input data patch and spatially organized based on the % overlaps among nodes, reconstructed the input.

Images

Our model learned three layers of features from natural images (downloaded from Google images). The images were converted to grayscale, and convolved with a Laplacian of Gaussian filter to crudely highlight edges (performed by center-surround cells before the signal reaches V1). The features learned in the first layer were edges/bars in different orientations and phases, similar to RFs of simple cells in V1 (Hubel and Wiesel, 1962). The features learned in the second layer were different combinations of these edges, similar to RFs found in V2 (Hegde and Van Essen, 2000; Ito and Komatsu, 2004) (see Fig. 3.5). Features learned in the third layer were unstable and did not show any coherent pattern. Our model also learned three layers of features from 60,000 images of ten handwritten numerals $\{0, 1, ...9\}$ from the MNIST dataset (LeCun et al., 1998). As shown

in Fig. 3.2, parts of numerals were learned in first layer, larger parts in the second layer, and whole numerals in the third layer. The grayscale intensity denotes the strength of a feature. η was chosen as 10^{-4} for natural images and 10^{-1} for MNIST as there are many more outliers in the former data set compared to the latter. Thus, the same model could learn three layers of features from the MNIST data but only two layers from natural images due to the absence of recurring coincidences among second layer features in the latter case.

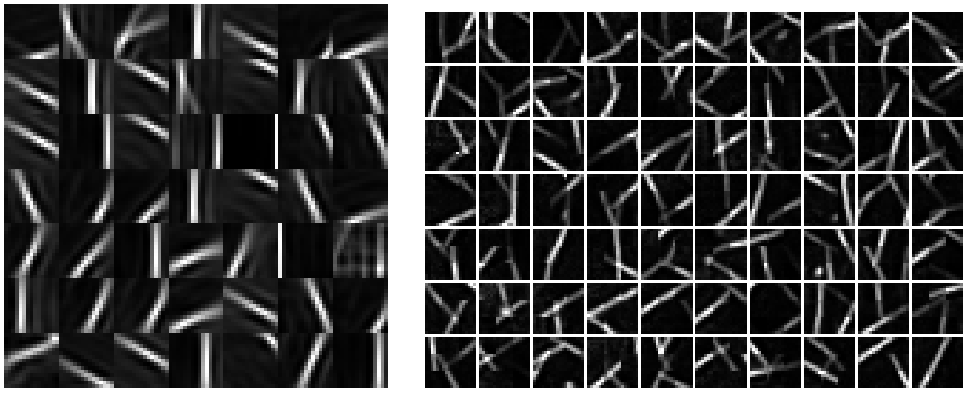


Fig. 3.1: Features of size 10×10 and 20×20 were learned from natural images in first (left) and second layers (right). 49 out of 150 and 70 out of 100 features from first and second layers are shown.

Videos

Spatiotemporal video features were learned in our model from 3D voxels where time is the third dimension. Such features have often been learned from voxels for computer vision and machine learning applications, particularly for action recognition (see for example, (Ji et al., 2010; Le et al., 2011)). When our model was exposed to videos of ten actions (e.g., walking, waving) performed by nine subjects from the Weizmann dataset (Gorelick et al., 2007) with $\eta = 10^{-2}$, the first layer neurons with RF size $10 \times 10 \times 5$ learned edges in different orientations and moving in different directions. That is, they developed orientation- and direction-selective RFs as in complex cells in V1 (Hubel and Wiesel, 1962) (see Fig. 3.3). Consequently, they respond to static edges/bars

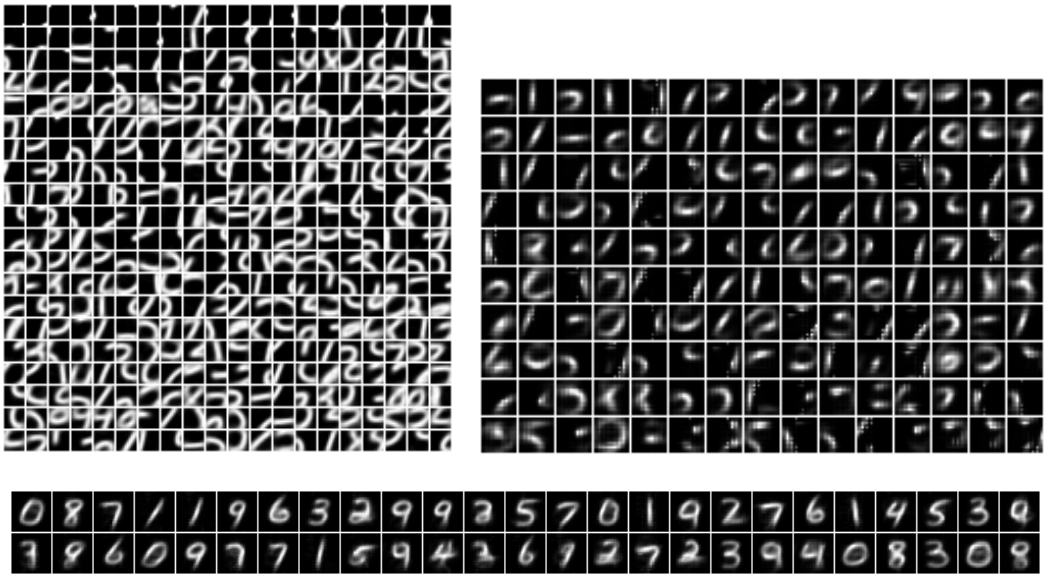


Fig. 3.2: A hierarchy of features were learned from handwritten numerals in MNIST dataset in first, second and third layers with receptive field sizes 10×10, 16×16 and 28×28 respectively. 400, 150 and 50 features from first (top left), second (top right) and third (bottom) layers are shown.

in a particular orientation in different locations within their RFs, and therefore, have learned position-invariant features.

Clustering

Our learning algorithm may be construed as a special case of clustering. We compared its clustering performance to that of three algorithms with interesting properties. First, the k-means is one of the most widely used clustering algorithms and its performance will serve as a benchmark. Second is the algorithm proposed by Einhäuser et al. (Einhäuser et al., 2002) for learning features from natural videos. It has two distinct properties: division by past trace for achieving translation or viewpoint invariance, proposed by Földiák (Földiák, 1990), and lateral inhibition for determining the winner. Third is the topology adaptive self-organizing neural network or TASONN (Datta et al., 2001) for skeletonization of data sets. It belongs to the class of algorithms known as *growing neural gas* (Fritzke, 1995) which start with a very few neurons and strategically

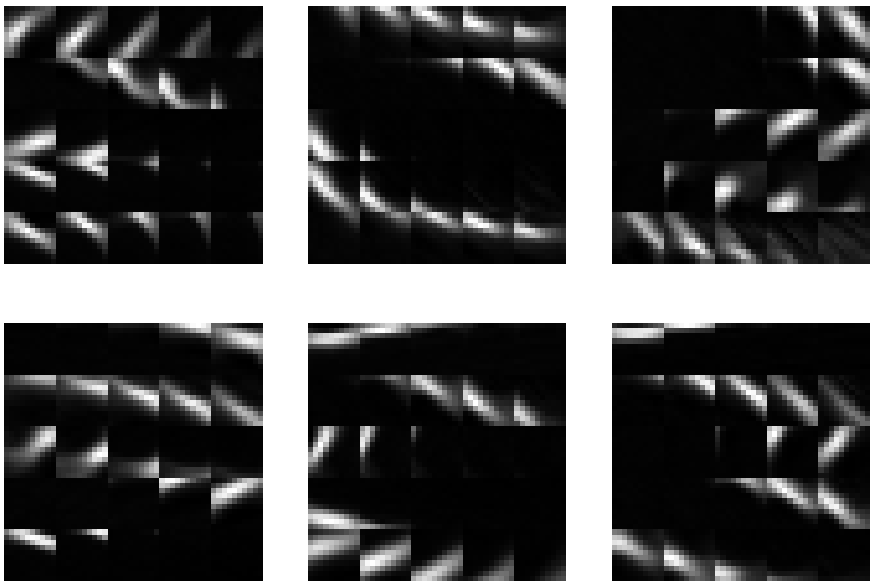


Fig. 3.3: 30 out of 100 features learned in first layer from action videos (e.g., walking, waving) are shown. Each row is a spatiotemporal feature with spatial RF size 10×10, temporal RF size 5, and direction from left to right.

add neurons and connections with learning until a stopping criterion is met. Hence, the final result is immune to bad initializations.

Five datasets from the UCI machine learning repository (Blake and Merz, 1998) were used in our experiments (see Table 3.1). Table 3.2 shows the performance (mean $\mu \pm$ std. dev. σ) of four unsupervised and two supervised algorithms over 1000 trials on each of the datasets. The advantage of TASONN and our model over k-means for initialization is revealed by the σ. On average over all datasets, the classification accuracies of Einhäuser et al.'s model and TASONN were 45%, k-means and our model were 64%, and the two supervised algorithms were 74%. For measuring similarity, k-means and TASONN use Euclidean distance while Einhäuser et al.'s and our models use dot product. Among the four unsupervised algorithms, our model performed with highest accuracy and lowest σ. Fig. 3.4 shows the variation in performance of our model for different values of η for each dataset. The best performance is achieved at $\eta = 10^{-2}$; however, for natural data with many more outliers, $\eta = 10^{-4}$ performs better.

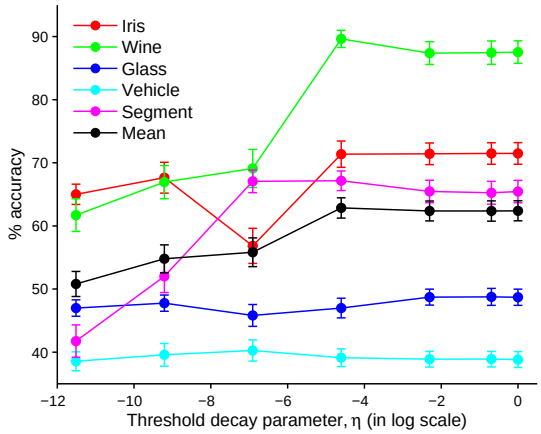


Fig. 3.4: The influence of η on the performance of our model on five UCI datasets is shown. The errorbars indicate standard deviations.

3.2 Feature Learning using Sparse Coding

3.2.1 Objective Function

Unlike spherical clustering, sparse coding uses more than one neuron to represent the input. If $\mathcal{X} = \{x_1, x_2, ...x_N\}$ and $\mathcal{W} = \{w_1, w_2, ...w_n\}$ are the set of d-dimensional data points and features respectively, $\mathcal{A} = \{a_1, a_2, ...a_N\}$ are the set of n-dimensional coefficient vectors, then a data point x can be represented as a linear combination of the features, satisfying $\|x - Wa\|_p \leq \epsilon$. For $p \geq 1$, we can define the l_p norm of a k dimensional vector y as $\|y\|_p = (\sum_{i=1}^{k} |y[i]|^p)^{\frac{1}{p}}$, where $y[i]$ denotes the i-th coordinate of y. In this work we will use $p = 2$ for the representation error.

But if $d < n$ and $\mathcal{W}$ is a full-rank martix, there are infinite number of solutions possible for this representation problem. In that case a sparsity function can be used as a

Table 3.1: Benchmark UCI datasets

Name of dataset	No. of points	No. of dimensions	No. of classes
Iris	150	4	3
Wine	178	13	3
Glass	214	9	6
Vehicle	846	18	4
Segment	2310	19	7

Table 3.2: Performance of algorithms on the UCI datasets

Name of dataset	Unsupervised ($\mu \pm \sigma$)				Supervised (μ)	
	k-means (Matlab)	Einhäuser et al.'s model	**Our model** ($\eta = 0.01$)	TASONN model	SVM (Matlab)	Mean Classifier
Iris	82.7 ± 12.4	47.1 ± 4.3	71.4 ± 2.9	90.3 ± 1.3	76.7	93.3
Wine	95.0 ± 4.0	59.4 ± 1.7	89.6 ± 1.9	42.8 ± 2.6	99.4	97.2
Glass	43.2 ± 2.8	46.6 ± 2.3	48.7 ± 1.6	45.7 ± 4.0	59.8	51.4
Vehicle	37.0 ± 0.7	30.1 ± 0.6	39.1 ± 2.0	27.7 ± 1.3	73.9	45.3
Segment	60.2 ± 6.8	37.0 ± 1.8	67.2 ± 2.4	18.7 ± 0.7	60.3	84.2

regularization term which can be viewed as a selection of relevant or important features. The overall goal is to minimize the following objective function on convergence:

$$\ell(\mathcal{X}, \mathcal{W}) = \frac{1}{2N} \sum_{i=1}^{N} \|x_i - W a_i\|_2^2 + \phi(a_i) \tag{3.6}$$

where $\phi(a_i)$ is the sparsity function. Generally we define the $\|l\|_0$ norm as the sparsity measure which counts the number of nonzero elements in a vector:
$\|y\|_0 \equiv \#\{i : y[i] \neq 0\}$, where n is a positive integer. Sometimes $\|l\|_1$ norm is also used to make the optimization problem convex.

3.2.2 Neuron

The task of the neruons in each layer is to explain the input where explanation is construed as reconstruction of the input $A^{(l-1)}$ using the learned features and their activations $A^{(l)}$, by minimizing the following loss function:

$$\mathcal{E}_{recon}(A^{(l-1)}|W^{(l-1,l)}) \equiv \frac{1}{2}\|A^{(l-1)} - W^{(l-1,l)} \times A^{(l)}\|_2^2 \tag{3.7}$$

$$\text{subject to } \|A^{(l)}\|_0 \leq n$$

where $\|A^{(l)}\|_0 \equiv \#\{i : A_i^{(l)} \neq 0\}$, n is a positive integer, and each column of $W^{(l-1,l)}$ is a feature that has been normalized to have unit norm. The condition on $A^{(l)}$ constrains the maximum number of features used in the reconstruction, thereby inducing sparsity.

If n is greater than or equal to the number of available features (i.e., number of columns in $W^{(l-1,l))}$, the simple neurons' activations may be computed using ordinary least squares in closed form as: $A^{(l)} = (W^{(l-1,l)\intercal} \times W^{(l-1,l)})^{-1} \times W^{(l-1,l)\intercal} \times A^{(l-1)}$. If n is less than the number of features, reconstruction in the model is achieved by an iterative process. This process may be implemented as orthogonal matching pursuit (OMP) (Pati et al., 1993) which is a greedy forward selection algorithm that starts with an empty list and includes at each iteration the feature most correlated with the current residual. Initially, the input (i.e., prediction error) is the residual. At each step, the feature for the maximally activated (absolute values of activations are considered) or winner simple neuron is included in the list, and all the activations are updated by computing the orthogonal projection of the input onto the linear subspace spanned by the features selected so far. The residual is updated as the difference between the input and the sum of selected features times their activations. This procedure continues until n features have been used. If $n = 1$, OMP reduces to computation of activation for spherical clustering (Dhillon and Modha, 2001) which is computationally more efficient than iterative algorithms, such as OMP with $n > 1$, and can be used to learn hierarchy of features from spatiotemporal data (Banerjee and Dutta, 2013a; Banerjee and Dutta, 2013b; Dutta and Banerjee, 2013) but has limited explanatory power.

3.2.3 Learning

Learning Lateral Connections: Consider an interconnected set of neurons, need not be fully connected but in the same layer. If the connection weights encode the correlations of neuronal activations, i.e.

$$W_{ij}(t) = A_i(t) \times A_j(t), \tag{3.8}$$

the activation of the i^{th} neuron may be spatially predicted from the activations of its neighboring neurons as:

$$\hat{A}_i(t) = \frac{1}{\kappa} \sum_{j \in \mathcal{N}(i), j \neq i} W_{ji}(t) \times A_j(t) \tag{3.9}$$

where $\kappa = \sum_{j \in \mathcal{N}(i), j \neq i} A_j^2$ is the normalization factor. This is a form of divisive normalization widely used for various purposes, such as enhancing sensitivity and discrimination, eliminating nonlinear statistical dependencies, etc. (Carandini and Heeger, 2012) in different modalities (Heeger, 1992; Olsen et al., 2010; Simoncelli and Schwartz, 2000; Wainwright et al., 2002). If the activations A_j have zero mean and unit variance, $\kappa = 1$. Otherwise, if κ is dropped, the structure in the input is still retained but with a different amplitude.

When a few neurons are activated by the partial presence of a learned input, the above model can predict remaining part of the input, thereby functioning as an associative memory. Beyond some minimum neighborhood size, smaller the neighborhood, stronger the memory i.e., more patterns can be stored accurately. If each neuron in layer ℓ encodes a feature in $W^{(\ell-1,\ell)}$, estimating the activations of these neurons from the neighboring neurons in its own layer provides a means for estimating the activations of neurons in the lower layers. To keep notations simple, this may be expressed in matrix form as:

$$\hat{A}^{(\ell-1)} = W^{(\ell-1,\ell)} \times \hat{A}^{(\ell)} = W^{(\ell-1,\ell)} \times W^{(\ell,\ell)} \times A^{(\ell)} \tag{3.10}$$

This expression does not make explicit the neighborhood information which plays a crucial role. A model that includes all neurons in a layer in the neighborhood will form a particularly inefficient storage architecture. We will continue with the matrix notation to keep expressions simple assuming the neighborhood to be implicit.

It is beneficial to keep a memory trace of the past as opposed to altering the connection weights abruptly with each input. This may be achieved by using a forgetting term β as:

$$W_{ij}(t+1) = \beta \times W_{ij}(t) + A_i(t) \times A_j(t) \tag{3.11}$$

where $0 < \beta < 1$; β may be initialized close to zero and increased exponentially with the number of observations.

Learning Feedback Connections: In our model, the top-down or feedback weights $W^{(\ell,\ell-1)}$ encode the correlation of neuronal activations between neurons in layers ℓ and $\ell - 1$. Thus, following equation 3.11,

$$W_{ij}^{(\ell,\ell-1)}(t+1) = \beta \times W_{ij}^{(\ell,\ell-1)}(t) + A_i^{(\ell)}(t) \times A_j^{(\ell-1)}(t) \tag{3.12}$$

Learning Feedforward Connections: The feedforward connections encode features in our neural architecture. We use block coordinate descent (BCD) with warm restart to learn these connections. Computational benefits of BCD, such as local computation, parameter free learning and faster convergence, over other gradient descent-like algorithms are well-known (Mairal et al., 2010). For the derivation of parameter update equations, refer to appendix in (Kong and Wang, 2012). Here we show how these equations may be implemented by exploiting the lateral and feedback connections in our model to learn the feedforward connections.

Using BCD, the i^{th} feature is updated as:

$$\triangle W_i^{(\ell-1,\ell)} = \gamma \times ((W^{(\ell,\ell-1)^{\mathrm{T}}})_i - W^{(\ell-1,\ell)} \times W_i^{(\ell,\ell)}) \tag{3.13}$$

where γ is a normalization factor, the subscript i refers to the i^{th} column of the matrix. Each column (i.e. feature) of $W^{(\ell-1,\ell)}$ is normalized to have unit norm after each update. As shown in (Mairal et al., 2010; Kong and Wang, 2012), this learning rule minimizes the well-known loss function in equation 1.1 for an optimal dictionary of features keeping the activations fixed. Convergence properties of this learning rule are explicated in (Bertsekas, 1999; Bottou and Bousquet, 2008; Mairal et al., 2010).

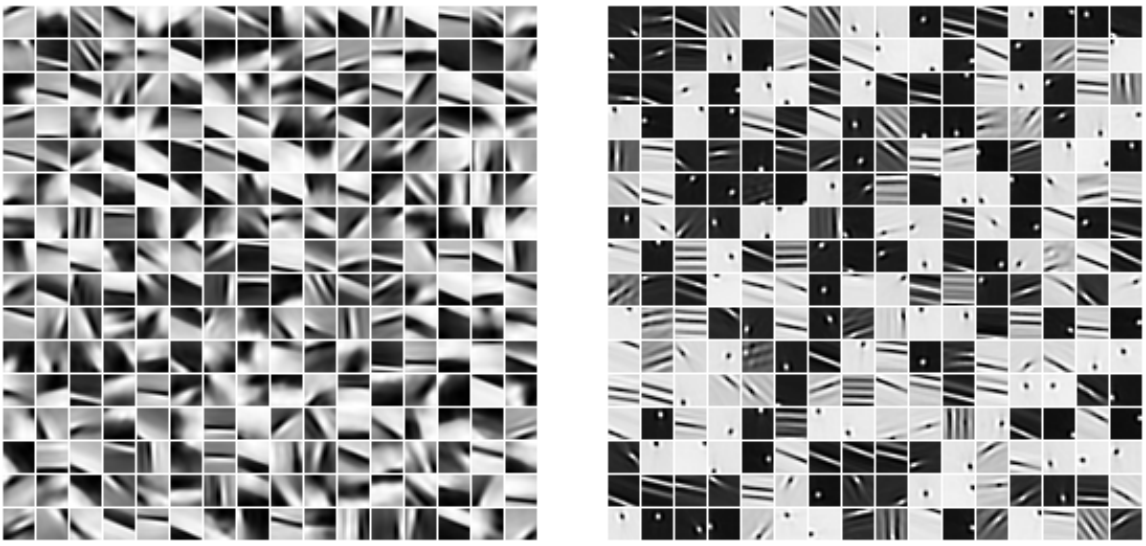
Given the activation $A^{(\ell)}$, $W^{(\ell-1,\ell)} \times W^{(\ell,\ell)} \times A^{(\ell)}$ estimates the activations $\hat{A}^{(\ell-1)}$ (ref. equation 3.10). Since $W^{(\ell,\ell-1)}$ encodes correlations between activations of layers ℓ and $\ell - 1$ (ref. equation 3.12), $(W^{(\ell,\ell-1)})^{\mathrm{T}} \times A^{(\ell)}$ also estimates $\hat{A}^{(\ell-1)}$. Thus, features encoded in the feedforward weights are learned one by one to account for the difference of estimations from two sources. By keeping the weight update rule independent of activations, BCD explains interlayer correlations by the features and lateral correlations thereby learning from relations in the input (Banerjee, 2013).

3.2.4 Experimental Results

The proposed model in this section was also deployed for learning feature hierarchies from images in an unsupervised and online manner. The layers were learned one by one. Overlap between spatial patches for adjacent nodes was 50% in the first layer. The top layer had only one node. The number of nodes in each layer is a function of the $\%$ overlaps and the RF sizes of neurons in the different layers. The second layer features are reconstructed as follows. For a neuron in the second layer, a sparse set of neurons in each first layer node that strongly (excitatory or inhibitory) connects to it is chosen. The features represented by these neurons are weighed by the connection strengths and spatially organized taking into consideration the $\%$ overlaps.

Our model learned two layers of features from thousands of natural images (downloaded from Google images). The images were converted to grayscale, and convolved with a Laplacian of Gaussian filter to crudely detect edges (performed by center-surround ganglion cells before the signal reaches V1 (Hartline, 1940; Barlow, 1953; Kuffler, 1953)). The features learned in the first layer were edges/bars in different orientations and phases (see Fig. 3.5b), similar to RFs of simple cells in V1 (Hubel, 1995). The features learned from the same data without any preprocessing, as shown in Fig. 3.5, are more useful for image reconstruction.

The second layer was learned using the same RF size as the first layer, which used the output of the first layer as input. After learning, the second layer weights became



(a) (b)

Fig. 3.5: Features of size 15×15 learned from natural images in first layer without any preprocessing (a) and after applying a Laplacian of Gaussian filter (b). A total of 256 features were learned in each case.

sparse. Each second layer unit was strongly connected to a small subset of first layer units with positive or negative values and the other values were quite small. The second layer bases are shown in Fig. 3.6. This result is similar to the model in (Lee et al., 2008) where the second layer bases encoded co-linear first layer features as well as complex features, such as intersections and angles. Several studies (Hegde and Van Essen, 2000; Ito and Komatsu, 2004) have shown that cells in V2 respond to such complex features.

Our model also learned two layers of features from 60,000 images of ten handwritten numerals $\{0, 1, ...9\}$ from the MNIST dataset (LeCun et al., 1998). As shown in Fig. 3.7, parts of numerals were learned in first layer while the neurons in second layer learned to respond to at least one instance of all the ten numerals. In the features, black color denotes inhibitory connection, white excitatory and grey neutral (close to zero). During learning from data with no temporal continuity, we set the temporal RF size (τ) of neurons to unity. These features can now be used for classification. Using our model, we also reconstructed very noisy images. As shown in Fig. 3.8, the reconstructions from

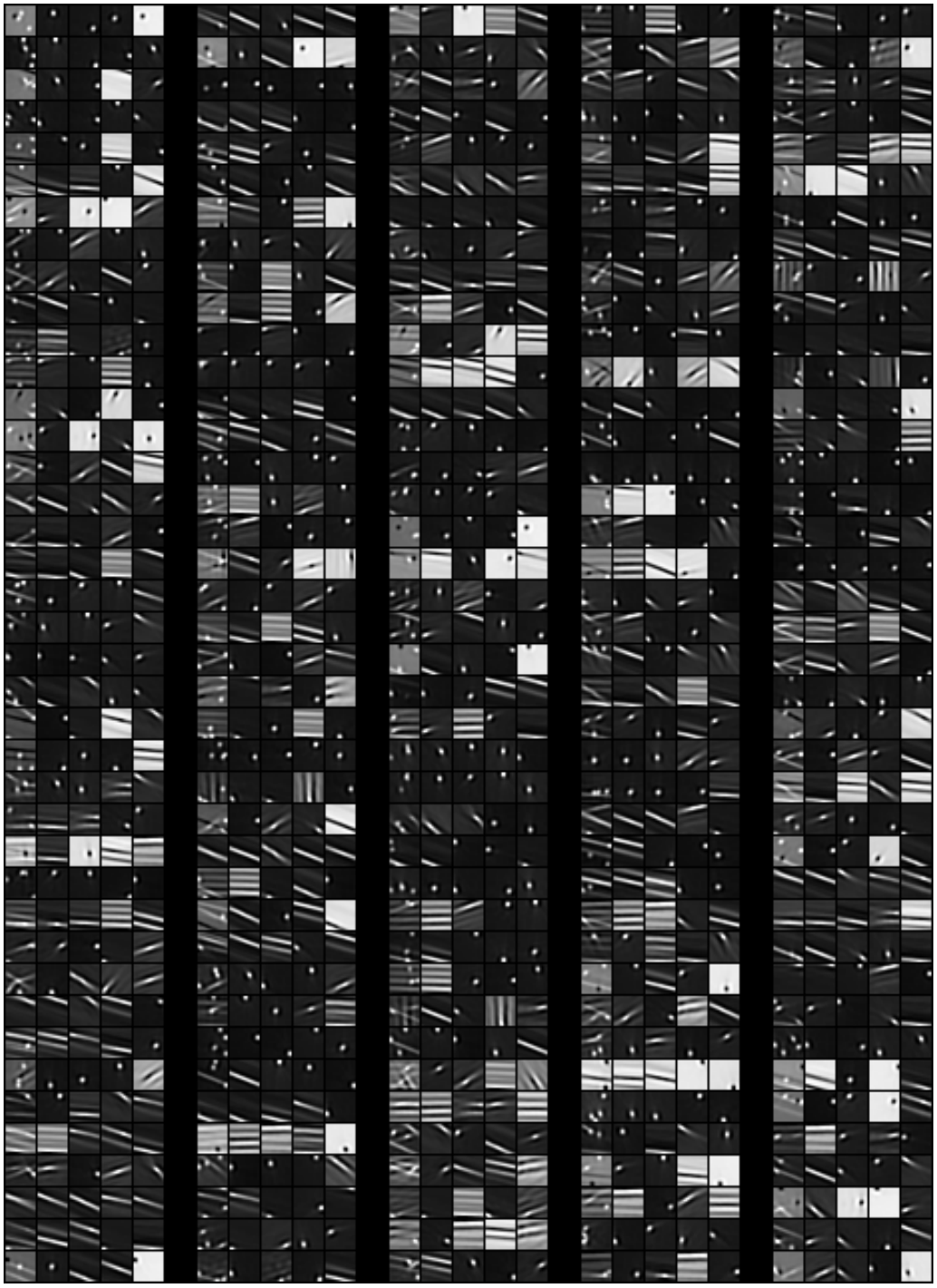
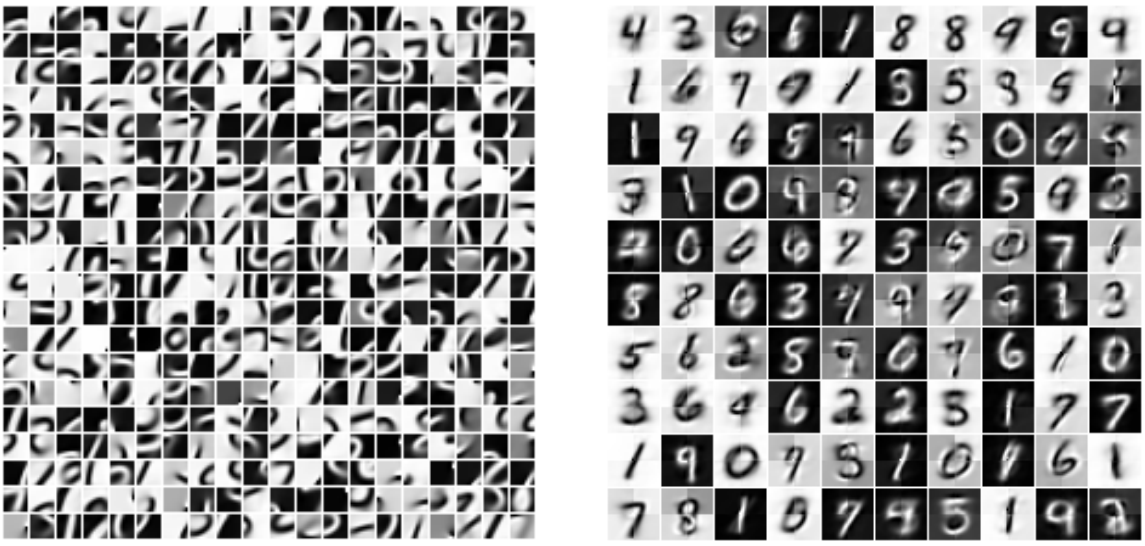


Fig. 3.6: 200 features learned by the simple sublayer of L_2 from the L_1 simple features shown in Fig. 3.5b. In each group of five patches arranged in a row, the leftmost patch represents the L_2 feature while the following four patches are the four features of L_1 simple neurons arranged in descending order of their connection strength to the L_2 neuron. Each L_2 feature is the weighted sum of the L_1 simple features. Cells in V2 are known to respond to such complex features.

second layer are much better than those from the first layer. This is expected as the higher layer has a more global view of the input and can restrict certain lower layer neurons from participating in reconstruction which would have otherwise added noise. The learning mechanism used in our model infuses high storage capacity. For example, it can learn features from 7×10^6 image patches, 12×12 pixels each, by only 256 simple neurons in one layer; these features can be used for state-of-the-art denoising (Mairal et al., 2010). This capacity grows manifold when stacked in multiple layers as done in our model. Further, the intermediate complex layers in our model induce transformation invariance which make the memory even more efficient and allows abstraction of higher-level features.



(a) (b)

Fig. 3.7: Features learned from 60,000 handwritten numerals in MNIST dataset in first (a) and second layers (b) with RF size 10×10 and 28×28 respectively. A total of 400 and 100 features were learned in the first and second layers respectively.

3.3 Comparison between Spherical Clustering and Sparse Coding

Clustering tries to learn centroids so that one input can be represented by only one cluster center. On the other way, sparse coding learns distributed representation as it can represent an input using multiple features. Empirical studies showed that spherical

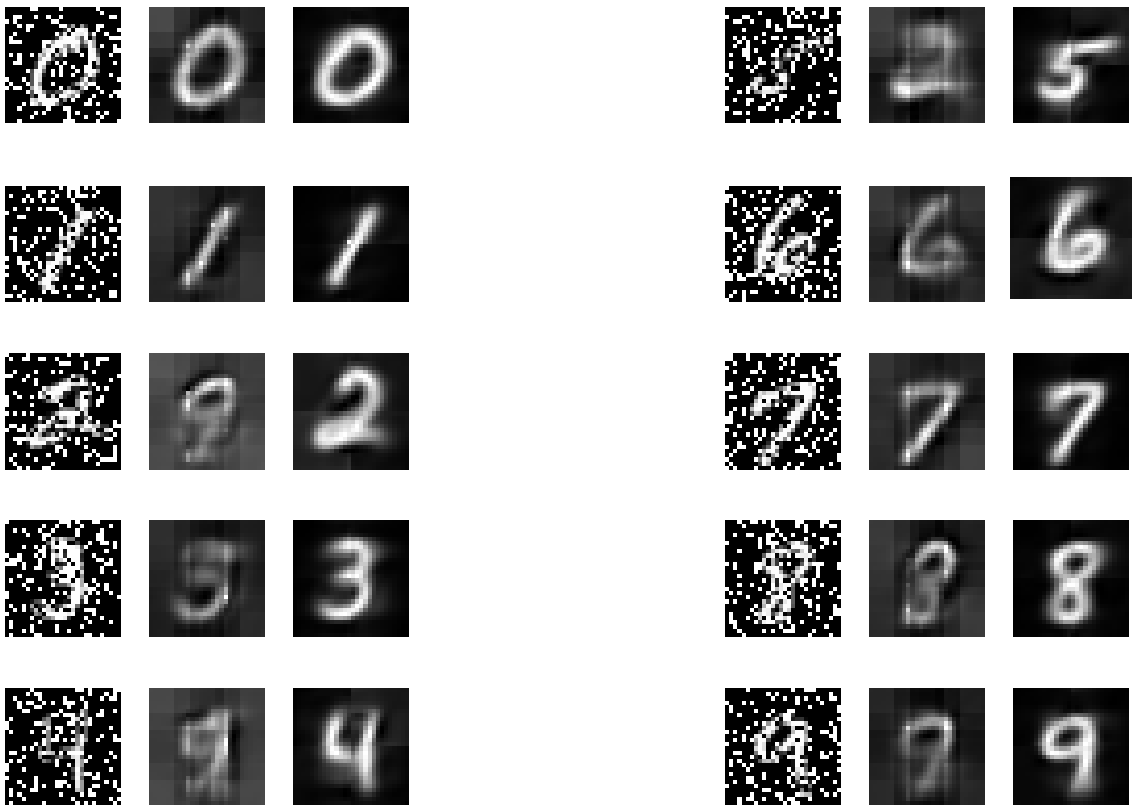


Fig. 3.8: In the left columns are shown noisy images generated by randomly inverting the intensities of at least 25% pixels. The reconstruction of these images from the first and second layers of our model are shown in the middle and right columns respectively.

Table 3.3: Classfication accuracy on MNIST using spherical clustering

Classification layer	Accuracy(%)
Layer-1 (RF = 10x10)	96.64
Layer-2 (RF = 16x16)	91.94
Layer-3 (RF = 28x28)	76.72
All layer together	97.1

Table 3.4: Classfication accuracy on MNIST using sparse coding

Classification layer	Accuracy(%)
Layer-1 (RF = 10x10)	87.47
Layer-1 (RF = 28x28)	96.19
Layer-2 (RF = 28x28)	85.45

clustering also learns sparse projections of the data under right conditions but fails as the data dimensionality increases. For details see (Coates and Ng, 2012). Sparse coding allows the features to develop both on-center and off- center receptive fields as it can use both positive and negative coefficients to explain the input. From Table 3.3 and 3.4, we can see that the classification accuracy for clustering and sparse coding is comparable which is consistent with other studies (Coates et al., 2011). In case of reconstruction sparse coding does better as it is allowed to use multiple features. And also the reconstruction from higher layer is better than lower layer. But a surprising result is that, the classification accuracy using the lower layer features are better than using higher layer features. One reason behind this is, applying a sparse learning algorithm to features that are already sparse cannot help much because sparsity can not be increased in this way. Also, when recursive layer by layer algorithm is used, higher layer learns more general structures and discards discriminative information.

One way to solve this problem is to add an invariant representation layer so that it can discard extraneous detail (like small translations of edges) in order to simplify the data. Once the data has been "smoothed" in this way, higher layers of sparse learning might actually discover structure that was not apparent before. In the next chapter, we will discuss about how to learn invariant representation from time varying data.

Chapter 4

Learning Invariant Representation

In this chapter, a two-layer network architecture is presented where the first and second layers are called simple and complex layers respectively. Each layer in our heirarchical architecture is composed of these simple and complex sublayers (see Fig. 2.3).

4.1 Invariant Representation Learning using Temporal Spherical Clustering

The model described in this section uses both topographic and temporal coherence approaches (see section 1.2 for details) and combines them in a simpler way. The model learns topographic organization of the simple cells from time varying data, assuming that the visual field changes slowly over time and also uses the same higher level representation for several consecutive time frames. It can learn variable length transformations by using an unique adaptive threshold for each complex neuron.

4.1.1 Objective Function

Formally, we define a set $\mathcal{X}$ as a finite collection of distinct alphabets, written as $\mathcal{X} = \{x_1, x_2, ...x_N\}$ where x_i is a d-dimensional alphabet or feature or event, $i \neq j$ implies $x_i \neq x_j$, and N is the cardinality of $\mathcal{X}$, i.e. $|\mathcal{X}| = N$. We define a sequence ζ over the set $\mathcal{X}$ as a finite ordered list of alphabets from $\mathcal{X}$, written as $\zeta = \langle x_1, x_2, ...x_n \rangle$ where $x_i \in \mathcal{X}$, $i < j$ implies x_i occurs before x_j, $i \neq j$ does not imply $x_i \neq x_j$, and n is the length of ζ. Therefore, learning a subset of features from recurring coincidences in the data requires clustering $\mathcal{X}$ into a set of k clusters $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, ...\mathcal{C}_k\}$. Soft-clustering is a better option for natural data.

Formation of a cluster may be viewed as a pseudo-event that occurs *where* (in case of spatial clustering) or *when* (in case of temporal clustering) all or most of the events in the cluster occur. Let,

$$S_i^{(l-1)}(p) = \begin{cases} 1, & \text{if } x_i \text{ occurs at } p \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

$$S_j^{(l)}(p) = \begin{cases} 1, & \text{if } \mathcal{C}_j \text{ occurs at } p \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

where p denotes location in case of spatial clustering and time in case of temporal. Also,

$$W_{ij}^{(l-1,l)} = Pr(S_j^{(l)} = 1 \mid S_i^{(l-1)} = 1) \tag{4.3}$$

Clustering may then be defined as an optimization problem that minimizes the following objective function:

$$\ell(\mathcal{S}^{(l-1)}, \mathcal{S}^{(l)}; W^{(l-1,l)}) = \frac{1}{2} \sum_{j=1}^{k} \sum_{i=1}^{N} \|W_{ij}^{(l-1,l)} - \frac{1}{|P_j^{(l)}|} \sum_{p \in P_j^{(l)}} S_i^{(l-1)}(p)\|^2 \tag{4.4}$$

where $W^{(l-1,l)} = [W_{ij}^{(l-1,l)}]_{N \times k}$ are the parameters of the model, $\mathcal{S}^{(l-1)} = [S_i^{(l-1)}(p)]_{N \times \mathcal{P}}$ and $\mathcal{S}^{(l)} = [S_j^{(l)}(p)]_{k \times \mathcal{P}}$ ($p = 1, ...\mathcal{P}$) are the observations. $P_j^{(l)} = \{p : S_j^{(l)}(p) = 1\}$. The $W^{(l-1,l)}$ that minimizes ℓ is a maximum a posteriori probability (MAP) estimate assuming uniform prior. This formulation is similar to *correlation clustering* (Bagon and Galun, 2011); it automatically recovers the underlying number of clusters k.

4.1.2 Neuron

A complex neuron in L_2 integrates activations from presynaptic neurons in L_1 (see Section 3.1 for learning in L_1) over its temporal RF and fires if the integrated input crosses its threshold. The activations of complex neurons in L_2 at time t are:

$$A^{(2)}(t) = \sum_{h=t_0}^{t} S^{(1)}(h) \times W^{(1,2)}(h) \tag{4.5}$$

where t_0 is a time instant from when the neurons start integrating, $t - t_0 \leq \tau^{(2)}$. Each feature in $W^{(1,2)}$ is normalized to have unit norm. A complex neuron acts as a temporal coincidence detector.

The state of the i^{th} neuron in any layer l is binary, given by

$$S_i^{(l)}(t) = \begin{cases} 1, & \text{if } A_i^{(l)}(t) > A_j^{(l)}(t), \forall j \neq i, \text{ and } A_i^{(l)}(t) > \theta_i^{(l)}(t) \\ 0, & \text{otherwise} \end{cases} \tag{4.6}$$

where $\theta_i^{(l)}(t)$ is the threshold of the i^{th} neuron in layer l at time t. This threshold is adaptive and unique for each neuron. Only the maximally activated neuron (or *winner*) in a layer is assigned the state 1 if its threshold is exceeded. Our model implements the winner-take-all mechanism which allows only the neuron of highest activity to learn. We say a neuron has *fired* if its state reaches 1.

4.1.3 Learning

Feedforward weights to neuron j in layer l with $S_j^{(l)}(t) = 1$ are updated following Hebbian rule.

$$W_{ij}^{(l-1,l)}(t+1) = (1-\alpha) \times W_{ij}^{(l-1,l)}(t) + \alpha \times S_i^{(l-1)}(h) \tag{4.7}$$

where $t_0 \leq h \leq t$, α is the learning rate that decreases with time for finer convergence, $0 < \alpha < 1$, $S^{(0)} = A^{(0)}$. This weight update rule is obtained by applying gradient descent on the objective function in eq. 4.4 in an online setting. Feedforward weights leading to each neuron are initialized to ones and normalized to have unit norm, which allows all neurons in a layer to compete on an equal footing. A new neuron is not recruited unless the incoming pattern is more similar to the initialized feature than to any of the learned features. After each update, weights to each neuron are normalized to have unit norm. Thus, feedforward connection from a presynaptic neuron (i) to a postsynaptic one (j) that

fire together are strengthened while the rest (to j) are weakened. The weakening of connections is crucial for robustness as it helps remove infrequent coincident patterns from memory which are probably noise.

In L_1, the lateral weight from neuron i to j is also updated following Hebbian rule as:

$$W_{ij}^{(1,1)}(t+1) = (1-\alpha) \times W_{ij}^{(1,1)}(t) + \alpha \times S_i^{(1)}(t-1) \times S_j^{(1)}(t) \tag{4.8}$$

Thus, connection from a presynaptic neuron (i) to a postsynaptic one (j) that fire at consecutive time instants are strengthened while the rest (from i) are weakened. The weights are randomly initialized in $(0.5-\delta, 0.5+\delta)$, $\delta \to 0$, such that $\sum_j W_{ij}^{(1,1)} = 1$, and the above learning rule ensures that constraint continues to be satisfied. Since $S^{(1)}$ is extremely sparse, $W^{(1,1)}$ can store a number of patterns from their correlations at consecutive time instants.

The threshold is updated as follows:

$$\theta_i^{(l)}(t+1) = \begin{cases} A_i^{(l)}(t), & \text{if } S_i^{(l)}(t) = 1 \\ (1-\eta) \times \theta_i^{(l)}(t), & \text{if } S_i^{(l)}(t) = 0 \text{ and } t - t_0 = \tau^{(l)} \end{cases} \tag{4.9}$$

where η is the threshold decay parameter, a constant, $0 < \eta < 1$. Due to the threshold, only a small subset of stimuli can trigger learning. The threshold decay ensures that the size of this subset remains fixed throughout the learning process, thereby maintaining the plasticity of the network. The winner-take-all mechanism along with threshold favor neurons with sparsely distributed activity.

4.1.4 Experimental Results

The proposed model was deployed for learning visual features in a node from spatiotemporal data in an unsupervised and online manner. The feedforward weights were learned layer by layer with $\alpha(t) = \alpha(t-1)/(1+t/10^6)$, $\alpha(0) = 0.1$. $\theta^{(l)}$ were initialized to a value slightly greater than $\tau^{(l)}$ such that the longest sequences may be captured. $\eta = 10^{-6}$. As stimuli we used 17 videos recorded at different natural locations with a CCD camera mounted on a cat's head exploring its environment (Betsch et al., 2004). These videos provided a continuous stream of stimuli similar to what the cat's visual system is naturally exposed to, preserving its temporal structure. The same catcam videos were used in (Einhäuser et al., 2002; Masquelier et al., 2007) for evaluating models on learning complex cell RF properties. As preprocessing, each frame (320×240 pixels) was converted to grayscale and convolved with a 3×3 Laplacian of Gaussian kernel followed by rectification to crudely highlight edges, believed to be performed by center-surround cells before the signal reaches V1. Spatiotemporal voxels of size 10×10 pixels spanning over the entire duration of a video were extracted at fixed points from a 9×11 grid, sampled every 25 pixels. These 99 voxels from each video formed our stimuli, leading to a total of about 5.3 million patches from the 17 videos.

4.1.5 Simple Layer

Our model was simulated with 625 simple neurons in L_1 with spatial RF size 10×10 pixels. Each simple neuron learned a unique visual feature from the stimuli. Qualitatively, the features belonged to three distinct classes of RFs – small unoriented features, localized and oriented Gabor-like filters, and elongated edge-detectors (see Fig. 4.1). Such features have been observed in macaque V1, and have been reported to be learned by computational models such as SAILnet (Zylberberg et al., 2011) and SSC (Rehn and Sommer, 2007).

If lateral connections encode transition probabilities and minimization of wiring length is an objective, neurons that fire in close temporal proximity will end up being

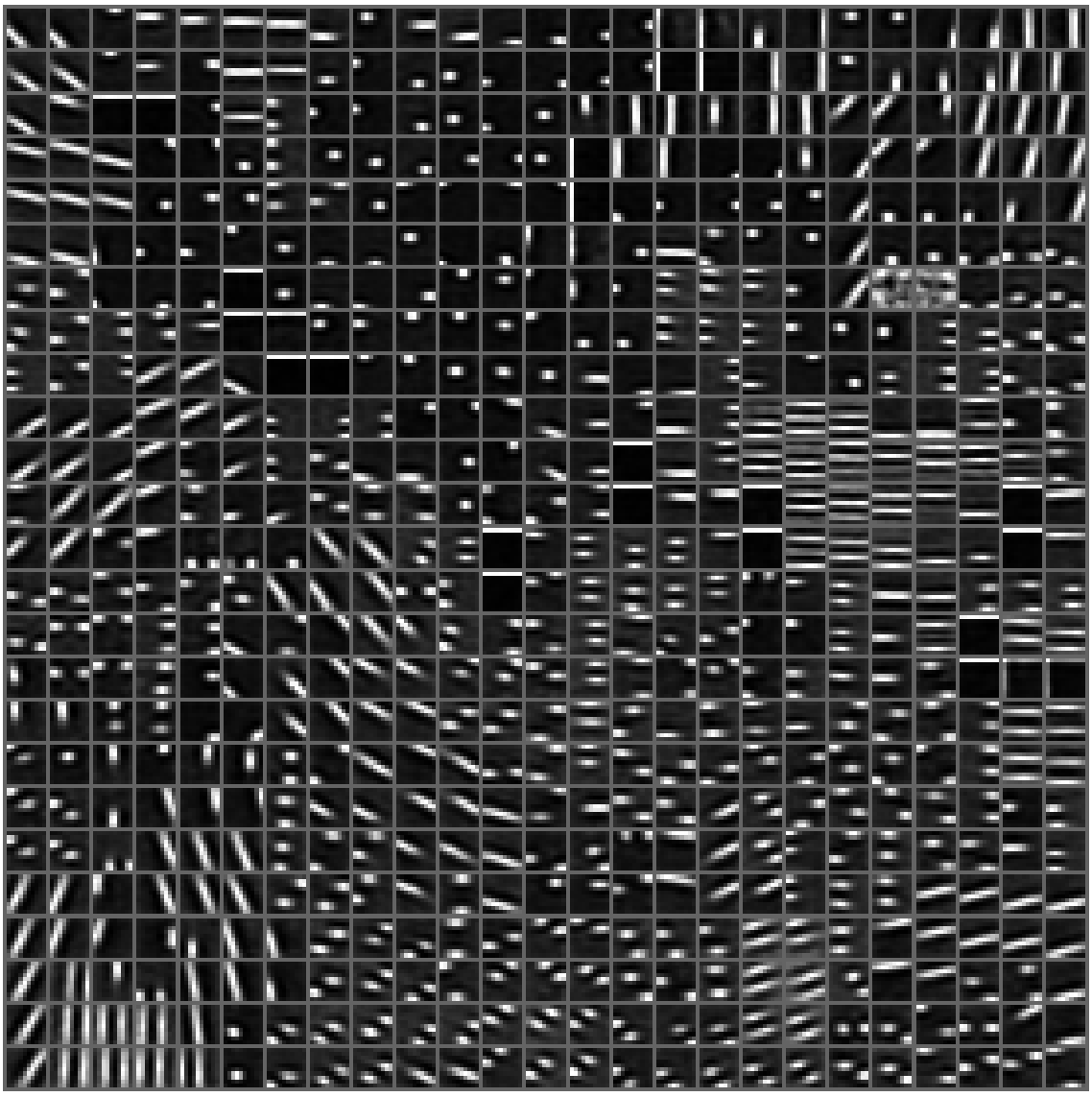


Fig. 4.1: Features learned by 625 neurons in L_1 from the catcam video.

spatial neighbors. Furthermore, if the stimulus changes gradually, neighboring neurons will develop similar feature preferences. In order to learn features in a topographic map, we organize the simple layer neurons on a 2D grid. At any time t, the activation A_i of the winner neuron i at time $t-1$ is propagated to its neighbor j ($j \neq i$), the effect of which exponentially decreases with square of the distance d_{ij} between i and j on the grid. For neighbor j at time t, the propagated activation is:

$$\psi_{ij}^{(1)}(t) = \begin{cases} e^{-\gamma \times d_{ij}^2}, & \text{if } i \neq j \\ 0, & \text{otherwise} \end{cases} \tag{4.10}$$

where γ is a constant, $\gamma = 2$. At any time t, in addition to feedforward activation, each simple neuron receives an activation from a neighboring winner in the same layer. The simple layer activation is:

$$A^{(1)}(t) = A^{(0)}(t) \times W^{(0,1)}(t) + S^{(1)}(t-1) \times \psi^{(1)}(t) \tag{4.11}$$

where $\psi^{(1)} = [\psi_{ij}^{(1)}]_{|L_1| \times |L_1|}$, $|L_1|$ is the number of neurons in L_1. The second term biases neighboring neurons to become the winner at the next instant. As a result, simple neurons that fire in close temporal proximity end up being spatially close in the 2D grid. Consequently, the wiring length for pooling by complex neurons is reduced, in agreement with biological evidence (Blasdel, 1992; DeAngelis et al., 1999). The topographic map is shown in Fig. 4.1. The pooling region in this topographic map as learned by each complex neuron is shown in Fig. 4.5.

4.1.6 Complex Layer

Our model was simulated with 25 complex neurons in L_2 with temporal RF size of 21 sampling instants. Being exposed to the catcam videos, each complex neuron got strongly connected to a subset of simple neurons in L_1 i.e., it learned a unique transformation to which it is now invariant. The spatial feature encoded by each simple neuron in this subset is an instance of the transformation. The activation of a complex neuron is high if the spatial stimulus matches any of these spatial features, and low otherwise. Thus, the response of complex neurons in our model is akin to that of complex cells in V1.

Due to the nature of stimulus, our model was exposed to sequences of spatial stimuli in the catcam video. Repeating sequences, if learned, would be useful for prediction. When trained with a sequence (e.g., $\langle A, B, C, D, E \rangle$), a complex neuron in our

model responds much more vigorously (as measured by its activation) to the corresponding set (e.g., $\{A, B, C, D, E\}$) than to any other (e.g., $\{I, J, K\}$), where each alphabet refers to a unique spatial feature. Further, it responds more vigorously to the training sequence than to any other (e.g., $\langle E, D, C, B, A\rangle$), thereby manifesting the complex neuron's direction selectivity. This is achieved by exploiting the set learned by the complex neuron in conjunction with the transition probabilities learned by the lateral connections in the simple layer. The difference in activations towards the training sequence and any of its other permutation depends on how often other permutations of the set are presented. If no other permutation is presented, the difference in activations is high. In V1, 10-20% cells show marked direction selectivity (Hubel, 1995).

Prediction in our model amounts to computing the probability of the i^{th} simple neuron being the winner at time $t+1$ given that the j^{th} simple neuron was the winner at time t, i.e. probability of $S_i^{(1)}(t+1) = 1$ given $S_j^{(1)}(t) = 1$, which depends on the transition probabilities as well as the sets learned by the complex neurons. At any instant, the winner complex neuron (say, k) restricts the set for the expected winner simple neuron. The highest expected one is then chosen from this set using the transition probabilities.

$$Pr(S_i^{(1)}(t+1) = 1 \mid S_j^{(1)}(t) = 1) = \kappa \times (\frac{W_{jk}^{(1,2)}}{\sum_k W_{jk}^{(1,2)}} \times \frac{W_{ik}^{(1,2)}}{\sum_i W_{ik}^{(1,2)}} + W_{ji}^{(1,1)}) \quad (4.12)$$

where κ is the uniform prior distribution. Fig. 4.2 shows the entropy of the system as it converges with learning. Fig. 4.5 shows the sets and sequences learned by eight L_2 neurons in our model. To reconstruct the sequence learned by a L_2 neuron, we select the strongest connected feature from its set; its successor is that feature from the set that has the strongest lateral connection (the algorithmic implementation of equ. 4.12), and so on until a feature is repeated, signifying the end of sequence.

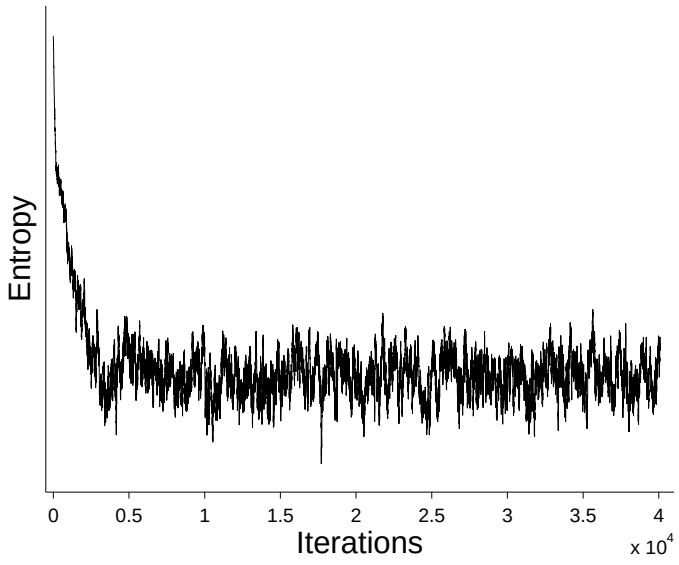


Fig. 4.2: Entropy of the system as it learns from natural stimuli.

4.2 Invariant Representation Learning using Generative Model

We can also apply the same approach like (Gregor and LeCun, 2011) but a slightly different objective funtion to learn the complex layer.

4.2.1 Objective Function

The complex neurons in L_2 can be learned by minimizing the following objective function:

$$l(W^{(1,2)}, A^{(2)}) = \frac{1}{2}\|(\sum_{t=t_0}^{\tau} |A^{(1)}(t)|) - W^{(1,2)}A^{(2)}\|_2^2 \tag{4.13}$$

$$\text{subject to } \; W^{(1,2)}, A^{(2)} \geq 0$$

where,

$$A^{(1)}(t) = \min_{A^{(1)}} \frac{1}{2}\|A^{(0)}(t) - W^{(0,1)}A^{(1)}(t)\|_2^2 \tag{4.14}$$

$$\text{subject to } \; \|A^{(1)}(t)\|_0 \leq n$$

(See Section 3.2 for learning in L_1)

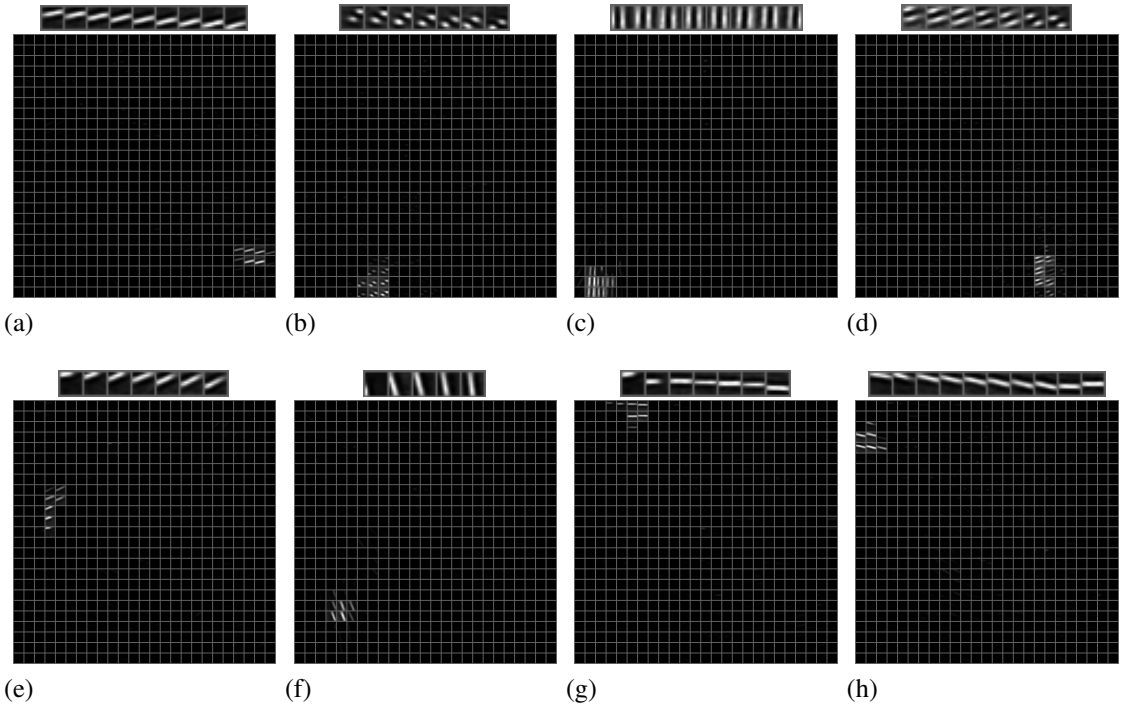


Fig. 4.3: Sequences and feedforward connection strengths learned by eight (out of 25) L_2 neurons from the catcam videos are shown in (a) through (h). In (a), the top figure shows the sequence of length 9 learned by this L_2 neuron. The bottom figure shows the connection strengths to the 625 L_1 neurons learned by this L_2 neuron. Similarly for (b) through (h). The L_2 neurons learn variable length sequences even with the same $\tau^{(2)}$ (=21).

It is an online version of Non-negative Matrix Factorization (NMF). It is well known that NMF can learn part based representation (Lee and Seung, 1999; Hoyer and Dayan, 2004). That means, if there are some simple neruons those co-occur frequently in close time interval, they will be strongly connected to one complex neuron.

4.2.2 Neuron

The task of the complex neurons is to reconstruct the accumulated simple neurons' activation using the learned weights between simple and complex layers, by minimizing the following loss function:

$$l(A^{(2)}|W^{(1,2)}) = \frac{1}{2}\|(\sum_{t=t_0}^{\tau}|A^{(1)}(t)|) - W^{(1,2)}A^{(2)}\|_2^2 \tag{4.15}$$

$$\text{subject to } A^{(2)} \geq 0$$

The solution of the objective function can be found using non-negative least square (Lawson and Hanson, 1995).

4.2.3 Learning

Learning can be done either by Stochastic Gradient Descent or the method described in section 3.2.3. After each update an additional thresholding should be done to keep the weights non-negative.

4.2.4 Experimental Results

The proposed model was deployed for learning from the same data described in Section 4.1. The data was preprocessed like in (Coates and Ng, 2012). Each image patch was first contrast normalized by subtracting the mean and divide by the standard deviation. A small value was added to the variance to avoid divide by zero problem. After contrast normalization each voxel was whitened by ZCA transform.

The network was simulated with 225 neurons in simple layer and 50 neurons in complex layer with the parameters n = 10 and τ = 10. The responses of the neurons in L_1 and L_2 are akin to that of simple and complex cells in V1.

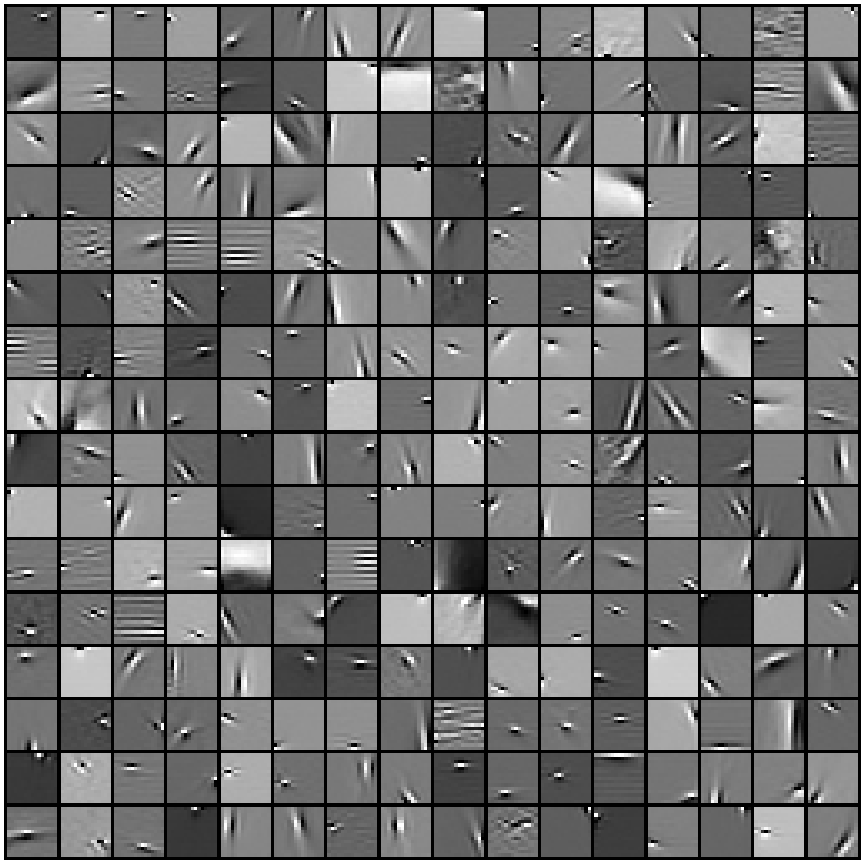


Fig. 4.4: Features learned by 225 neurons in simple sublayer of L_1 from the catcam videos.

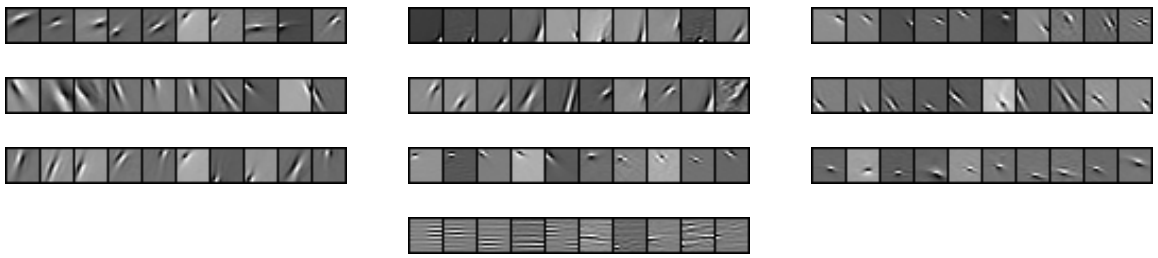


Fig. 4.5: Ten most strongly connected simple features (from Fig. 4.4) to each of 10 (out of 50) complex neurons in L_1. These connections were learned from the catcam videos. Temporal RF size was 10.

Chapter 5

Conclusion

This thesis has presented a hierarchical neural network that learns a hierarchy of overcomplete and sparse feature dictionaries in an unsupervised and online manner by capturing repeating coincident patterns from space- and time-varying data. The model learns meaningful features in each layer that correspond to objects in higher layers and object-parts in lower layers. Two algorithms are investigated: recursive layer-by-layer spherical clustering and sparse coding to learn feature hierarchies. For the spherical clustering, we have presented a fully-learnable model, with only two manually tunable parameters. We have used the McCulloch-Pitts neuron model with a variable threshold that is unique for each neuron and adaptive to the data. A constant parameter was used to decay this threshold such that the influence of outliers on learning may be controlled. This is crucial for using the same model for learning from data with different proportion of outliers, such as, natural images with a large number of outliers and clean handwritten numerals, as in MNIST dataset, with very few outliers. Learning was facilitated by the Hebbian rule and winner-take-all mechanism. For the case of sparse coding, higher layer neurons, when exposed to noisy data, could denoise the data better than their lower layer counterparts, thereby justifying a hierarchical organization. Classification accuracy is comparable for both algorithms and classification accuracy obtained from a lower layer is better than a higher layer. The architecture scales to realistic-sized high-dimensional data and an arbitrary number of layers.

Learning features invariant to arbitrary transformations in the data is a requirement for any recognition system, biological or artificial. In each layer of the hierarchy, there should be some procedure so that the responses of the features are invariant to small transformations and distortions for robust recognition. Biological evidence and computational models have supported the role of simple-complex layers in V1 in achieving this goal. We have presented a two-layered neural model that learns features in

simple layer and feature subset invariant to arbitrary transformations in complex layer using spatial and temporal spherical clustering respectively. When exposed to natural videos recorded with a camera mounted on a cats head, the first layer neurons learned spatial features that resemble the RFs in macaque V1 while the second layer neurons learned arbitrary transformations in the data; their activations were then invariant to these transformations akin to the response of complex cells in V1. The simple and complex RFs were learned by spherical clustering in space and time respectively where the outliers were not allowed to influence the cluster centers. The model could make higher-order predictions by simultaneously exploiting the transformations learned in the complex layer and transition probabilities learned by the lateral connections in the simple layer. We showed the convergence of this predictive model while learning from the catcam videos. Unlike other models with predefined pooling regions or presumed group sparsity for learning topographic maps from spatial data, we used temporal continuity of data and physical constraints to learn topographic feature map. We have also presented a generative model that can learn arbitrary transformations from the data using non-negative matrix factorization.

We have separately presented a hierarchical neural network to learn feature hierarchies and procedure for learning features invariant to arbitrary transformations in the data. In the future, we will include this invariance learning procedure in the hierarchical network to do robust and state-of-the art recognition performance in different real world data sets.

REFERENCES

Abbott, L. F. (1999). Lapique's introduction of the integrate-and-fire model neuron (1907). *Brain Research Bulletin*, 50(5/6):303–304.

Aharon, M., Elad, M., and Bruckstein, A. (2006). K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Trans. Signal Processing*, 54(11):4311–4322.

Bach-y-Rita, P. (2004). Tactile sensory substitution studies. *Annals New York Acad. Sci.*, 1013:83–91.

Bach-y-Rita, P. and Kercel, S. W. (2003). Sensory substitution and the human-machine interface. *Trends in Cognitive Sci.*, 7(12):541–546.

Bagon, S. and Galun, M. (2011). Large scale correlation clustering optimization. *Comput. Res. Repos.*, arXiv:1112.2903.

Banerjee, B. (2012). Learning lateral connections among neurons from correlations of their surprises. 28th Symp. Comput. Foundations of Perception and Action. Center for Visual Science, University of Rochester, NY.

Banerjee, B. (2013). How can the blind men see the elephant? In Risi, S., Lehman, J., and Clune, J., editors, *How Should Intelligence be Abstracted in AI Research*, number FSS-13-02 in AAAI Fall Symp., page [Forthcoming]. Arlington, VA.

Banerjee, B. and Dutta, J. K. (2013a). Hierarchical feature learning from sensorial data by spherical clustering. In *IEEE BigData Workshop on Scalable Machine Learning*, page [Forthcoming], Santa Clara, CA.

Banerjee, B. and Dutta, J. K. (2013b). An online clustering algorithm that ignores outliers: Application to hierarchical feature learning from sensory data. In *IEEE ICDM Workshop on Incremental Clustering, Concept Drift and Novelty Detection*, page [Forthcoming], Dallas, TX.

Banerjee, B. and Dutta, J. K. (2013c). SELP: A general-purpose framework for learning the norms from saliencies in spatiotemporal data. *Neurocomputing: Special Issue on Brain Inspired Models of Cognitive Memory*, page [Forthcoming].

Barlow, H. B. (1953). Summation and inhibition in the frog's retina. *J. Physiology*, 119(1):69–88.

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127.

Bergstra, J. and Bengio, Y. (2009). Slow, decorrelated features for pretraining complex cell-like networks. In Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C. K. I., and Culotta, A., editors, *Advances in Neural Information Processing Systems*, volume 22, pages 99–107. MIT Press.

Berkes, P. and Wiskott, L. (2005). Slow feature analysis yields a rich repertoire of complex cell properties. *J. Vision*, 5(6):579–602.

Bertsekas, D. P. (1999). *Nonlinear programming*. Athena Scientific, Belmont, MA.

Betsch, B. Y., Einhäuser, W., Körding, K. P., and König, P. (2004). The world from a cat's perspective – statistics of natural videos. *Biol. Cybernetics*, 90(1):41–50.

Blake, C. L. and Merz, C. J. (1998). UCI repository of machine learning databases. University of California Irvine, Available at www.ics.uci.edu/∼mlearn.

Blasdel, G. G. (1992). Orientation selectivity, preference, and continuity in monkey striate cortex. *J. Neurosci.*, 12(8):3139–3161.

Bottou, L. and Bousquet, O. (2008). The trade-offs of large scale learning. In *Advances in Neural Information Processing Systems*, volume 20, pages 161–168. MIT Press.

Cadieu, C., Kouh, M., Pasupathy, A., Connor, C. E., Riesenhuber, M., and Poggio, T. (2007). A model of V4 shape selectivity and invariance. *J. Neurophysiology*, 98:1733–1750.

Cadieu, C. and Olshausen, B. A. (2008). Learning transformational invariants from natural movies. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems*, volume 21, pages 209–216. MIT Press.

Carandini, M. and Heeger, D. J. (2012). Normalization as a canonical neural computation. *Nature Reviews Neurosci.*, 13(1):51–62.

Coates, A. and Ng, A. Y. (2012). Learning feature representations with k-means. In *In Neural Networks: Tricks of the Trade, Reloaded*. Springer LNCS.

Coates, A., Ng, A. Y., and Lee, H. (2011). An analysis of single-layer networks in unsupervised feature learning. *Journal of Machine Learning Research - Proceedings Track*, 15:215–223.

Constantine-Paton, M. and Law, M. I. (1978). Eye-specific termination bands in tecta of three-eyed frogs. *Science*, 202(4368):639–641.

Datta, A., Parui, S. K., and Chaudhuri, B. B. (2001). Skeletonization by a topology-adaptive self-organizing neural network. *Pattern Recognition*, 34(3):617–629.

David, S. V., Mesgarani, N., Fritz, J. B., and Shamma, S. A. (2009). Rapid synaptic depression explains nonlinear modulation of spectro-temporal tuning in primary auditory cortex by natural stimuli. *J. Neurosci.*, 29(11):3374–3386.

DeAngelis, G. C., Ghose, G. M., Ohzawa, I., and Freeman, R. D. (1999). Functional micro-organization of primary visual cortex: Receptive field analysis of nearby neurons. *J. Neurosci.*, 19(10):4046–4064.

Dhillon, I. S. and Modha, D. S. (2001). Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1-2):143–175.

Douglas, R. J. and Martin, K. A. C. (2010). *Canonical cortical circuits*, pages 15–21. Handbook of Brain Microcircuits.

Dutta, J. K. and Banerjee, B. (2013). Learning features and their transformations by spatial and temporal spherical clustering. *Comput. Res. Repos.*, arXiv:1308.2350.

Dutta, J. K., Gu, J., Kasani, R. P., and Banerjee, B. (2012). A multilayered neural network model for verifying the common cortical algorithm hypothesis. 28th Symp. Comput. Foundations of Perception and Action. Center for Visual Science, University of Rochester, NY.

Einhäuser, W., Kayser, C., König, P., and Körding, K. P. (2002). Learning the invariance properties of complex cells from their responses to natural stimuli. *European J. Neurosci.*, 15(3):475–486.

Farabet, C., LeCun, Y., Kavukcuoglu, K., Culurciello, E., Martini, B., Akselrod, P., and Talay, S. (2011). Large-scale FPGA-based convolutional networks. In Bekkerman, R., Bilenko, M., and Langford, J., editors, *Scaling up Machine Learning: Parallel and Distributed Approaches*. Cambridge University Press.

Földiák, P. (1990). Forming sparse representations by local anti-hebbian learning. *Biol. Cybernetics*, 64:165–170.

Földiák, P. (1991). Learning invariance from transformation sequences. *Neural Comput.*, 3(2):194–200.

Fritzke, B. (1995). A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems 7*, pages 625–632. MIT Press.

Fukushima, K. (1980). Neocognitron: a self organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol. Cybernetics*, 36(4):193–202.

Fukushima, K. (1988). Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural Networks*, 1(2):119–130.

Fukushima, K. (2003). Neocognitron for handwritten digit recognition. *Neurocomputing*, 51(1):161–180.

Garrigues, P. and Olshausen, B. A. (2010). Group sparse coding with a laplacian scale mixture prior. *Advances in Neural Information Processing Systems*, 23:1–9.

George, D. (2008). *How the brain might work: A hierarchical and temporal model for learning and recognition*. PhD thesis, Stanford University, CA.

George, D. and Hawkins, J. (2005). A hierarchical bayesian model of invariant pattern recognition in the visual cortex. In *Proc. IEEE Intl. Joint Conference on Neural Networks*, volume 3, pages 1812–1817.

George, D. and Hawkins, J. (2009). Towards a mathematical theory of cortical micro-circuits. *PLoS Comput. Biol.*, 5(10).

Gorelick, L., Blank, M., Shechtman, E., Irani, M., and Basri, R. (2007). Actions as space-time shapes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29(12):2247–2253.

Gregor, K. and LeCun, Y. (2010). Emergence of complex-like cells in a temporal product network with local receptive fields. *Comput. Res. Repos.*, arXiv:1006.0448.

Gregor, K. and LeCun, Y. (2011). Efficient learning of sparse invariant representations. *CoRR*, arXiv:1105.5307.

Hartline, H. K. (1940). The effects of spatial summation in the retina on the excitation of the fibers of the optic nerve. *American J. Physiology–Legacy Content*, 130(4):700–711.

Hawkins, J., Ahmad, S., and Dubinsky, D. (2011). *Hierarchical Temporal Memory including HTM Cortical Learning Algorithms*. Numenta, Inc., 0.2.1 edition.

Heeger, D. J. (1992). Normalization of cell responses in cat striate cortex. *Visual Neurosci.*, 9:181–197.

Hegde, J. and Van Essen, D. C. (2000). Selectivity for complex shapes in primate visual area V2. *J. Neurosci.*, 20(5):RC61 1–6.

Hinton, G. E. (2007). Learning multiple layers of representation. *Trends in Cognitive Sci.*, 11:428–434.

Hinton, G. E., Osindero, S., and Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Ho, Y. C. and Pepyne, D. L. (2002). Simple explanation of the no-free-lunch theorem and its implications. *J. Optimization Theory and Applications*, 115(3):549–570.

Hoyer, P. O. and Dayan, P. (2004). Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469.

Hubel, D. H. (1995). *Eye, Brain, and Vision*. W. H. Freeman, 2nd edition.

Hubel, D. H. and Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cats visual cortex. *J. Physiology*, 160:106–154.

Hubel, D. H. and Wiesel, T. N. (1965). Binocular interaction in striate cortex of kittens reared with artificial squint. *J. Neurophysiology*, 28:1041–1059.

Hubel, D. H. and Wiesel, T. N. (1968). Receptive fields and functional architecture of monkey striate cortex. *J. Physiology*, 195:215–243.

Hubel, D. H. and Wiesel, T. N. (1977). Ferrier lecture. Functional architecture of macaque monkey visual cortex. *Proc. Royal Society Lond. B Biol. Sci.*, 198:1–59.

Hyvarinen, A. and Hoyer, P. O. (2001). A two-layer sparse coding model learns simple and complex cell receptive fields and topography from natural images. *Vision Res.*, 41(18):2413–2423.

Ito, M. and Komatsu, H. (2004). Representation of angles embedded within contour stimuli in area V2 of macaque monkeys. *J. Neurosci.*, 24(13):3313–3324.

Ji, S., Xu, W., Yang, M., and Yu, K. (2010). 3D convolutional neural networks for human action recognition. In *Intl. Conf. Machine Learning*, pages 221–231.

Kavukcuoglu, K., Ranzato, M. A., Fergus, R., and LeCun, Y. (2009). Learning invariant features through topographic filter maps. In *IEEE Conf. Computer Vision Pattern Recog.*

Kong, S. and Wang, D. (2012). Online discriminative dictionary learning for image classification based on block-coordinate descent method. *CoRR*, abs/1203.0856.

Kouh, M. and Poggio, T. (2008). A canonical neural circuit for cortical nonlinear operations. *Neural Computation*, 20:1427–1451.

Kuffler, S. W. (1953). Discharge patterns and functional organization of mammalian retina. *J. Neurophysiol.*, 16(1):37–68.

Larochelle, H., Bengio, Y., Louradour, J., and Lamblin, P. (2009). Exploring strategies for training deep neural networks. *J. Machine Learning Res.*, 10:1–40.

Lawson, C. L. and Hanson, R. J. (1995). *Solving least squares problems*, volume 15 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA.

Le, Q. V., Zou, W., Yeung, S., and Ng, A. Y. (2011). Learning hierarchical spatio-temporal features for action recognition with independent subspace analysis. In *IEEE Conf. Computer Vision Pattern Recog.*, pages 3361–3368.

LeCun, Y. and Bengio, Y. (1995). Convolutional networks for images, speech and time series. In Arbib, M. A., editor, *The Handbook of Brain Theory and Neural Networks*, pages 255–258. MIT Press.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1(4):541–551.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324.

Lee, D. D. and Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791.

Lee, H., Ekanadham, C., and Ng, A. (2008). Sparse deep belief net model for visual area V2. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA.

Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Intl. Conf. Machine Learning*, pages 609–616.

Long, P. M., Servedio, R. A., and Simon, H. U. (2007). Discriminative learning can succeed where generative learning fails. *Information Processing Letters*, 103(4):131–135.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *Intl. J. Computer Vision*, 60:91–110.

Mairal, J., Bach, F., Ponce, J., and Sapiro, G. (2010). Online learning for matrix factorization and sparse coding. *J. Machine Learning Res.*, 11:19–60.

Mairal, J., Jenatton, R., Obozinski, G., and Bach, F. (2011). Convex and network flow optimization for structured sparsity. *J. Mach. Learn. Res.*, 12:2681–2720.

Martinez, L. M. and Alonso, J. M. (2003). Complex receptive fields in primary visual cortex. *The Neuroscientist*, 9(5):317–331.

Masquelier, T., Serre, T., Thorpe, S., and Poggio, T. (2007). Learning complex cell invariance from natural videos: A plausibility proof. Technical Report 60, MIT, Cambridge, MA.

Metin, C. and Frost, D. O. (1989). Visual responses of neurons in somatosensory cortex of hamsters with experimentally induced retinal projections to somatosensory thalamus. *Proc. Natl. Acad. Sci.*, 86(1):357–361.

Mountcastle, V. (1978). An organizing principle for cerebral function: The unit model and the distributed system. In *The Mindful Brain*. MIT Press.

Ng, A. Y. and Jordan, M. (2001). On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In *Advances in Neural Information Processing Systems*, volume 14.

Olsen, S. R., Bhandawat, V., and Wilson, R. I. (2010). Divisive normalization in olfactory population codes. *Neuron*, 66(2):287–299.

Olshausen, B. A. and Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609.

Pati, Y. C., Rezaiifar, R., and Krishnaprasad, P. S. (1993). Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *27th Asilomar Conf. Signals, Systems and Computers*, pages 40–44. IEEE.

Ranzato, M., Boureau, Y., Chopra, S., and LeCun, Y. (2007). A unified energy-based framework for unsupervised learning. *J. Machine Learning Res.*, 2:371–379.

Rehn, M. and Sommer, F. T. (2007). A network that uses few active neurones to code visual input predicts the diverse shapes of cortical receptive fields. *J. Comput. Neurosci.*, 22(2):135–146.

Riesenhuber, M. and Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature Neurosci.*, 2(11):1019–1025.

Roe, A. W., Pallas, S. L., Kwon, Y. H., and Sur, M. (1992). Visual projections routed to the auditory pathway in ferrets: Receptive fields of visual neurons in primary auditory cortex. *J. Neurosci.*, 12:3651–3664.

Rust, N. and DiCarlo, J. J. (2008). Increases in selectivity are offset by increases in tolerance ("invariance") to maintain sparseness across the ventral visual pathway. Society for Neurosci. Annual Meeting, Washington, DC.

Rust, N., Schwartz, O., Movshon, J. A., and Simoncelli, E. P. (2005). Spatiotemporal elements of macaque V1 receptive fields. *Neuron*, 6(6):945–956.

Schmah, T., Hinton, G., Zemel, R. S., Small, S. L., and Strother, S. C. (2009). Generative versus discriminative training of RBMs for classification of fMRI images. In *Advances in Neural Information Processing Systems*, volume 21, pages 1409–1416. MIT Press.

Serre, T. (2006). Learning a dictionary of shape-components in visual cortex: Comparison with neurons, humans and machines. Technical Report 28, MIT, CSAIL.

Serre, T., Oliva, A., and Poggio, T. (2007a). A feedforward architecture accounts for rapid categorization. *Proc. Natl. Acad. Sci.*, 104(15):6424–6429.

Serre, T., Wolf, L., Bileschi, S., Riesenhuber, M., and Poggio, T. (2007b). Robust object recognition with cortex-like mechanisms. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29:411–426.

Simoncelli, E. P. and Schwartz, O. (2000). Natural sound statistics and divisive normalization in the auditory system. In *Advances in Neural Information Processing Systems*, volume 13, pages 166–172. MIT Press.

Sirosh, J. and Miikkulainen, R. (1997). Topographic receptive fields and patterned lateral interaction in a self-organizing model of the primary visual cortex. *Neural Computation*, 9(3):577–594.

Vapnik, V. (1998). *Statistical learning theory*. John Wiley and Sons, New York, NY.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P. A. (2008). Extracting and composing robust features with denoising autoencoders. In *Intl. Conf. Machine Learning*, pages 1096–1103.

von Melchner, L., Pallas, S. L., and Sur, M. (2000). Visual behaviour mediated by retinal projections directed to the auditory pathway. *Nature*, 404(6780):871–876.

Wainwright, M. J., Schwartz, O., and Simoncelli, E. P. (2002). Natural image statistics and divisive normalization. In Rao, R. P. N., Olshausen, B. A., and Lewicki, M. S., editors, *Probabilistic Models of the Brain: Perception and Neural Function*, chapter 10, pages 203–222. MIT Press.

Wiskott, L. and Sejnowski, T. J. (2002). Slow feature analysis: Unsupervised learning of invariances. *Neural Computation*, 14(4):715–770.

Zylberberg, J., Murphy, J. T., and DeWeese, M. R. (2011). A sparse coding model with synaptically local plasticity and spiking neurons can account for the diverse shapes of V1 simple cell receptive fields. *PLoS Comput. Biology*, 7(10).