A HIERARCHICAL DYNAMIC MODEL FOR OBJECT RECOGNITION

By
RAKESH CHALASANI

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY UNIVERSITY OF FLORIDA 2013
In memory of my grandfather
ACKNOWLEDGMENTS

I would like to take this opportunity to thank many people who helped and supported my journey to pursue a PhD degree. I thank my advisor, Dr. Jose C. Principe, for giving me the opportunity and means to pursue this work; without his constant support and guidance this work would not have been possible. I am very indebted to Dr. Yunmei Chen for many patient and insightful discussions that helped shape this work significantly and Dr. Anand Rangarajan for stimulating my intellectual curiosity through his course work. I would also like to thank Dr. Sean Meyn and Dr. John Harris for serving on my committee and for many constructive comments.

I feel fortunate to be in company of some amazing people, who made this last five years very memorable. I thank Austin Brockmeier, Rosha Pokharel, Sohan Seth, Luis Sanchez Giraldo and Alexander Singh Alvarado for their friendship, sharing, and many interesting discussions. I thank Evan Kriminger, Matthew Emigh, Erion Hasanbelliu, Kittipat ‘Bot’ Kampa, Goktug Cinar, Stefan Craciun, and Eder Santana for all the fun and laughter, and In Jun Park, Jihye Bae, Songlin Zhao, Miguel Teixeira, Bilal Fadlallah, Gabriel Nallathambi, Pingping Zhu, Jongmin Lee and all the CNELers for the good times. I thank Anusha Vallurupalli and Krishna Chaitanya for being the family away from home.

Finally, I thank my parents for all the love, support and faith, my brother and sister-in-law for constant care and encouragement and Swathi Sri Kondagari for being there and bringing so much joy into my life.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>4</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>8</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>9</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>11</td>
</tr>
<tr>
<td><strong>CHAPTER</strong></td>
<td></td>
</tr>
<tr>
<td>1 INTRODUCTION</td>
<td>13</td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>13</td>
</tr>
<tr>
<td>1.2 Outline</td>
<td>16</td>
</tr>
<tr>
<td>2 THEORETICAL BACKGROUND</td>
<td>18</td>
</tr>
<tr>
<td>2.1 Deep Networks and Representation Learning</td>
<td>18</td>
</tr>
<tr>
<td>2.2 Predictive Coding Models</td>
<td>20</td>
</tr>
<tr>
<td>2.2.1 Hierarchy of Attractors</td>
<td>23</td>
</tr>
<tr>
<td>2.2.2 Relationship with Other Methods</td>
<td>24</td>
</tr>
<tr>
<td>3 FEATURE EXTRACTION FROM TIME-VARYING SIGNALS</td>
<td>27</td>
</tr>
<tr>
<td>3.1 Model and Energy function</td>
<td>27</td>
</tr>
<tr>
<td>3.2 Learning</td>
<td>30</td>
</tr>
<tr>
<td>3.2.1 Inference</td>
<td>31</td>
</tr>
<tr>
<td>3.2.2 Parameter Updates</td>
<td>35</td>
</tr>
<tr>
<td>3.3 Experiments</td>
<td>37</td>
</tr>
<tr>
<td>3.3.1 Inferring Sparse States with Known Parameters</td>
<td>37</td>
</tr>
<tr>
<td>3.3.2 Learning from Natural Image Sequences</td>
<td>41</td>
</tr>
<tr>
<td>3.3.2.1 Visualizing invariance</td>
<td>42</td>
</tr>
<tr>
<td>3.3.2.2 Learning temporal structure</td>
<td>44</td>
</tr>
<tr>
<td>3.3.3 Role of Temporal Connections: Toy data</td>
<td>45</td>
</tr>
<tr>
<td>4 DEEP PREDICTIVE CODING NETWORKS</td>
<td>48</td>
</tr>
<tr>
<td>4.1 Multi-Layered Architecture</td>
<td>48</td>
</tr>
<tr>
<td>4.2 Inference in Multi-Layered Network with Top-Down Connections</td>
<td>50</td>
</tr>
<tr>
<td>4.3 Experiments</td>
<td>52</td>
</tr>
<tr>
<td>4.3.1 Receptive Fields in the Hierarchical Model</td>
<td>52</td>
</tr>
<tr>
<td>4.3.2 Role of Top-Down Information</td>
<td>53</td>
</tr>
<tr>
<td>5 CONVOLUTIONAL DYNAMIC NETWORK FOR LARGE SCALE OBJECT RECOGNITION</td>
<td>57</td>
</tr>
</tbody>
</table>
5.1 Model Architecture ......................................................... 57
  5.1.1 Single Layer Model .................................................... 57
  5.1.2 Building a Hierarchy ................................................ 60
  5.1.3 Implementation Details ............................................. 61
5.2 Inference ............................................................................. 61
  5.2.1 Procedure ............................................................... 62
  5.2.2 Approximate Inference with Top-Down Connections .......... 66
5.3 Learning ............................................................................. 68
5.4 Experiments ......................................................................... 69
  5.4.1 Learning from Natural Video Sequences ......................... 70
  5.4.2 Object Recognition - Caltech-101 dataset ....................... 71
  5.4.3 Recognition With Context .......................................... 72
    5.4.3.1 Role of contextual information during inference .......... 73
    5.4.3.2 Sequential labeling ............................................. 75
    5.4.3.3 Analysis of temporal and top-down connections ......... 80
  5.4.4 Learning Hierarchy of Attractors ................................... 81
    5.4.4.1 Learning parts of object from unlabeled sequences .... 82
    5.4.4.2 Denoising videos using top-down information .......... 82
5.5 Discussion ............................................................................ 86
  5.5.1 Relationship with Feed-Forward Networks ..................... 86
  5.5.2 Comparison with Other Methods .................................... 87
5.6 Summary .............................................................................. 88
6 CONCLUSION ........................................................................... 89
  6.1 Summary ............................................................................ 89
  6.2 Avenues for Future Work ................................................ 91
APPENDIX
A A FAST PROXIMAL METHOD OF CONVOLUTIONAL SPARSE CODING ...... 92
  A.1 Convolutional FISTA and Dictionary Learning .................... 93
    A.1.1 Inference .................................................................. 94
    A.1.2 Dictionary Learning ................................................. 96
    A.1.3 Extension to PSD .................................................. 97
  A.2 Experiments ....................................................................... 98
    A.2.1 Dictionary Learning ................................................. 98
    A.2.2 Convergence Rate .................................................. 99
    A.2.3 Predictive Sparse Decomposition ............................... 100
  A.3 Summary .......................................................................... 101
B ADDITIONAL RESULTS FOR MODEL VISUALIZATION ....................... 102
  B.1 Visualizing Invariance Encoded by Layer 1 Causes ............. 102
  B.2 Hierarchical Decomposition Obtained from the YouTube dataset 104
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-1</td>
<td>Computational time of different methods on synthetic data (per time sample).</td>
<td>38</td>
</tr>
<tr>
<td>5-1</td>
<td>Classification performance over Caltech-101 dataset with only a single bottom-up inference</td>
<td>71</td>
</tr>
<tr>
<td>5-2</td>
<td>Classification performance over COIL-100 dataset with various configurations.</td>
<td>74</td>
</tr>
<tr>
<td>5-3</td>
<td>Recognition rate for face recognition in Honda/UCSD dataset</td>
<td>77</td>
</tr>
<tr>
<td>5-4</td>
<td>Classification performance over YouTube Celebrities dataset</td>
<td>78</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>The visual pathway in the brain responsible for object recognition.</td>
<td>14</td>
</tr>
<tr>
<td>2-1</td>
<td>Comparison between shallow and deep networks.</td>
<td>19</td>
</tr>
<tr>
<td>3-1</td>
<td>A dynamic network consisting of two distinctive processing units: states for extracting features from the inputs and causes that pooling the states to form a more complex representation.</td>
<td>28</td>
</tr>
<tr>
<td>3-2</td>
<td>The combined prior acting on the states in dynamic sparse coding.</td>
<td>29</td>
</tr>
<tr>
<td>3-3</td>
<td>Effect of smoothing on the cost function.</td>
<td>34</td>
</tr>
<tr>
<td>3-4</td>
<td>Performance of dynamic sparse coding with sparse innovation while tracking sparse states.</td>
<td>39</td>
</tr>
<tr>
<td>3-5</td>
<td>Performance of the dynamic sparse coding with varying number of observation dimensions.</td>
<td>40</td>
</tr>
<tr>
<td>3-6</td>
<td>Performance of the dynamic sparse coding with varying (sparse) noise levels.</td>
<td>40</td>
</tr>
<tr>
<td>3-7</td>
<td>Receptive fields of the model at different levels.</td>
<td>41</td>
</tr>
<tr>
<td>3-8</td>
<td>Visualizing the observation matrix learned from natural videos using Gabor fit.</td>
<td>42</td>
</tr>
<tr>
<td>3-9</td>
<td>Visualizing invariance represented by the causes.</td>
<td>43</td>
</tr>
<tr>
<td>3-10</td>
<td>Visualizing the temporal structure learned by the model.</td>
<td>44</td>
</tr>
<tr>
<td>3-11</td>
<td>Role of temporal connections during inference.</td>
<td>46</td>
</tr>
<tr>
<td>4-1</td>
<td>A two-layered hierarchical model constructed by stacking several state-space models.</td>
<td>49</td>
</tr>
<tr>
<td>4-2</td>
<td>Block diagram showing the flow of bottom-up and top-down information in the model.</td>
<td>50</td>
</tr>
<tr>
<td>4-3</td>
<td>Visualization of the receptive fields of the invariant units learned from natural videos.</td>
<td>53</td>
</tr>
<tr>
<td>4-4</td>
<td>Clean and corrupted video sequences constructed using three different shapes.</td>
<td>54</td>
</tr>
<tr>
<td>4-5</td>
<td>The scatter plot of the 3 dimensional causes at the top-layer obtained from the toy dataset.</td>
<td>55</td>
</tr>
<tr>
<td>5-1</td>
<td>Block diagram of a single layer convolutional dynamic network.</td>
<td>58</td>
</tr>
<tr>
<td>5-2</td>
<td>Block diagram of the inference procedure, with arrows indicating the flow of information during inference.</td>
<td>67</td>
</tr>
</tbody>
</table>
5-3 Receptive fields of the two-layered convolutional network learn on natural video sequences. .................................................. 70
5-4 Examples from the COIL-100 dataset. ........................................... 73
5-5 Part of the face sequence belonging to three different subjects extracted from Honda/UCSD dataset. .......................................... 75
5-6 Example sequences from the Youtube celebrities dataset. ............... 76
5-7 Classification performance on noisy Honda/UCSD dataset with 50 frames per sequence. ......................................................... 79
5-8 The performance on noisy Honda/UCSD dataset for various values of $\lambda$ and $\eta$. ................................................................. 80
5-9 Hierarchical decomposition of object parts learned by the model from face videos of 16 different subjects in the VidTIMIT dataset. ................. 83
5-10 Video denoising with temporal and top-down connections. ............... 84
5-11 The PCA projections of layer two causes in the denoising experiment. .. 85
A-1 Learned dictionary and image reconstruction obtained using convolutional FISTA. ................................................................. 98
A-2 Comparison of the convergence rate between convolutional CoD (ConvCoD) and convolutional FISTA (ConvFISTA). ............................. 99
A-3 Encoder weights and instantaneous total loss in predictive sparse decomposition.100
B-1 Visualizing grouping of the dictionary elements encoded by layer 1 causes. . 102
B-2 Visualizing invariance encoded by the layer 1 causes using frequency-orientation polar plot. ...................................................... 103
B-3 Visualizing invariance encoded by the layer 1 causes using center-orientation scatter plot. ......................................................... 104
B-4 Receptive fields in a two-layered network learned from the YouTube celebrities dataset. ......................................................... 105
A HIERARCHICAL DYNAMIC MODEL FOR OBJECT RECOGNITION

By

Rakesh Chalasani

December 2013

Chair: Jose C. Principe
Major: Electrical and Computer Engineering

This work focuses on building a hierarchical dynamic model for object recognition in video. It is inspired from predictive coding framework that is used to explain the working of sensory signal processing in the brain. Using this framework, we propose a new architecture that embodies one of the important characteristics of biological vision; namely, finding a causal representation of the visual inputs while using contextual information coming in from various sources.

The proposed model is a deep network to process an input video sequence in a hierarchical and distributive manner, and includes several innovations. At the core of the model is a dynamic system at each level in the hierarchy encoding time-varying observations. We propose a novel procedure called dynamic sparse coding to infer sparse states of a state-space model and extend it further to model locally invariant representation. These dynamic models endow the network with long term memory (parameters) and short-term contextual information and lead to invariant representation of the observations over time. Another important part of the proposed model is bidirectional flow of information in the hierarchy. The representation at each level in the hierarchy is obtain from data driven bottom-up signals and top-down expectations, which are both driving and modulatory. The top-down expectations are usually task specific, which bias and modulate the representations at the lower layers to extract relevant information from the milieu of noisy observations. We also propose a convolutional
dynamic model that allows us to scale the proposed architecture to large problems. We show that this model decomposes the inputs in a hierarchical manner, starting from low level representations to higher level abstractions, mimicking the processing of information in the early visual cortex. We evaluate the performance of the proposed model on several benchmark data-sets for object and face recognition and show that it performs better than several existing methods.
CHAPTER 1
INTRODUCTION

1.1 Motivation

Recognizing objects and patterns in an environment is one of the important functions of the human visual system and it is capable of doing this remarkably well. However, building a computational model for machine vision that can emulate this functional block of the brain has turned out to be an extremely difficult task. The complexity of this task lies in our ability to deal with large number of variations an object can cast onto a 2D sensory space like images, be it in scale, rotation, position, etc [Pinto et al., 2008]. So, how does the brain solve this complex problem?

From our current understanding, the brain is a large distributed network with several units, called neurons, working in parallel to perform complex tasks like object recognition. Particularly, the visual system, as other parts of cerebral cortex, are arranged as a hierarchical network [Felleman and Van Essen, 1991]. Figure 1-1 shows the “visual pathway” in the brain that is responsible to determine “what” the object is. There is growing convergence within the neuroscience community that the role of this hierarchical distributive network is to maintain some internal representations of the environment and uses these internal representations to constantly predict the time-varying changes in the sensory inputs [Clark, 2013]. Perception (or recognition) can be simply considered measuring how accurate these predictions are. Biological vision uses such internal representations for variety of tasks — recognizing objects, dealing with missing or occluded parts of an object, attention, tracking and segmentation of the scene to name a few. It has two important characteristics that helps it to obtain such rich representations: firstly, it decomposes the inputs in a hierarchical fashion [Felleman and Van Essen, 1991] and secondly, it acts as a dynamic process and uses contextual information very efficiently [Schwartz et al., 2007, Bar, 2004]. In other words, it has the ability to obtain information from various sources and combine
Figure 1-1. The visual pathway in the brain responsible for object recognition. It is made up of four major blocks, viz, V1-V2-V4-IT (Inferior temporal cortex), arranged in a hierarchy. The blocks are interconnected with both forward and backward connections, while LGN projects the external environment onto V1 through the retina.

them at different levels — bottom-up information from the sensory inputs (from LGN in Figure 1-1), spatio-temporal relationships over space and time (through lateral connections are each level) and the top-down expectations coming from the higher levels (V4 and IT in Figure 1-1) carrying prior knowledge gained about the external environment through experience.

These functional characteristics of the brain — namely, the idea of analysis-by-synthesis, the bidirectional flow of information fusing bottom-up and top-down signals, spatio-temporal recurrent connections conveying contextual information — can be modeled, in simple mathematical terms, as a generative model [Friston, 2005]. Predictive coding, proposed by Rao and Ballard [1997] and Friston [2008], explores this idea of using a generative model and provides a unified mathematical framework. In this work, inspired by predictive coding, we propose a novel hierarchical dynamic model for robust object recognition in video sequences that is consistent with the biological vision.

Keeping in mind these properties of the the visual system, we restrict our model to have certain important properties:
• the model has to consider the inputs as a dynamic process, i.e., it has to maintain the temporal relationships during inference. This leads to incorporating contextual information into the model and may be beneficial for robust recognition [Schwartz et al., 2007].

• it has to decompose the inputs in a hierarchical and distributive fashion. Such “deep” representation should extract abstract information from the inputs, leading to better generalization and robustness to several variations in the inputs [Bengio, 2009].

• it has to maintain sparsity at every level. This helps not only to obtain efficient representation of the observations but also for learning better discriminability [Rigamonti et al., 2011].

• it should combine the top-down feedback information from the higher layers in the hierarchy during bottom-up inference. Such bidirectional information flow can be helpful at the lower layers to reconstruct the missing parts of the object or disambiguate between different objects in a noisy environment [Gilbert and Li, 2013].

Previously, many methods are proposed to model visual perception and they encompass some of the above mentioned properties. For example, deep learning methods — like deep belief networks [Hinton et al., 2006], stacked auto-encoders [Vincent et al., 2010], HMAX [Riesenhuber and Poggio, 1999][Serre et al., 2007], convolutional neural networks [LeCun et al., 1989], etc — encode the inputs as a hierarchical representation. It is observed that increasingly invariant representations can be obtained with the depth of the hierarchical models [Goodfellow et al., 2009]. These models focus mainly on “rapid” object recognition with feed-forward networks, without considering neither temporal nor top-down connections. On the other hand, temporal coherence was found to be useful to learn complex invariances from the data, see [Wiskott and Sejnowski, 2002, Mobahi et al., 2009, Zou et al., 2012]. Also, models like deep Boltzmann machine [Salakhutdinov and Hinton, 2012], convolutional deep belief networks [Lee et al., 2009] consider the top-down influences using undirected graphical models.

Rao and Ballard [Rao and Ballard, 1997] previously proposed a predictive coding model based on Kalman filter like update rules, while Firston [Friston, 2008] proposed
a similar hierarchical dynamic model with generalized coordinates based on empirical
priors and presented a unified theory based on free energy principles. Lee and Mumford
[Lee and Mumford, 2003] proposed a particle filtering based approach to build a model
for visual perception. Though these models are successful in incorporating top-down
connections and temporal relationships, neither of these models consider sparsity on
the hierarchical representations and does not scale well to large images and videos. In
spite of these limitation, predictive coding models are able to explain the importance of
context in sensory perception [Rao and Ballard, 1999, Kiebel et al., 2008a, 2009] and
make it possible to explore the use of context (or empirical priors) in deep networks for
visual object recognition.

From machine vision point of view, imposing some prior knowledge on the model
can lead to better representation of the inputs. Such prior knowledge can be domain
specific — as in SIFT [Lowe, 1999], HOG [Dalal and Triggs, 2005] — or can be the
use of generic fixed priors like sparsity [Olshausen and Field, 1996] and temporal
coherence [Wiskott and Sejnowski, 2002]. The use these fixed priors has become
particularly useful while learning deep networks [Zou et al., 2012, Lee et al., 2007a,
Kavukcuoglu et al., 2010a]. However, contextual information might contain more task
or objective specific information that can not easily be incorporated while using fixed
priors. So, one way to incorporate this information is by empirically altering the priors on
the model depending on the context. In this work, we use this idea of empirical priors
to incorporate contextual information obtained from the temporal relationships in the
video and top-down abstract information about the objects to build a robust recognition
system.

1.2 Outline

Following is the outline of the rest of the dissertation: in chapter 2 we discuss the
theoretical background for the rest of this work. In this chapter, we first discuss the
importance of learning representations is a distributive and hierarchical manner. Then
we motivate the idea of predictive coding as unifying framework and discuss the idea of building a hierarchical dynamic model. We review several existing methods and their the relationship with the proposed model.

In chapter 3, we propose dynamic sparse coding (DSC), where the observation sequences are modeled using a dynamic system with appropriate sparsity constraint on the states and the innovations. We further extend this to model invariant representation of the observations. This forms the basic block for building a hierarchical model later.

In chapter 4 we propose deep predictive coding network (DPCN), which is a hierarchical dynamic model, with the state-space model developed in chapter 3 as the building block. We propose an efficient approximate inference procedure in this hierarchical network, that combines the bottom-up data driven information coming into the lower layers with the top-down expectation from the higher layers.

In chapter 5, we propose a convolutional model for the dynamic networks discussed in the aforementioned chapters. This allows us to scale the proposed architecture to large scale images/videos. Here we compare the proposed model on several benchmark object and face recognition datasets and show that it performs better than several existing methods.

Finally, in chapter 6, we conclude the dissertation by summarizing the contributions of this work and discuss some future avenues to explore.
CHAPTER 2
THEORETICAL BACKGROUND

In this chapter, we first consider the problem of visual object recognition from a machine learning perspective and understand the complexity of the task. We discuss the importance of hierarchical and distributive nature of the representations to perform better classification and argue that such representations can disentangle the underlying discriminative factors of an object from low-level sensory inputs. We then focus on such framework called predictive coding to construct a hierarchical network. It provides a unified theory to construct a “deep” dynamic models and forms the theoretical underpinning for the rest of the models proposed in this work.

2.1 Deep Networks and Representation Learning

One the important pre-requisite for good classification is to find a representation of the data such that the classification becomes easier in that projected space. For example, consider a simple two class problem shown in Figure 2-1A, where the samples in a 2D space belonging to two different classes are not linearly separable. Generally, popular learning algorithms—like kernel (or non-linear) support vector machines [Vapnik, 1998], radial basis network [Haykin, 1994], multilayer perceptron with single hidden layer [Haykin, 1994] — find a non-linear mapping such that these samples are well separated in some projected space. Such models which perform a single non-linear transformation on the inputs are called as shallow networks. Though these methods perform relatively well in simple tasks like above, such models have limited capacity when variations within a class are large and requires to learn some complex mapping to find a good representations [Bengio and LeCun, 2007], such as in the case of object recognition. The reason being, as complexity of the decision boundary between classes becomes large, shallow networks requires more number of computational units to represent it. In fact, Bengio [2009] has shown theoretically that such shallow models
Figure 2-1. Comparison between shallow and deep networks. (A) Shallow networks use a single non-linear function to map the observations into a space where they are linearly separable. (B) However, when the decision boundary becomes complicated (as in case of two class object recognition shown here), a single non-linear mapping might not suffice. Deep networks use multiple non-linear transformations on the observations and progressively extract abstract information. After these multiple non-linear mappings, different classes become more distinguishable.

requires exponentially large number of computational units and memory proportional to the complexity of the tasks.

Instead, deep networks that use multiple hierarchical non-linear mappings are shown to have much more flexibility to model complex mappings and in a compact manner [Bengio and LeCun, 2007, Hinton and Salakhutdinov, 2006]. The key to such models is their distributive and hierarchical nature while representing the inputs/observations. The observations are represented such that simple features
extracted at the lower layers are combined in the higher layer through complex relationships and become progressively more abstract with the depth of the model (Figure 2-1B).

The important point here is the re-usability of the lower layer feature extractors for creating distinctive and abstract higher layer features. Theoretically, every distinct path traversing from low-level features to the higher level form a different representation. Such re-usability of feature extractors common for several distinctive classes not only lead to compact representations but also better generalization for some unknown classes.

But, how do we know these complex higher-level relationships? As we will show later in this work, such relationships can be learned from the observations themselves, called as representation learning. Representation learning allows us to leverage the structure present in the data, even when there is no label information [Bengio et al., 2013]. It is already shown [Hinton et al., 2006, Lee et al., 2009, Jarrett et al., 2009a, Boureau et al., 2010] to outperform some handcrafted techniques like SIFT [Lowe, 1999], HOG [Dalal and Triggs, 2005]. Another advantage of the representation learning is its use in transfer learning, where it allows the model learned on a particular data (or task) to easily generalize to another data (or tasks).

### 2.2 Predictive Coding Models

This idea of deep and distributive representations discussed above is in fact loosely inspired from working of visual cortex (or neocortex, in general) in the brain. As discussed in Chapter 1, visual cortex uses similar multiple layers of non-linear transformations on the sensory inputs/observations. We also discussed how generative models could potentially be used design such hierarchical models. The key assumption here is that the sensory signal coming from the external environment unfold as a dynamic process with a distinctive causal “event”. One of the most prominent theories for modeling such sensory inputs, following the idea of Helmholtz, is to consider “the
perceptual system as an inference engine whose function is to infer the probable causes of the sensory input” [Dayan et al., 1995]. In other words, the goal of perception is to solve an inverse problem to find the underlying signal that might have “caused” the sensory inputs. To elaborate, one would like to build a generative model that tries to learn the external environment using a set of parameters. Using these parameters, the model tries to then infer the underlying cause that might have “generated” the observations. This idea of analyzing the observations by synthesis is shown to be very useful and several methods were proposed to solve this inverse problem [Friston, 2005, Riesenhuber and Poggio, 1999, Dayan et al., 1995, George and Hawkins, 2005]. Here we are interested in one such framework proposed in [Friston, 2005, Rao and Ballard, 1999], called predictive coding (or empirical Bayes).

The basic idea of predictive coding is to model the visual system as a generative model that tries to predict the external responses using some internal states. Then the prediction error, from a generative model perspective, is the difference between the actual observation and the “generated” input from the model and the underlying causes (also called as latent or hidden variables) [Friston, 2005]. Mathematically, if \( y_t \) is an observation at time \( t \), then it is described by the underlying cause \( u_t \) as follows:

\[
y_t = \mathcal{F}(u_t) + v_t
\]

(2–1)

where \( \mathcal{F}(.) \) is some observation (or measurement) function. In addition, since we assume that the observations are time-varying, an intermediate hidden states \( x_t \) can be considered to encode the dynamics over time. Hence, a unified model that encodes a sequence of observations can be written as a generalized state-space model of the form [Friston, 2008] :

\[
y_t = \mathcal{F}(x_t, u_t) + n_t
\]

\[
x_t = \mathcal{G}(x_{t-1}, u_t) + v_t
\]

(2–2)
where $G$ is called state transition function. We assume that $F$ and $G$ can be parameterized by some set of parameters, $\theta$. The terms $u_t$ are called the unknown causes and, since we are usually interested in obtaining abstract information from the observations, they are encouraged to have a non-linear relationship with the observations. The hidden states, $x_t$, then “mediate the influence of the cause on the observations and endow the system with memory” Friston [2008]. The terms $v_t$ and $n_t$ are stochastic and model uncertainty in the predictions.

Now, to build a multi-layer hierarchical network, several such state-space models can be stacked such that the output from one acts as observations to the model in the layer above. Mathematically, an $L$-layered network of this form can be written as:

\[
\begin{align*}
y_t &= f(x^{(1)}_t, u^{(1)}_t) + n^{(1)}_t \\
x^{(1)}_t &= g(x^{(1)}_{t-1}, u^{(1)}_t) + v^{(1)}_t \\
&\vdots \\
u^{(l-1)}_t &= f(x^{(l)}_t, u^{(l)}_t) + n^{(l)}_t \\
x^{(l)}_t &= g(x^{(l)}_{t-1}, u^{(l)}_t) + v^{(l)}_t \\
&\vdots \\
u^{(L-1)}_t &= f(x^{(L)}_t, u^{(L)}_t) + n^{(L)}_t \\
x^{(L)}_t &= g(x^{(L)}_{t-1}, u^{(L)}_t) + v^{(L)}_t
\end{align*}
\]

The terms $v^{(l)}_t$ and $n^{(l)}_t$ are stochastic fluctuations at the higher layers and enter each layer independently. In other words, this model forms a Markov chain across the layers and the latent variables at any layer are now only dependent on the observations coming from the layer below and the predictions from the layer above. Notice how the causes at lower layer form the “observations” to the layer above, i.e., the causes form the link between the layers while the states link the dynamics over time.
The important point in this design is that the higher-level predictions influence the representation at lower levels. These predictions from the higher layer non-linearly enter into the state space model at a bottom layer by empirically altering the prior on the causes of the bottom layer. Hence, the top-down connections, along with the horizontal (or recurrent) connections in the state space, directly influence the inference in the bottom layers. As we will discuss later, these two connections together provides contextual-awareness to the model while encoding the observations.

Learning in this hierarchical generative model is tantamount to building internal models that can explain observations in terms of some underlying causes. Paraphrasing the same, if the model is able to synthesize an observation accurately, then it means the model has previously seen a similar observation. Recognition then is simply solving an inverse problem by jointly minimizing the prediction error at all levels [Friston, 2005]. This framework form the theoretical underpinning of the methods proposed in this work.

2.2.1 Hierarchy of Attractors

So, how does this dynamic (or predictive coding) model related to hierarchical representations discussed in Section 2.1? The key idea while designing this dynamic generative model is that the observations coming into the model “unfold as an ordered sequence of spatio-temporal dynamics” [George and Hawkins, 2005, Kiebel et al., 2008b], and these are generated as trajectories in some underlying attractor manifolds, while the causes govern the position of the attractors in the manifold [Friston and Kiebel, 2009]. Critically, in a hierarchical setting, the shape of the attractors in the lower-layers is governed by other dynamic models in the higher layers, which themselves can have their own attractors. This coupling across the layers in the hierarchy allows the dynamics in the higher-layers influence the lower-layer “trajectories”. From object recognition perspective, such hierarchical dynamic models have some important characteristics [Friston and Kiebel, 2009]
Firstly, since we assume that the dynamic models at each layer have their own attractors, each can “generate” and hence, encode, its own spatio-temporal sequence. While the dynamics within each model determines “where” in the trajectory the sequence is from, the shape determines “which” attractor is encoding the observations. In other words, each dynamic model at any layer encode a particular shape (or part of the object) and is invariant to certain transformations (or “fluctuations”) of that shape.

Secondly, as higher layer models influence the manifold of the lower layer attractors, the higher layers encode the category of the sequence represented by the lower layers. That is to say, it is possible to generate sequence of sequences, as the causes in the higher layer control how the attractors in the lower-layer manifold are “mixed” together (This is equivalent to re-using the lower layer feature extractors described in Section 2.1). More important, while encoding, the higher layers determine (rather, guide the lower layers to) which sequence (or object) to encode in miliue of noisy observations. This entitles a non-linear interaction of the top-down effect between the higher layer attractor and lower layer manifold.

Finally, the higher layers encode more abstract knowledge and evolve more slowly over time. The reason being, as the lower layers are guided by the higher layers to encode a particular object, the lower layer encode the transformation in the object and pass on only an invariant representation to the layer above. Since the objects in a scene change slowly than the transformation they undergo, the the higher layers are more invariant and, hence, much slower to evolve over time. In the following chapters, we show that proposed model encompass all these characteristics and lead to a robust object recognition system.

2.2.2 Relationship with Other Methods

One can also argue that the above state-space model in (2–2), and its corresponding hierarchical model (2–3), encompasses several existing methods that are popular for object recognition [Friston, 2008, Roweis and Ghahramani, 1999]. The ability of these
models to generalize several models comes from considering different priors and design while encoding the inputs. For example, sparse coding [Olshausen and Field, 1996], which is popularly used in object recognition, can be considered as special case of the state space model in (2–2), when we do not have the hidden states and a sparsity inducing prior is considered over the causes. Even more complex methods like independent subspace analysis (ISA) (or topographic ICA) [Hyvärinen et al., 2009] can also be considered as a case where we have both the hidden states and causes, without any dynamical state-space equation. Similarly, many other popular deep networks used for object recognition contains max or average pooling functions along with feature extraction methods [Riesenhuber and Poggio, 1999, Jarrett et al., 2009b]. This can also be subsumed into the proposed model by considering the states encode the features, while the non-linear function between the states and the causes is simply a max or averaging pooling function.

Recently, restricted Boltzmann machine [Hinton et al., 2006], auto-encoders [Bengio et al., 2007], encoder-decoder models [Ranzato et al., 2007], etc have become popular to build deep networks. The key to these models is to learn both encoding and decoding concurrently and build the deep network as a feed forward model using only the encoder while discarding the decoder. Though these models appear to be different from the hierarchical models described before, the function of the encoder is only to approximate the decoding (or inference) and hence, can be subsumed in the above framework as well. (However, see [Salakhutdinov and Hinton, 2012, Lee et al., 2009] that constructs deep Boltzmann machine using undirected graphical models, which can also process bidirectional information flow in the hierarchical models. Also see [Bengio et al., 2013] for more thorough comparison between directed and undirected graphical models for learning deep networks.)

We would also like remark that there are several popular handcrafted features, like SIFT [Lowe, 1999], SURF [Bay et al., 2008], HOG [Dalal and Triggs, 2005], etc.
that are very successful in object recognition. These methods do not fall into adaptive learning paradigm and hence, are not biologically plausible. But they are very successful in object recognition tasks and are worth taking a note of.
CHAPTER 3
FEATURE EXTRACTION FROM TIME-VARYING SIGNALS

In this chapter, we consider a dynamic network to extract features from a small part of a video sequence. This forms the basic block for building hierarchical models later. The centerpiece of the proposed model is extracting sparse features from time-varying observation sequences using a dynamic model. To this end, we propose a novel procedure based on proximal methods to infer sparse states (or features) of a dynamic system. We then extend this feature extraction (or sparse coding) block to introduce a pooling strategy to learn invariant feature representations from the data. We show that this two stage model first extracts simple features and then combine them to form more complex representations, similar to simple and complex cells in the V1 regions of the visual cortex.

3.1 Model and Energy function

To begin with, let \( \{y_1, y_2, \ldots, y_t, \ldots\} \in \mathbb{R}^P \) be a \( P \)-dimensional sequence of a patch extracted from the same location across all the frames in a video\(^1\). To process this, our network consists of two distinctive parts (Figure 3-1): feature extraction (inferring states) and pooling (inferring causes).

For the first part, sparse coding is used in conjunction with a linear state space model to map the inputs \( y_t \) at time \( t \) onto an over-complete dictionary of \( K \)-filters, \( C \in \mathbb{R}^{P \times K} (K > P) \), to get sparse states \( x_t \in \mathbb{R}^K \). To keep track of the dynamics in the latent states we use a linear function with state-transition matrix \( A \in \mathbb{R}^{K \times K} \). More formally, we assume that the inputs are synthesized using the following generative

\[^1\) Here \( y_t \) is a vectorized form of \( \sqrt{P} \times \sqrt{P} \) square patch extracted from a frame at time \( t \).\)
We call this model dynamic sparse coding (DSC). To infer the states $x_t$ in this model, we minimizes the following energy function:

$$E_1(x_t, y_t, C, A) = \|y_t - Cx_t\|_2^2 + \lambda\|x_t - Ax_{t-1}\|_1 + \gamma\|x_t\|_1$$

(3–2)

Notice that the second term involving the state-transition is also constrained to be sparse, implying that the number of change in the features over time is small. This not only makes sense in practice where visual inputs we usually encounter are slowly changing over time, but also makes the state-space representation more consistent and leads to more sparser solution. Figure 3-2 shows the comparison between modeling the innovations as sparse versus “dense” (or Gaussian distribution) (similar to Kalman filtering but without updating the covariance matrix over time). Notice that the shape of the combined regularizer over the states around the solution is sharper with sparse innovations, indicating that it promotes better sparsity than when the innovations are modeled as a Gaussian distribution.
Figure 3-2. The combined prior acting on the states in dynamic sparse coding. (A) In case of Gaussian innovations, the combined prior is smooth around the solution and might not maintain the sparsity of the output solution. (B) In case of sparse innovations, the combined prior acting in the states is sharper and leads to a much sparser solution.

Now, to take advantage of the spatial relationships in a local neighborhood, a small group of states $x_t^{(n)}$, where $n \in \{1, 2, \ldots, N\}$ represents a set of contiguous patches w.r.t. the position in the image space, are added (or sum pooled) together. Such pooling of the states may be lead to local translation invariance. On top this, a $D$-dimensional causes $u_t \in \mathbb{R}^D$ are inferred from the pooled states to obtain representation that is invariant to more complex local transformations like rotation, spatial frequency, etc. In line with [Karklin and Lewicki, 2005], this invariant function is learned such that it can capture the dependencies between the components in the pooled states. Specifically,
the causes $u_t$ are inferred by minimizing the energy function:

$$E_2(u_t, x_t, B) = \sum_{n=1}^{N} \left( \sum_{k=1}^{K} \left| \gamma_k \cdot x_{t,k}^{(n)} \right| \right) + \beta \|u_t\|_1$$  \hspace{1cm} (3–3)$$

$$\gamma_k = \gamma_0 \left[ \frac{1 + \exp(-B_{t,k})}{2} \right]$$

where $\gamma_0 > 0$ is some constant. Notice that here $u_t$ multiplicatively interacts with the accumulated states through $B$, modeling the shape of the sparse prior on the states. Essentially, the invariant matrix $B$ is adapted such that each component $u_t$ connects to a group of components in the accumulated states that co-occur frequently. In other words, whenever a component in $u_t$ is active it lowers the coefficient of a set of components in $x_t^{(n)}, \forall n$, making them more likely to be active. Since co-occurring components typically share some common statistical regularity, such activity of $u_t$ typically leads to locally invariant representation [Karklin and Lewicki, 2005].

Though the two cost functions are presented separately above, we can combine both to devise a unified energy function of the form:

$$E(x_t, u_t, \theta) = \sum_{n=1}^{N} \left( \frac{1}{2} \|y_t^{(n)} - Cx_t^{(n)}\|_2^2 + \lambda \|x_t^{(n)} - Ax_t^{(n)}\|_1 + \sum_{k=1}^{K} \left| \gamma_{t,k} \cdot x_{t,k}^{(n)} \right| \right) + \beta \|u_t\|_1$$  \hspace{1cm} (3–4)$$

$$\gamma_{t,k} = \gamma_0 \left[ \frac{1 + \exp(-B_{t,k})}{2} \right]$$

where $\theta = \{A, B, C\}$. As we will discuss next, both $x_t$ and $u_t$ can be inferred concurrently from (3–4) by alternatively updating one while keeping the other fixed using an efficient proximal gradient method.

### 3.2 Learning

To learn the parameters in (3–4), we alternatively minimize $E(x_t, u_t, \theta)$ using a procedure similar to block co-ordinate descent. We first infer the latent variables $(x_t, u_t)$ while keeping the parameters fixed and then update the parameters $\theta$ while keeping the variables fixed. This is done until the parameters converge. We now discuss separately
the inference procedure and how we update the parameters using a gradient descent method with the fixed variables.

### 3.2.1 Inference

We jointly infer both $x_t$ and $u_t$ from (3–4) using proximal gradient methods, taking alternative gradient descent steps to update one while holding the other fixed. In other words, we alternate between updating $x_t$ and $u_t$ using a single update step to minimize $E_1$ and $E_2$, respectively. However, updating $x_t$ is relatively more involved. So, keeping aside the causes, we first focus on inferring sparse states alone from $E_1$, and then go back to discuss the joint inference of both the states and the causes.

**Inferring States**

Inferring sparse states, given the parameters, from a linear dynamical system forms the crux of our model. This is performed by finding the solution that minimizes the energy function $E_1$ in (3–2) with respect to the states $x_t$ (while keeping the sparsity parameter $\gamma$ fixed). Here there are two priors of the states: the temporal dependence and the sparsity term. Although this energy function $E_1$ is convex in $x_t$, the presence of two non-smooth terms makes it hard to use standard optimization techniques used for sparse coding alone.

Previously, many methods are proposed to explore the possibility of using dynamics to recover the sparse, time-varying signals. Notably, some modifications to the Kalman filter are proposed based on selecting a constrained subset of basis [Vaswani, 2008] or using hierarchical Bayesian sparse coding [Karseras et al., 2013]. Others addressed the problem using dynamic programming [Angelosante et al., 2009], homotopy [Charles et al., 2011], modified compressive sensing problem [Vaswani and Lu, 2010] or modeling the state innovations as Gauss-Bernoulli signal while using sampling methods Sejdinovic et al. [2010]. Finally, Charles and Rozell [2012] proposed re-weighted $\ell_1$ dynamic filtering (RWL1-DF), where dynamics over time are used to model the weighted sparsity prior on the states. However, it requires performing an $\ell_1$ optimization multiple
times for each time instance. Also, as we will show later, it also becomes unstable when
the noise in the observations becomes large. To summarize, the optimization used in all
these methods is either computationally expensive and/or unstable for use in large scale
problems like object recognition.

In this work, inspired by the method proposed by Chen et al. [2012a] for structured
sparsity, we propose a smooth proximal gradient method that can approximate
the energy function $E_1$ and able to use efficient solvers like fast iterative shrinkage
thresholding algorithm (FISTA) [Beck and Teboulle, 2009]. The key to our approach
is to first use Nesterov’s smoothness method [Nesterov, 2005] to approximate the
non-smooth state transition term. The resulting energy function is a convex and
continuously differentiable function in $x_t$ with a sparsity constraint, and hence, can
be efficiently solved using proximal methods like FISTA.

**Smooth approximation of sparse innovations:**

To begin, let $\Omega(x_t) = \|e_t\|_1$ where $e_t = (x_t - Ax_{t-1})$. The idea is to find a smooth
approximation to this function $\Omega(x_t)$ in $e_t$. Notice that, since $e_t$ is a linear function on $x_t$,
the approximation will also be smooth w.r.t. $x_t$. Now, we can re-write $\Omega(x_t)$ using the
dual norm of $\ell_1$ as

$$\Omega(x_t) = \arg \max_{\|\alpha\|_\infty \leq 1} \alpha^T e_t$$

where $\alpha \in \mathbb{R}^k$. Using the smoothing approximation from Nesterov [2005] on $\Omega(x_t)$:

$$\Omega(x_t) \approx f_\mu(e_t) = \arg \max_{\|\alpha\|_\infty \leq 1} [\alpha^T e_t - \mu d(\alpha)]$$

(3–5)

where $d(\cdot) = \frac{1}{2}\|\alpha\|_2^2$ is a smoothing function and $\mu$ is a smoothness parameter.

From Nesterov’s theorem [Nesterov, 2005], it can be shown that $f_\mu(e_t)$ is convex and
continuously differentiable in $e_t$ and the gradient of $f_\mu(e_t)$ with respect to $e_t$ takes the
form

$$\nabla_{e_t} f_\mu(e_t) = \alpha^*$$

(3–6)
where $\alpha^*$ is the optimal solution to $f_\mu(e_t) = \arg \max_{\|\alpha\|_\infty \leq 1} [\alpha^T e_t - \mu d(\alpha)]$. This optimal solution of $\alpha$ in (3–5) is given by

$$
\alpha^* = \arg \max_{\|\alpha\|_\infty \leq 1} [\alpha^T e_t - \frac{\mu}{2}\|\alpha\|^2] = \arg \min_{\|\alpha\|_\infty \leq 1} \|\alpha - \frac{e_t}{\mu}\|^2 = S\left(\frac{e_t}{\mu}\right)
$$

(3–7)

where $S(\cdot)$ is a function projecting $\left(\frac{e_t}{\mu}\right)$ onto an $\ell_\infty$-ball. This is of the form:

$$
S(x) = \begin{cases} 
 x, & -1 \leq x \leq 1 \\
 1, & x > 1 \\
 -1, & x < -1 
\end{cases}
$$

Now, by using the chain rule and since $f_\mu(e_t)$ is also convex and continuously differentiable in $x_t$, the gradient of $f_\mu(e_t)$ w.r.t $x_t$ also turns out to the same.

**Effect of smoothing:**

To visualize the effect of the above described smoothing operation, we plot the function $f_\mu(e_t)$ for a one-dimensional error signal $e_t \in \mathbb{R}$ for various values of $\mu$. Note that $\mu$ determines the maximum value of $\alpha$ in (3–5) ($\alpha^*$) corresponding to each error value. Figure 3-3 shows the resulting plots. As it indicates, the sharp point in $\ell_1$-norm around the origin is smoothed in the approximated function $f_\mu(e_t)$. Also note that, as the value of $\mu$ increases, the approximation, though smoother, starts to deviate more from the $\ell_1$-norm. In fact, one can show that, given the desired accuracy $\epsilon$ of the solution, following convergence results from Theorem 2 in Chen et al. [2012a] suggests $\mu = \frac{\epsilon}{k}$, where $k$ is the dimensions of the states, leads to the best convergence rate. We refer the reader to Chen et al. [2012a] for details.

**Smoothing proximal gradient descent for DSC:**
Fig. 3-3. Effect of smoothing on the cost function. Plot shows the smooth function $f_\mu(e_t)$ versus a one dimensional error signal $e_t$ for various values of the smoothness parameter $\mu$.

With this smoothing approximation, the overall cost function from (3–2) can now be re-written as

$$x_t = \arg\min_{x_t} \frac{1}{2}\|y_t - Cx_t\|_2^2 + \lambda f_\mu(e_t) + \gamma\|x_t\|_1$$  \hspace{1cm} (3–8)

with the smooth part $h(x_t) = \frac{1}{2}\|y_t - Cx_t\|_2^2 + \lambda f_\mu(e_t)$ whose gradient with respect to $x_t$ is given by

$$\nabla_{x_t} h(x_t) = C^T(y_t - Cx_t) + \lambda \alpha^*$$  \hspace{1cm} (3–9)

Using the gradient information in (3–9), we solve for $x_t$ from (3–8) using FISTA [Beck and Teboulle, 2009].

Inferring Causes

Given a group of state vectors, $u_t$, can be inferred by minimizing $E_2$, where we define a generative model that modulates the sparsity of the pooled state vector, $\sum_n |x^{(n)}|$. Here we observe that FISTA can be readily applied to infer $u_t$, as the smooth part of the
function $E_2$:

$$h(u_t) = \sum_{k=1}^{K} \left( \gamma_0 \left[ \frac{1 + \exp(-[Bu_t]_k)}{2} \right] \cdot \sum_{n=1}^{N} |x_{t,k}^{(n)}| \right)$$  \hspace{1cm} (3–10)

is convex, continuously differentiable. We note here that the matrix $B$ is initialized with non-negative entries and continues to be non-negative without any additional constraints [Gregor and LeCun, 2011]. This allows the gradient of $h(u_t)$, given by:

$$\nabla_{u_t} h(u_t) = -B^T \sum_{k=1}^{K} \left( \gamma_0 \left[ \frac{\exp(-[Bu_t]_k)}{2} \right] \cdot \sum_{n=1}^{N} |x_{t,k}^{(n)}| \right)$$  \hspace{1cm} (3–11)

to be Lipschitz continuous and hence, guarantees convergence with a bound on the convergence rate of the solution [Beck and Teboulle, 2009].

**Joint Inference**

We showed thus far that both $x_t$ and $u_t$ can be inferred from their respective energy functions using a first-order proximal method called FISTA. However, for joint inference we have to minimize the combined energy function in (3–4) over both $x_t$ and $u_t$. We do this by alternately updating $x_t$ and $u_t$ while holding the other fixed and using a single FISTA update step at each iteration. It is important to point out that the internal FISTA step size parameters are maintained between iterations. This procedure is equivalent to alternating minimization using gradient descent. Although this procedure no longer guarantees convergence of both $x_t$ and $u_t$ to the optimal solution, in all of our simulations it lead to a reasonably good solution. Please refer to Algorithm. 1 for details. Note that, with the alternating update procedure, each $x_t$ is now influenced by the feed-forward observations, temporal predictions and the feedback connections from the causes.

**3.2.2 Parameter Updates**

With $x_t$ and $u_t$ fixed, we update the parameters by minimizing $E$ in (3–4) with respect to $\theta$. Since the inputs here are a time-varying sequence, the parameters are updated using dual estimation filtering [Nelson, 2000]; i.e., we put an additional
Algorithm 1 Updating $x_t, u_t$ simultaneously using FISTA-like procedure.

Require: Take $L_{0,n}^x > 0 \forall n \in \{1, 2, ..., N\}$, $L_{0}^u > 0$ and some $\eta > 1$.

1: Initialize $x_{0,n} \in \mathbb{R}^K \ \forall n \in \{1, 2, ..., N\}$, $u_0 \in \mathbb{R}^D$ and set $\xi_1 = u_0$, $z_{1,n} = x_{0,n}$.

2: Set step-size parameters: $\tau_1 = 1$.

3: while no convergence do

4: Update

\[ \gamma = \gamma_0(1 + \exp(-[B u_i]))/2 \]

5: for $n \in \{1, 2, ..., N\}$ do

6: Line search: Find the best step size $L_{k,n}^x$.

7: Compute $\alpha^*$ from (3–7).

8: Update $x_{i,n}$ using the gradient from (3–9) with a soft-thresholding function.

9: Update internal variables $z_{i+1}$ with step size parameter $\tau_i$ as in [Beck and Teboulle, 2009].

10: end for

11: Compute $\sum_{n=1}^{N} |x_{i,n}|$.

12: Line search: Find the best step size $L_{k}^u$.

13: Update $u_{i,n}$ using the gradient of (3–10) with a soft-thresholding function.

14: Update internal variables $\xi_{i+1}$ with step size parameter $\tau_i$ as in [Beck and Teboulle, 2009].

15: Update

\[ \tau_{i+1} = \frac{\left(1 + \sqrt{(4\tau_i^2 + 1)}\right)}{2} \]

16: Check for convergence.

17: $i = i + 1$

18: end while

19: return $x_{i,n}$ $\forall n \in \{1, 2, ..., N\}$ and $u_i$

constraint on the parameters such that they follow a state space equation of the form:

\[ \theta_t = \theta_{t-1} + z_t \quad (3-12) \]

where $z_t$ is Gaussian transition noise over the parameters. This keeps track of their temporal relationships. Along with this constraint, we update the parameters using
gradient descent. Notice that with fixed $x_t$ and $u_t$, each of the parameter matrices can be updated independently, whose gradient is obtained as follows:

$$
\nabla_A E(\cdot) = \text{sign}(x_t - A_t x_{t-1}) x_t^T + \zeta(A_t - A_{t-1})
$$

$$
\nabla_C E(\cdot) = (y_t - C_t x_t) x_t^T + \zeta(C_t - C_{t-1})
$$

$$
\nabla_B E(\cdot) = (\exp\{-[B u_t]\} \cdot x_t) u_t^T + \zeta(B_t - B_{t-1})
$$

(3-13)

where $\zeta$ acts as a forgetting factor. Matrices $C$ and $B$ are column normalized after the update to avoid any trivial solution.

**Mini-Batch Update:** To get faster convergence, the parameters are updated after performing inference over a large sequence of inputs instead of at every time instance. With this “batch” of signals, more sophisticated gradient methods, like conjugate gradient, can be used and, hence, can lead to more accurate and faster convergence.

### 3.3 Experiments

#### 3.3.1 Inferring Sparse States with Known Parameters

We consider an experimental set-up similar to one used by Charles and Rozell [2012] with synthetic data and compare the performance of the proposed dynamic sparse coding (DSC) with other methods—sparse coding using FISTA (SC) [Beck and Teboulle, 2009], Kalman filter [Kalman, 1960], re-weighted $\ell_1$ dynamic filtering (RWL1-DF) [Charles and Rozell, 2012]. We also compare our method while considering the states innovations in (3–2) as Gaussian (SC-L2 Innov.), as depicted in Figure 3-2.

Specifically, the experimental set-up is as follows: we simulate a state sequence with only 20 non-zero elements in a 500-dimensional state vector evolving with a permutation matrix (note that this keeps the number of non-zero elements same over time), which is

---

We perform inference on this model using FISTA [Beck and Teboulle, 2009]
different for every time instant. This state sequence is then passed through a Gaussian scaling matrix to generate a sequence of observations. We vary observation dimensions \( (\rho) \) depending on the experiment, which will be specified later. We consider that both the permutation and the scaling matrices are known \textit{apriori}. The observation noise is Gaussian with zero mean and variance \( \sigma^2 = 0.001 \). We consider sparse state-transition noise, which is simulated by choosing a subset of active elements \( (n) \) in the state vector chosen randomly and switching each of them with a randomly chosen element (with uniform probability over the state vector). This resemble a sparse innovation in the states with \( 2n \) number of wrongly placed elements, one “missing” element and one “additional” element. We use these generated observation sequences as inputs and use the \textit{apriori} known parameters to infer the states \( x_t \). To set the hyper-parameters, we perform a parameter sweep to find the best configuration for each method. We compare the inferred states from different methods with the true states in terms of relative mean squared error (rMSE); defined as

\[
\frac{\|x_t^{\text{est}} - x_t^{\text{true}}\|}{\|x_t^{\text{true}}\|}
\]

Figure 3-4 shows the tracking performance of different methods — see caption for details about the model used. Also, Table 3-1 shows the computation time (per time instance) for all the methods\(^3\). We observe that the dynamic sparse coding (DSC) is able to track the states over time more accurately than sparse coding (SC), which does not use any dynamics. The dynamic model with Gaussian innovations (SC- L2 Innov.),

\(^3\) All computations are done on 8-core Intel Xeon, 2.4 GHz processor.
though performs better than the sparse coding model at times, is not able to track the state accurately, re-asserting our argument that considering sparse innovations make the model more stable and consistent. Finally, RWL1-DF is able to track the states as accurately as our model, but requires several observations before reaching a steady state and is computationally more expensive. In fact, we observed that RWL1-DF becomes very unstable when the observations have “inadequate” information, as a result of very noisy observation or when the number of observation dimensions are less. We discuss more about this in the following experiments.

Figure 3-5 shows the “steady” state error (rmSE) after 50 time instances versus with the dimensionality of the observation sequence (p). Each point is obtained after averaging over 50 runs. We observe that DSC is able to track the true states even for low dimensional observations, when other methods fail. This shows that the temporal relations adopted in the model provide contextual information necessary to track the
Figure 3-5. Performance of the dynamic sparse coding with varying number of observation dimensions. We use similar set of parameters as before, $\lambda = 10$, $\gamma = 10$ and $n = 3$.

Figure 3-6. Performance of the dynamic sparse coding with varying (sparse) noise levels ($n$). We use similar set of parameters as before, $\lambda = 10$, $\gamma = 10$ and $p = 70$.

changes in the observation, even when the information provided by the instantaneous observations is not sufficient. Notice also that RWL1-DF becomes very unstable when the dimensions of the observations are small.

Same can be extrapolated in case of noisy observation sequences, where the essential information in the time sequence is at scarce. Figure 3-6 shows the
performance of all the methods versus varying sparse noise levels \( (n) \). Again, we observe that DSC outperforms other methods, particularly when the noise levels are high. Also, notice that the RWL1-DF becomes very unstable when the noise levels are high.

### 3.3.2 Learning from Natural Image Sequences

![Figure 3-7. Receptive fields of the model at different levels. (A) Shows the receptive fields of the states, i.e., the columns of the observation matrix \( C^{(1)} \). (B) shows the receptive fields of the causes. The receptive here are constructed as a weighted combination of the columns of the layer 1 observation matrix \( C^{(1)} \).](image)

In the following experiments we show that the working of the states and causes resemble that of simple and complex cells in the visual cortex. The states act as simple feature detectors, while causes encode complex invariances. However, the key to our model is that the responses of both the states and the causes are influenced by the context, coming from both temporal and top-down connections, making them capable to represent observations that are beyond their characteristic receptive fields.

Firstly, we consider learning a model from natural video sequences obtained from the Van Hateren’s video database [van Hateren and Ruderman, 1998]. This database contains several video clips of natural scenes containing animals, tree, etc. and each frame of these video clips is preprocessed using local contrast normalization.
as described by Jarrett et al. [2009a]. Sequences of patches are then extracted from the preprocessed video sequences to learn the parameters of the model.

We use $17 \times 17$ pixel patches to learn 400 dimensional states and 100 dimensional causes. We consider the pooling between the states and the causes to be $2 \times 2$, i.e., we further divide each of the $17 \times 17$ patches into 4 overlapping $15 \times 15$ pixel patches and the states extracted from each of these subdivided patches are pooled to obtain the causes (refer to Figure 3-1).

Figure 3-7 shows the receptive fields of the states/causes at different levels of the learned model. The receptive fields of the states are simple inclined lines resembling Gabor function, while that of the causes resemble grouping of these primitive feature, localized in orientation and/or spatial position (more about this is Section 3.3.2.1).

### 3.3.2.1 Visualizing invariance

To get a better understanding of the invariance learned by the model, we visualize the connections between the first layer states and the causes. Figure 3-9 shows the results obtained—see caption for more details. We observe that most of the columns of the invariance matrix group together dictionary elements that have similar orientation and frequency, while being invariant to the other properties like translation. However,
Figure 3-9. Visualizing invariance represented by the causes. Firstly, each dictionary element (with a receptive field of $15 \times 15$ pixels) in the observation matrix ($C^{(1)}$) is fitted with a Gabor function and is parametrized in terms of the center position, spatial frequency and orientation of the Gabor functions, as shown in Figure 3-8. Then the connection strength between the invariance matrix ($B^{(1)}$) and the observation matrix ($C^{(1)}$) is plotted, i.e., the subset of the dictionary elements that are most likely to be active when a column of the invariance matrix is active are shown. Each box represents one column of the invariance matrix $B^{(1)}$ and 10 out of 100 columns are randomly selected. The figure shows the center and orientation scatter plots (A) and the corresponding the spatial frequency and orientation polar plots (B), highlighting the subset of most active dictionary elements for a select columns of $B^{(1)}$ (darker colors indicate stronger connections). Notice, for each active column in $B^{(1)}$, a subset of the dictionary elements (not unique) are grouped together according to orientation and/or spatial position, indicating invariance to the other properties like spatial frequency and center position.
Figure 3-10. Visualizing the temporal structure learned by the model. (A) Depicts the connection strength (of matrix $A^{(1)}$) between the layer 1 state elements over time. Read it as follows: if the basis on the left is active at time $t$ (presynaptic), then the corresponding set of basis on the right indicate the prediction for time $t + 1$ (postsynaptic). This indicates that certain properties, like orientation and spatial positions, change smoothly over time. (B) The scatter plot of 15 strongest connections per each element in the matrix $A^{(1)}$, arranged according to the orientation selectivity of the pre and post synaptic elements. Notice that most points are within $\pi/6$ from the diagonal, indicated by the black lines.

there are other types of the invariances as well, where the dictionary elements are grouped only by spatial location while being invariant to other properties\[^4\].

3.3.2.2 Learning temporal structure

As shown above, the receptive fields of the bottom layer states in our model resemble that of simple cells in V1 area of visual cortex. It is now well known that these cells act as simple oriented filters and strongly respond to particular inputs [Olshausen

\[^4\] Appendix B.1 shows visualization of all the causes
and Field, 1996]. However, recent studies show that their influence extends beyond their receptive fields [Rust et al., 2005], modulating the response of other cells, in both excitatory and inhibitory ways, depending on the spatial and temporal contextual information. In our approach, such temporal context at each layer is modeled using the parameter matrix \( A^{(l)} \). Figure 3-10 shows a visualization of this matrix at the bottom layer. We observe that the model learns to maintain certain properties, like orientation and spatial position, over time. In other words, given a basis is active at a particular time it has excitatory connections with a group of basis (sometimes with strong self-recurrence connection) making them more likely to be active at the next time instance. On the other hand, along with the sparsity regularization, it also inhibits response of other elements that are not strongly connected with the active basis over time.

### 3.3.3 Role of Temporal Connections: Toy data

In the previous experiment we have shown that the states at any time \( t \) are closely related to the states at time \( (t - 1) \). But we left out an important question, how does the previous state of the system affect our perception (or encoding) of the present observations? In other words, does the context helps us to disambiguate an aliased observation, i.e., can we differentiate a similar pattern occurring in two different sequences? We try to answer this here by considering a toy data example. This problem is previously studied in predictive coding framework as well [Rao, 1999].

In this experiment, an observation sequence is made up of patches, with each patch containing parallel lines (number of lines chosen uniformly between 1 to 5) with the same orientations (also chosen uniformly from four different orientations), such that from one frame to the other the patch is shifted by only one pixel. Figure 3-11A shows a set of sequences, each row representing one sequence, generated using the above procedure. Again, such sequences are concatenated to form a longer sequence of observations.
Figure 3-11. Role of temporal connections during inference. (A) Toy sequences used in the experiment; each row indicating one sequence. (B) The basis (or matrix C) learned from these inputs. (C) The inferred state vector at time $t = 3$ when the same sequence is presented in two different orders. We observe that depending on the ordering the same pattern has different representation.
Since we are interested here in only the temporal connections, we do not consider any causes (i.e., we fix $u_t = 0$) during this experiment. After learning the system on the observation sequences, we fix the parameters and present to the system two sequences: a sequence of a particular shape and the same sequence in reverse order, as shown in Figure 3-11C. Note that at time $t = 3$ the input is the same but in a different context; in the first case the pattern moving from top to bottom while in the second case it is moving in the opposite direction. We observe that the inferred states at time $t = 3$ are different, i.e., the context in which a particular pattern is observed changes its representation. Note that having observations alone at each time can do the same. In addition to this, the system is also capable to learn a representation such that it can still “generate” the observations back using the basis, even though with different representations.
CHAPTER 4
DEEP PREDICTIVE CODING NETWORKS

In this chapter we discuss a hierarchical dynamic model called deep predictive coding networks. The feature extraction block discussed in Chapter 3 is used as the basic building block to construct this “deep” network. In line with other deep learning methods, we use these basic building blocks to construct a hierarchical model using greedy layer-wise unsupervised learning. The hierarchical model is built such that the output from one layer acts as an input to the layer above. In other words, the layers are arranged in a Markov chain such that the states at any layer are only dependent on the representations in the layer below and above, and are independent of the rest of the model. The overall goal of the dynamical system at any layer is to make the best prediction of the representation in the layer below using the top-down information from the layers above and the temporal information from the previous states. Hence, the name deep predictive coding networks (DPCN).

4.1 Multi-Layered Architecture

The architecture of the multilayered processing model is a tree structure, with the simple encoding module described in Chapter 3 replicated at each node of the tree (Figure 4-1). At the bottom layer, the nodes are arranged as a tiling of the entire visual scene and the parameters across all the nodes are tied, resembling a convolution over the input frame. Each node encodes a small patch of the input video sequence, which is useful for parallelizing the computation. The computational model is uniform within a layer, and across layers, albeit with different dimensions, the only thing that changes is the nature of the input data. Note that within each block the features extracted from a spatial neighborhood are pooled together, indicating a progressively increasing receptive field size of the nodes with the depth of the network. For this reason, we can also expect that the activation duration of a given feature also slows down with the depth of the module. Parameter learning at each layer uses a greedy layer-wise procedure, i.e. the
Figure 4-1. A two-layered hierarchical model constructed by stacking several state-space models. For visualization no overlapping is shown between the image patches here, but overlapping patches are considered during actual implementation.

parameters at the bottom layer modules are learned first from a sequence of small patches extracted from the input video sequences, and only after the learning of the next layer starts. Figure 4-2 exemplifies the inference on a two-layer network with a single module in each layer for simplicity. Here the layers in the hierarchy are arranged in a Markov chain, i.e., the variables at any layer are only influenced by the variables in the layer below and the layer above. Specifically, at the bottom layer for example, sequences of patches ($y_t$) extracted from fixed spatial locations spread across the entire 2D space of the video frames is fed as input to each first layer modules. On the other hand, the top-down predictions of the first layer causes coming from the second layer try to modulate the representations. The bidirectional nature of the model is apparent in this figure, and in general there may be an extra top down prediction as input to provide context for the analysis. Next we will include the modifications in the general equation to exploit this extra information.
Figure 4-2. Block diagram showing the flow of bottom-up and top-down information in the model.

4.2 Inference in Multi-Layered Network with Top-Down Connections

With the parameters fixed, inferring latent variables at any intermediate layer involves obtaining useful representation of the data driven bottom-up information while combining the top-down influences from the higher layers. In other words, while the dynamic network at each layer try to extract useful information from the inputs for recognition, the top-down connections modulate the representations at each level with abstract knowledge from the higher layers. As we will show next, the top-down connections “convey” contextual information to endow the model with a prior knowledge for extracting task specific information from noisy inputs.

More formally, at any layer \((l)\), the energy function that needs to be minimized to infer \(x_t^{(l)}\) and \(u_t^{(l)}\) is given by:

\[
E_l(x_t^{(l)}, u_t^{(l)}, \theta^{(l)}) = \sum_{n=1}^{N} \left( \frac{1}{2} ||u_t^{(l-1,n)} - C^{(l)} x_t^{(l,n)}||^2_2 + \lambda ||x_t^{(l,n)} - A^{(l)} x_t^{(l,n)}||^2_1 \right)
+ \sum_{k=1}^{K} \left| y_t^{(l,k)} - x_t^{(l,k)} \right| + \beta ||u_t^{(l)}||_1 + \frac{1}{2} ||u_t^{(l)} - \hat{u}_t^{(l+1)}||^2_2 \quad (4-1)
\]

\[
\gamma_t^{(l)} = \gamma_0 \left[ 1 + \exp \left( - |B^{(l)} u_t^{(l)}|^2 \right) \right]
\]
where $\hat{\mathbf{u}}^{(l)}_t = C^{(l+1)} \hat{x}^{(l+1)}_t$ is the top-down prediction of the causes coming from the state-space model in the layer above. This additional term involving $\hat{\mathbf{u}}_t$ influences the representation at the $(l)^{th}$ layer by reducing the top-down prediction error. In other words, the goal is to match the representation of the inputs from the layer below with the belief of the layer above about the same representation.

Ideally, to perform inference in this hierarchical model, all the states and the causes have to be updated simultaneously depending on the present state of all the other layers until the model reaches equilibrium [Friston, 2008]. However, such a procedure can be very slow in practice. Instead, we propose an approximate inference procedure that only requires a single top-down flow of information and then a single bottom-up inference using this top-down information. Specifically, before the “arrival” of a new observation at time $t$, at each layer $(l)$ (starting from the top-layer) we first propagate the most likely causes to the layer below using the state at the previous time instance $\hat{x}^{(l)}_{t-1}$ and the predicted causes $\hat{\mathbf{u}}^{(l+1)}_t$. More formally, the top-down prediction at layer $l$ is obtained as:

$$\hat{\mathbf{u}}^{(l)}_t = C^{(l)} \hat{x}^{(l)}_t$$

where $\hat{x}^{(l)}_t = \arg\min_{\hat{x}^{(l)}_t} \lambda^{(l)} ||\hat{x}^{(l)}_t - A^{(l)} \hat{x}^{(l)}_{t-1}||_1 + \gamma_0 \sum_{k=1}^{K} |\tilde{\gamma}_{t,k} \cdot \hat{x}^{(l)}_{t,k}|$ \hspace{1cm} (4–2)

and \(\tilde{\gamma}_{t,k} = (\exp(-[B^{(l)} \hat{\mathbf{u}}^{(l+1)}_t]_k))/2\)

At the top most layer, $L$, a “bias” is set such that $\hat{\mathbf{u}}^{(L)}_t = \hat{\mathbf{u}}^{(L)}_{t-1}$, i.e., the top-layer induces some temporal coherence on the final outputs. From (4–2), it is easy to show that the predicted states for layer $l$ can be obtained as

$$\hat{x}^{(l)}_{t,k} = \begin{cases} [A^{(l)} \hat{x}^{(l)}_{t-1}]_k, & \gamma_0 \gamma_{t,k} < \lambda^{(l)} \\ 0, & \gamma_0 \gamma_{t,k} \geq \lambda^{(l)} \end{cases}$$

(4–3)
These predicted causes $\hat{u}_t^{(l)}$, $\forall l \in \{1, 2, ..., L\}$ are substituted in (4–1) and a single layer-wise bottom-up inference is performed as described in section 3.2.1 \(^1\). The combined prior now imposed on the causes, $\beta\|u_t^{(l)}\|_1 + \frac{1}{2}\|u_t^{(l)} - \hat{u}_t^{(l+1)}\|_2^2$, is similar to the elastic net prior discussed in [Zou and Hastie, 2005], leading to a smoother and biased estimate of the causes.

4.3 Experiments

4.3.1 Receptive Fields in the Hierarchical Model

Firstly, we would like to test the ability of the proposed model to learn complex features in the higher-layers of the model. For this we train a two layered network from a natural video. Each frame in the video was first contrast normalized as described in [Kavukcuoglu et al., 2010b]. Then, we train the first layer of the model on 4 overlapping contiguous $15 \times 15$ pixel patches from this video; this layer has 400 dimensional states and 100 dimensional causes. The causes pool the states related to all the 4 patches. The separation between the overlapping patches here was 2 pixels, implying that the receptive field of the causes in the first layer is $17 \times 17$ pixels. Similarly, the second layer is trained on 4 causes from the first layer obtained from 4 overlapping $17 \times 17$ pixel patches from the video. The separation between the patches here is 3 pixels, implying that the receptive field of the causes in the second layer is $20 \times 20$ pixels. The second layer contains 200 dimensional states and 50 dimensional causes that pools the states related to all the 4 patches.

Figure 4-3 shows the visualization of the receptive fields of the invariant units (columns of matrix $B$) at each layer. We observe that each dimension of causes in the first layer represents a group of primitive features (like inclined lines) which are

\(^1\) Note that the additional term $\frac{1}{2}\|u_t^{(l)} - \hat{u}_t^{(l+1)}\|_2^2$ in the energy function only leads to a minor modification in the inference procedure, namely this has to be added to $h(u_t)$ in (3–10).
Figure 4-3. Visualization of the receptive fields of the invariant units learned in (a) layer 1 and (b) layer 2 when trained on natural videos. The receptive fields are constructed as a weighted combination of the dictionary of filters at the bottom layer.

localized in orientation or position \(^2\). Whereas, the causes in the second layer represent more complex features, like corners, angles, etc. These filters are consistent with the previously proposed methods like Lee et al. [2009] and Zeiler et al. [2010].

4.3.2 Role of Top-Down Information

In this section, we show the role of the top-down information during inference, particularly in the presence of structured noise. Video sequences consisting of objects of three different shapes (Refer to Figure 4-4A) were constructed. The objective is to classify each frame as coming from one of the three different classes. For this, several 32 × 32 pixel 100 frame long sequences were made using two objects of the same shape bouncing off each other and the “walls”. Several such sequences were then concatenated to form a 30,000 long sequence. We train a two layer network using this sequence. First, we divided each frame into 12 × 12 patches with neighboring patches overlapping by 4 pixels; each frame is divided into 16 patches. The bottom layer was trained such the 12 × 12 patches were used as inputs and were encoded using a 100

\(^2\) Please refer to supplementary material for more results.
Figure 4-4. Part of the (A) clean and (B) corrupted video sequences constructed using three different shapes. Each row indicates one sequence.

dimensional state vector. A 4 contiguous neighboring patches were pooled to infer the causes that have 40 dimensions. The second layer was trained with 4 first layer causes as inputs, which were itself inferred from $20 \times 20$ contiguous overlapping blocks of the video frames. The states here are 60 dimensional long and the causes have only 3 dimensions. It is important to note here that the receptive field of the second layer causes encompasses the entire frame. We test the performance of the DPCN in two conditions. The first case is with 300 frames of clean video, with 100 frames per shape, constructed as described above. We consider this as a single video without considering any discontinuities. In the second case, we corrupt the clean video with “structured” noise, where we randomly pick a number of objects from same three shapes with a Poisson distribution (with mean 1.5) and add them to each frame independently at a random locations. There is no correlation between any two consecutive frames regarding where the “noisy objects” are added (see Figure 4-4B).

First we consider the clean video and perform inference with only bottom-up inference, i.e., during inference we consider $\hat{u}_t^{(l)} = 0$, $\forall l \in \{1, 2\}$. Figure 4-5A shows the scatter plot of the three dimensional causes at the top layer. Clearly, there are 3 clusters recognizing three different shape in the video sequence. Figure 4-5B shows the
Figure 4-5. The scatter plot of the 3 dimensional causes at the top-layer for (A) clean video with only bottom-up inference, (B) corrupted video with only bottom-up inference and (C) corrupted video with top-down flow along with bottom-up inference. At each point, the shape of the marker indicates the true shape of the object in the frame.

scatter plot when the same procedure is applied on the noisy video. We observe that 3 shapes here cannot be clearly distinguished. Finally, we use top-down information along with the bottom-up inference as described in section 4.2 on the noisy data. We argue that, since the second layer learned class specific information, the top-down information can help the bottom layer units to disambiguate the noisy objects from the true objects. Figure 4-5C shows the scatter plot for this case. Clearly, with the top-down
information, in spite of largely corrupted sequence, the DPCN is able to separate the frames belonging to the three shapes (the trace from one cluster to the other is because of the temporal coherence imposed on the causes at the top layer).

In this work we proposed the deep predictive coding network, a generative model that empirically alters the priors in a dynamic and context sensitive manner. This model composes to two main components: (a) linear dynamical models with sparse states used for feature extraction, and (b) top-down information to adapt the empirical priors. The dynamic model captures the temporal dependencies and reduces the instability usually associated with sparse coding, while the task specific information from the top layers helps to resolve ambiguities in the lower-layer improving data representation in the presence of noise. Our approach can be extended with convolutional methods, paving the way for implementation of high-level tasks like object recognition, etc., on large scale videos or images. This will be discussed in the next chapter.
In this chapter, we consider a specific architecture based on spatial convolution for the state-space model discussed in the previous chapters, designed to extract information that is invariant to transformations of the objects in the input scene. We show that this convolutional dynamic network (CDN) can combine the bottom-up, top-down and lateral (or temporal) influences in the network effectively. More importantly, it can scale to large images/frames and learn decomposition of object parts in a hierarchical fashion. We show the performance of the model on several benchmark object and face recognition tasks and show that the proposed model is better/comparable to other methods in the literature. We also discuss, using several experiments, the influence of the top-down and temporal connections have on the performance of the proposed model.

5.1 Model Architecture

5.1.1 Single Layer Model

We first consider a single layer model shown in Figure 5-1 to process a video sequence. Here the inputs/observations to the model is a sequence of video frames $I_t, \forall t \in \{1, 2... T\}$ and each frame is composed of $M$ color channels, denoted as $\{I^1_t, I^2_t...I^M_t\}$. Now, we assume that each channel $I^m_t$ is modeled as an observation to a state-space model, with the same set of states used across all the channels. Specifically, each channel $I^m_t$ is modeled as a linear combination of $K$ matrices, $X^k_t \forall k \in \{1, 2...K\}$, convolved with filters $C_{m,k} \forall k$. The state space equations for this model can be written as:

$$I^m_t = \sum_{k=1}^{K} C_{m,k} \ast X^k_t + N^m_t \quad \forall m \in \{1, 2...M\}$$

$$X^k_t(i,j) = \sum_{k=1}^{K} a_{k,k} X^k_{t-1}(i,j) + V^k_t(i,j) \quad (5-1)$$
Figure 5-1. Block diagram of a single layer convolutional dynamic network. Inputs here contain 3 channels (denoted in RGB colors) and each channel is modeled as a combination of the state maps (black) convolved with filters $C$ (blue). The pooled state maps (orange) are decomposed using the cause maps (purple) convolved with filters $B$ (blue). During inference there is a two-way interaction between the state and the cause mappings through pooling/unpooling operations, which is left implicit here.

where $*$ denotes convolution. If $I^k_{t}$ is a $w \times h$ frame and $C_{m,k}$ is a $s \times s$ pixel filter, then $X^k_{t}$ is a matrix of size $(w + s - 1) \times (h + s - 1)$. We refer to $X_t = \{X^k_{t}\} \forall k$ as state maps (or sometimes simply as states). Also, $a_{k,\tilde{k}}$ indicates the lateral connections between the state maps over time. Since we are only interested in object recognition in this work, we assume

$$a_{k,\tilde{k}} = \begin{cases} 
1, & k = \tilde{k} \\
0, & otherwise 
\end{cases}$$

i.e., we consider only self-recurrent connections between state maps, which encourages temporal coherence. However, one can model the motion in the input sequences by
alternatively learning the coefficients $a_{k, \tilde{k}}$ along with the rest of the model parameters [Cadieu and Olshausen, 2008].

Since (5–1) is a under-determined model, we regularize it with a sparsity constraint on the states to obtain a unique solution. Hence, the combined energy function for the state-space model in (5–1) can be written as follows:

$$E_x(X_t, C) = \sum_{m=1}^{M} \|I^m_t - \sum_{k=1}^{K} C_{m,k} \cdot X^k_t\|_2^2 + \lambda \|X_t - X_{t-1}\|_1 + \sum_{k=1}^{K} \gamma_k^k \|X^k_t\|_1$$

(5–2)

Notice that we consider that the state transition noise $V_t$ in (5–1) to also be sparse, so that it is consistent with the sparsity on the states. This makes practical sense, as the number of changes between two consecutive frames in a typical video sequence is small.

In (5–2), $\gamma_k$ is a sparsity parameter on the $k$th state map. Instead of assuming that the sparsity of the states is constant (or that the prior distribution over the states is stationary) as in [Zeiler et al., 2010], here we consider that the cause maps (or causes) $U_t$ modulate the activity of the states through the sparsity parameter. In line with the model proposed by Karklin and Lewicki [2005], we consider the sparsity parameter $\gamma \in \mathbb{R}^{(w+s-1) \times (h+s-1) \times K}$ in terms of the causes $U_t \in \mathbb{R}^{(w+s-p) \times (h+s-p) \times D}$ as:

$$\gamma_k = \frac{\gamma_0}{2} \left(1 + \exp \left\{-\sum_{d=1}^{D} B_{k,d} \cdot U^d_t \right\} \right)$$

(5–3)

where $\gamma_0 > 0$ is a constant. This non-linear multiplicative interaction between the state and the cause mappings leads to extracting information that is invariant to several transformations from the inputs. Essentially, through the filters $B_{k,d} \in \mathbb{R}^{p \times p}$, $U^d_t$ learn to group together the states that co-occur frequently. Since co-occurring components typically share some common statistical regularity, such activity typically leads to locally invariant representation [Karklin and Lewicki, 2005]. More importantly, unlike many other
deep learning methods [Hinton et al., 2006, Zeiler et al., 2010], the activity of the causes influences the states directly through the top-down connections \((B_{k,d})\) and the statistical grouping is learned from the data, instead of a pre-determined topographic connections [Hyvärinen and Hoyer, 2001].

Given fixed state maps, the energy function that needs to be minimized to obtain the causes is:

\[
E_u(U_t, B) = \sum_{k=1}^{K} \frac{\gamma_0}{2} \left( 1 + \exp \left\{ - \sum_{d=1}^{D} B_{k,d} \cdot U_{t}^d \right\} \cdot |X_{t}^{k}| \right) + \beta \|U_t\|_1
\]

(5–4)

where we regularized the solution using an \(\ell_1\) sparsity penalty. We note here that all the elements of \(B\) are initialized to be non-negative and they remain so without any additional constraint. This ensures that the gradient of the smooth part (or the first term) of \(E_u(\cdot)\) is Lipschitz continuous, allowing us to use proximal methods to infer \(U_t\) with guaranteed convergence [Beck and Teboulle, 2009], as discussed in Section 5.2.

5.1.2 Building a Hierarchy

Several of these single-layer models described above can easily be stacked to form a hierarchical model. Like many other deep architectures, such as deep belief networks [Hinton et al., 2006], stacked auto-encoders [Vincent et al., 2010], etc., the outputs (or cause maps) from one layer act as input to the layer above. However, unlike these models, each layer gets, along with the bottom-up inputs, a top-down predictions of its output causes. Then the goal during inference of the states and the causes at any layer is to come up with representations that best predicts the inputs while reducing the top-down prediction error. More formally, combining the top-down predictions into the single layer model described in Section 5.1.1, the energy function at the \(l^{th}\) layer in the
Hierarchical model can be written as:

$$E_l(X^l_t, U^l_t, C^l, A^l, B^l) = \sum_{m=1}^{D_{t-1}} \|U^m_{t,l}-1 - \sum_{k=1}^{K_l} C_{m,k}^l \cdot X^k_{t,l}\|^2_2 + \lambda' \|X^l_t - X^l_{t-1}\|_1 + \sum_{k=1}^{K_l} \gamma^k \|X^k_{t,l}\|_1$$

$$+ \beta' \|U^l_{t,1}\|_1 + \eta' \|U^l_t - \hat{U}^l_t\|^2_2$$

(5–5)

where $U^{l-1}_t$ are the causes coming from the layer below and $\hat{U}^l_t$ is the top-down prediction coming from the state-space model in the layer above. As indicated by the energy function in (5–5), the architecture at each layer is similar to the single layer model described before, though the number of states ($K^l$) and causes ($D^l$) might vary over the layers.

### 5.1.3 Implementation Details

To make the implementation more efficient, we introduce some restrictions on the architecture. Firstly, like other convolutional models [LeCun et al., 1989, Lee et al., 2009, Kavukcuoglu et al., 2010a, Zeiler et al., 2010], we assume sparse connectivity between both the inputs and states and also, between states and causes. This not only increases the efficiency during inference but also breaks the symmetry between layers and helps to learn complex relationships. Secondly, we shrink the size of the states using max pooling between the states and the causes. Correspondingly, the sparsity parameters ($\gamma$) obtained from the causes are unpoled during inference of the states (see Section 5.2 for details). This reduces the size of the inputs going into the higher layers and hence, is more efficient during inference. Also, the pooling is shown to produce better invariant representations [Boureau et al., 2010].

### 5.2 Inference

At any layer $l$, inference involves finding the states $X^l_t$ and the causes $U^l_t$ that minimizes the energy function $E_l$ in (5–5). To perform this joint inference, we alternately update the states with the causes fixed and then update the causes with the states fixed.
Algorithm 2 Inference in Convolutional Dynamic Network

Require: Inputs - $I_{1:T}, N$ - # FISTA iterations, $L$ - # layers, Parameters - $C^{1:L}, B^{1:L}$

Require: Hyper-parameters - $\lambda^{1:L}, \beta^{1:L}, \eta^{1:L}, \gamma^{1:L}$

Require: Initialize states - $X^{1:L}_0 = 0$

1: for $t = 1 : T$ do // Loop over time
   2:    // Top-down predictions.
   3:       for $l = L : -1 : 1$ do // Loop over layers
   4:          Compute $\hat{X}^{l}_t$ using (5–18)
   5:          Predict: $\hat{U}^{l}_t$ using (5–17)
   6:      end for
   7:    // Bottom-up inference.
   8:    Initialize: $X^{l}_t = X^{l}_{t-1}, U^{l}_t = \hat{U}^{l}_t$
   9:    for $l = 1 : L$ do // Loop over layers
  10:       for $n = 1 : N$ do // FISTA iteration
  11:          Compute state prediction term: $\alpha^*$.
  12:          Update states $X^{l}_t$ using (5–12) and (5–13).
  13:          Max-pooling: $[\text{down}(X^{k,l}_t), p^{k,l}_t] = \text{pool}(X^{k,l}_t)$.
  14:          Update causes $U^{l}_t$ using (5–15).
  15:          Unpool and re-compute $\gamma'$ using (5–16).
  16:     end for
  17:    end for
  18: end for

until convergence. Updating either of them involves solving an $\ell_1$ convolutional sparse coding problem and we use a proximal gradient based method called FISTA [Beck and Teboulle, 2009, Chalasani et al., 2013] (and some variations [Chalasani and Principe, 2013]) for this, where each update step involves computing the gradient, followed by a soft thresholding function to obtain a sparse solution.

5.2.1 Procedure

Algorithm 2 shows the steps involved in this iterative inference procedure and below we will elucidate each of the steps in detail.
Updating States

Firstly, with the causes fixed, updating the states involve finding the gradient of all the terms other than the sparsity penalty in $E$ w.r.t $X$. For convenience, we re-write these terms as:

$$h(X) = \sum_{m=1}^{D} \| U_{m,t}^{l} - \sum_{k=1}^{K} C_{m,k}^{l} \|_{2}^{2} + \lambda \| X_{t}^{l} - X_{t-1}^{l} \|_{1}$$ (5–6)

Since $h(X)$ is non-smooth, the second term involving state transitions has $\ell_{1}$ penalty, it is not possible to find its exact gradient. However, in order to approximately compute it, we first use the idea of Nesterov's smoothness [Nesterov, 2005] to approximate the non-smooth state transition term in $h(X)$ with a smooth function.

To begin, let's consider $\Omega(X) = \| e_{t} \|_{1}$ where $e_{t} = \| \text{vec}(X_{t}) - \text{vec}(X_{t-1}) \|_{1}$. The idea is to approximate $\Omega(X)$ with a smooth function and compute its gradient with respect to $e_{t}$. Since, $e_{t}$ is a linear function of $X$, computing the gradient of $\Omega(X)$ w.r.t $X_{t}$ then becomes straightforward.

Now, using the dual of $\ell_{1}$-norm, we can re-write $\Omega(X)$ as

$$\Omega(X) = \arg \max_{\| \alpha \|_{\infty} \leq 1} \alpha^{T} e_{t}$$ (5–7)

where $\alpha \in \mathbb{R}^{\text{card}(e_{t})}$. Using Nesterov's smoothness property, we can approximate $\Omega(X)$ with a smooth function of the form:

$$\Omega(X) \approx f_{\alpha}(e_{t}) = \arg \max_{\| \alpha \|_{\infty} \leq 1} \alpha^{T} e_{t} - \mu d(\alpha)$$ (5–8)

where $d(\alpha) = \frac{\| \alpha \|_{2}^{2}}{2}$ is a smoothness function and $\mu$ is the smoothness parameter.

Following Theorem 1 in Chen et al. [2012a], we can show that $f_{\alpha}(e_{t})$ is convex and smooth and, moreover, the gradient of $f_{\alpha}(e_{t})$ w.r.t $e_{t}$ is given by:

$$\nabla_{e_{t}} f_{\alpha}(e_{t}) = \alpha^{*}$$ (5–9)
where \( \alpha^* \) is the optimal solution to (5–8). We can obtain a closed-form solution to \( \alpha^* \) as (for proof refer to Chalasani and Principe [2013]):

\[
\alpha^* = S\left( \frac{e_t}{\mu} \right) \tag{5–10}
\]

where \( S(.) \) is a projection operator applied over every element in \( \alpha^* \) and is defined as follows:

\[
S(x) = \begin{cases} 
  x, & -1 \leq x \leq 1 \\
  1, & x > 1 \\
  -1, & x < -1 
\end{cases}
\]

As discussed above, using chain rule, \( f_\mu(e_t) \) is also convex and smooth in \( X^l_t \) and its gradient \( \nabla_{X^l_t} f_\mu(e_t) \) remains the same as in (5–9).

Given this smooth approximation of the non-smooth state transition term and its gradient, we now apply the iterative shrinkage-thresholding algorithm [Beck and Teboulle, 2009] for convolutional states-space model with a sparsity constraint. The gradient of re-formulated \( h(X^l_t) \) w.r.t \( X^l_t \) is given as follows:

\[
\nabla_{X^l_t} h(X^l_t) = -\sum_{m=1}^{D_l-1} \tilde{C}_{k,m} * \left( U_{m,1}^l - \sum_{k=1}^{K_l} C_{k,m} * X^k_{t,1} \right) + \lambda \hat{M}^k_{\alpha^*} \tag{5–11}
\]

where \( \tilde{C}_{k,m} \) indicates that the matrix \( C_{k,m} \) is flipped vertically and horizontally and \( \hat{M}^k_{\alpha^*} \) is the \( \hat{k}^{th} \) map from a matrix obtained by reshaping \( \alpha^* \).

Once we obtain the gradient, the states can be updated as:

\[
X^l_t = X^l_t - \gamma \nabla_{X^l_t} h(X^l_t) \tag{5–12}
\]
where $\tau$ is a step size for the gradient descent update$^1$. Following this, we pass the updated states through a soft thresholding function that clamps the smaller values, leading to a sparse solution:

$$X'_t = \text{sign}(X'_t)(\max |X'_t| - \gamma')$$  \hspace{1cm} (5–13)

Max Pooling

We then perform a spatial max pooling [Jarrett et al., 2009a] over small neighborhoods across each and every 2D state map as:

$$[\text{down}(X'^{k,l}_t), p'^{k,l}_t] = \text{pool}(X'^{k,l}_t)$$

where $p'^{k,l}_t$ indicates the pooling indexes. We do not pooling across the state maps, so the number of state maps remains the same, while the resolution of each map decreases (denoted as $\text{down}(X'^{k,l}_t)$). We use non-overlapping spatial windows for the pooling operation.

Update Causes

Similar to the state updates described above, we fix the states and compute the gradient using only the smooth part of the energy function $E_t$ (denoted as $h(U'^{k,l}_t)$) w.r.t $U'_t$. Given the pooled states, the gradient can be computed as follows:

$$\nabla_{U'^{i,j}_t} h(U'^{i}_t) = -\frac{\gamma_0}{2} \sum_{k=1}^{K_t} \tilde{B}_{k,d} * \left[ \exp \left\{ - \sum_{d=1}^{D_t} B'^{l,k,d}_t \cdot U'^{d,i}_t \right\} \right] . \left[ \text{down}(X'^{k,i}_t) \right] + 2\eta(U'^{d,i}_t - \tilde{U}'^{d,i}_t)$$  \hspace{1cm} (5–14)

$^1$ FISTA Beck and Teboulle [2009] uses a momentum term during the gradient update and leads to much faster convergence. We use this in our implementation. Please refer to Chalasani et al. [2013] for convolutional sparse coding with FISTA.
Similar to the state updates described above, using this gradient information, we update the causes by first taking a gradient step, followed by a soft thresholding function:

\[
U'_t = U'_t - \beta' \tau \nabla U'_t h(U'_t)
\]

\[
U'_t = \text{sign}(U'_t)(\max |U'_t| - \beta')
\] (5–15)

**Unpooling**

Now, after updating the causes, we re-evaluate the sparsity parameter \(\gamma\) for the next iteration. We do this as follows:

\[
\gamma^{k,l} = \frac{\gamma_0}{2} \left( 1 + \exp \left\{ -\text{unpool}_{p_t}^k \left( \sum_{d=1}^{D_k} B_{d,k,a} \ast U_{d,l}^k \right) \right\} \right)
\] (5–16)

where \(\text{unpool}_{p_t}^k(\cdot)\) indicates reversing the pooling operation using the indexes \(p_{k,l}\) obtained during the max pooling operation described above [LeCun et al., 1998]. Notice that, while the inputs to the pooling operation are the inferred states, the inputs to the unpooling operations are the likely states “generated” by the causes.

**Overall Iteration**

A single iteration consists of the above mentions four steps: update the states using a single FISTA step, perform max pooling over the states, update the causes using a single FISTA step and, finally, re-evaluate the sparsity parameter for the next iteration.

All the computations during inference involves only basic operations such as convolution, summation, pooling and unpooling. All of these can be efficiently implemented on a GPU with parallelization, making the overall process very quick.

**5.2.2 Approximate Inference with Top-Down Connections**

In the inference procedure described above, while updating the causes, we assumed that the top-down predictions \(\hat{U}_t\) are already available and are constant through out the inference procedure. However, ideally, this shouldn’t be the case. Since the layers are arranged in a Markov chain, all the layers have to be concurrently updated, while passing top-down and bottom-up information, until the system reaches
an equilibrium. In practice, this can be very slow to converge. In order to avoid this, we do an approximate inference, where we make a single approximate top-down predictions at each time step using the states from the previous time instance and perform a single bottom-up inference with fixed top-down predictions, starting from the bottom layer.

More formally, at every time step, using the state-space model at each layer we predict the most likely cause at the layer below ($\hat{U}_t^{l-1}$), given only the previous states and the predicted causes from the layer above. Mathematically, the top-down prediction at layer $l$ can be written as:

$$\hat{U}_t^{m,l-1} = \sum_{k=1}^{K_l} C_{m,k}^l \cdot \hat{X}_t^{k,l} \quad \forall m \in \{1, 2..., D_{l-1}\}$$

where

$$\hat{X}_t^l = \arg \min_{X_t^l} \lambda^l \|X_t^l - X_t^{l-1}\|_1 + \hat{\gamma} \cdot \|X_t^l\|_1$$

$$\hat{\gamma}^k = \frac{\gamma_0}{2} \left(1 + \exp \left\{ - \text{unpool}_{l+1} \left( \sum_{d=1}^{D_{l+1}} B_{k,d}^l \cdot \hat{U}_t^{d,l} \right) \right\} \right)$$

and $\hat{U}_t^l$ itself is a top-down prediction coming from layer $l + 1$. At the top-layer, we consider the output from the previous time as the predicted causes, i.e., $\hat{U}_t^L = U_{t-1}^L$, allowing temporal smoothness over the outputs of the model.
Simple analytic solution can be obtained for $\hat{X}_t^i$ in (5–17) and is given by:

$$
\hat{X}_t^{k,l}(i,j) = \begin{cases} 
X_{t-1}^{k,l}(i,j) & \hat{r}^{k}(i,j) < \lambda^l \\
0 & \hat{r}^{k}(i,j) \geq \lambda^l
\end{cases}
$$  \hspace{1cm} (5–18)

Figure 5-2 shows the block diagram of a two-layered network, indicating the flow of information during inference. Before going further, there are several important things about this inference procedure that are worth noting. Firstly, the prior (or the regularization term) on the causes in the hierarchical model (5–5) involves two terms: an $\ell_1$ regularization encouraging sparsity and $\ell_2$ term with a bias coming from the top-down predictions. This resembles the elastic net regularization [Zou and Hastie, 2005], albeit with a bias. Secondly, observing the role of top-down predictions during inference through (5–14) and (5–18), one can see that they play a dual role of driving and as well as modulatory signals. While in the former the predictions $\hat{U}_t^l$ drives the representations through the gradient and bias them to some top-down expectations, in (5–18) they “shut” some of the state elements while performing top-down predictions and hence, act as a modulatory signal.

Also, at any layer the mapping between the inputs and the output causes is highly non-linear. This non-linearity comes from several factors: (i) the thresholding function while updating the states and the causes, (ii) the pooling operation and (iii) the causes and the states interact through an exponential function as shown in (5–14).

## 5.3 Learning

During learning the goal is to estimate the filters across all the layers in the model, such that they capture the structure across the entire observation sequence $\{I_1, I_2, \ldots, I_T\}$. We do this in a greedy layer-wise fashion, where we estimate the parameters of one layer at a time, starting from the bottom layer. At any layer $l$, the objective is to minimize the cost function $E_l(.)$ in (5–5) by alternating between inferring the representations (both the states and the causes) and updating the parameters. We note that during
learning, we do not consider any top-down connections by setting \( \eta^l = 0 \) \( \forall l \) in (5–5) while inferring the representations.

At layer \( l \), after inferring \( X^l \) and \( U^l \) and fixing them, we update the filters \( C^l \) and \( B^l \) using gradient descent (with momentum) minimizing the cost function \( E_i(\cdot) \). The gradient of \( E_i(\cdot) \) with respect to \( C^l \) can be computed as:

\[
\nabla_{c_{m,k}^{l}} E_i = -2 \tilde{X}_{t}^{k,l} \ast \left( l_{m}^{i} - \sum_{k=1}^{K_l} C_{k,m} \ast X_{t}^{k,l} \right) \tag{5–19}
\]

and the gradient of \( E_i(\cdot) \) w.r.t \( B^l \) can be computed as:

\[
\nabla_{b_{k,d}^{l}} E_i = - \tilde{U}_{t}^{d,l} \ast \left[ \left( \exp \left\{ - \sum_{d=1}^{D_l} B_{k,d}^{l} \ast U_{t}^{d,l} \right\} \right) \cdot \down(X_{t}^{k,l}) \right] \tag{5–20}
\]

After updating the filters, we normalize each filter to be of unit norm to avoid a trivial solution.

### 5.4 Experiments

In this section, we evaluate the performance of the proposed model on various tasks: (i) its ability to learn hierarchical representations and objects parts from unlabeled video sequences, (ii) object recognition with contextual information, and (iii) sequential labeling of video frames for recognition and (iv) its robustness in noisy environment.

**Preprocessing:** In all the experiments we perform the same pre-processing on the inputs. Each frame in a video sequence (or each image) is converted into gray-scale. We then normalize each frame to be zero mean and unit norm, followed by local contrast normalization as described by Jarrett et al. [2009a].

**Classification:** The feature vectors used for classification tasks described below are vectorized form of the causes extracted from the video frames. Also, sometimes we use different kinds of pooling on the causes depending on the dataset before feeding it to the classifier, which will be specified later. Given these feature vectors, we use a
linear L2-SVM (from LibLinear package [Fan et al., 2008]) for all the classification tasks described below.

5.4.1 Learning from Natural Video Sequences

Figure 5-3. Receptive fields of the two-layered network learn on natural video sequences. (Top) Receptive fields of layer-1 states. (Bottom) Receptive fields of layer-2 causes. They are constructed as a linear combination of bottom layer filters.

Firstly, to visualize what internal representations the model can learn, we construct a two layered network using the Hans van Hateren natural scene videos [van Hateren and Ruderman, 1998]. Here each frame is 128 x 128 pixels in size and is pre-processed as described above. The first layer consists of 16 states of 7 x 7 filters and 32 causes of 6 x 6 filters, while the second layer is made up of 64 states of 7 x 7 filters and 128 causes of 6 x 6 filters. The pooling size between the states and the causes for both the layers is 2 x 2. Figure 5-3 shows the receptive fields of the first layer states and the second
Table 5-1. Classification performance over Caltech-101 dataset with only a single bottom-up inference

<table>
<thead>
<tr>
<th>Methods</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our Method - Layer 1</td>
<td>62.1 ± 1.1%</td>
</tr>
<tr>
<td>Our Method - Layer 1 + 2</td>
<td>66.8 ± 0.5%</td>
</tr>
<tr>
<td>Zeiler et al. [2010] - Layer 1 + 2 (DN)</td>
<td>66.9 ± 1.1%</td>
</tr>
<tr>
<td>Lee et al. [2009] - Layer 1 + 2 (CDBN)</td>
<td>65.4 ± 0.5%</td>
</tr>
<tr>
<td>Kavukcuoglu et al. [2010a] (ConvPSD)</td>
<td>65.7 ± 0.7%</td>
</tr>
<tr>
<td>Chen et al. [2011] - Layer 1 + 2 (ConvFA)</td>
<td>65.7 ± 0.7%</td>
</tr>
<tr>
<td>Jarrett et al. [2009a] (PSD)</td>
<td>65.6 ± 1.0%</td>
</tr>
<tr>
<td>Boureau et al. [2010] (Macrofeatures)</td>
<td>70.9 ± 1.0%</td>
</tr>
<tr>
<td>Lazebnik et al. [2006] (SPM)</td>
<td>64.6 ± 0.7%</td>
</tr>
</tbody>
</table>

layer causes. We observe that the receptive fields of the first layer states (Figure 5-3 top) resemble simple oriented filters, similar to those obtained from sparse encoding methods [Olshausen and Field, 1996]. The receptive fields of the second layer causes, shown in Figure 5-3 (bottom), contains more complex structures like edge junctions and curves. These are constructed as weighted combination of the lower layer filters.

5.4.2 Object Recognition - Caltech-101 dataset

One advantage of the distributive models, like ours, is their ability to “transfer” the model learned on unlabeled data to extract features for generic object recognition, the so called self-taught learning [Raina et al., 2007]. We use this to access the quality of the learning procedure and perform object recognition in static image from Caltech-101 dataset [Fei-Fei et al., 2007]. Each image in the dataset is re-sized to be 152 × 152 (zero padded to preserve the aspect ratio) and pre-processed as described above.

We use the same two-layered model learned from natural videos sequences as above and extract features for each image using a single bottom-up inference (i.e., without any temporal or top-down information by setting λ = 0 and η = 0 for both the layers in (5–5)). The output causes from layer 1 and layer 2 are taken and are made into a three level spatial pyramid for each layer output [Lazebnik et al., 2006]. They both are
then concatenate to form a feature vector for each image and are fed as inputs to linear classifier. Table 5-1 shows the results obtained when 30 images per class are used for training and testing, following standard protocol, averaged over 10 runs. The parameters of the model are set through cross validation. We observe that using layer 1 causes alone leads to accuracy of 62.1%, while using the causes from both the layers improves the performance to 66.9%. These results is comparable to other similar methods that use convolution architecture [Lee et al., 2009, Kavukcuoglu et al., 2010a, Zeiler et al., 2010, Chen et al., 2011] and slightly better than using hand-designed features like SIFT [Lazebnik et al., 2006].

5.4.3 Recognition With Context

As discussed by Schwartz et al. [2007], visual perception is not static and uses contextual information from both space and time. We argue that our model can effectively utilizes this contextual information and produces a robust representation of the objects in video sequences. While the temporal relationships are encoded through the state-space model at each layer, the spatial context modulates the representation through two different mechanisms: (i) spatial convolution along with sparsity ensures that there is competition between elements, leading to some kind of interaction across space and (ii) the top-down modulations coming from higher-layer representations, which first accumulates information from the lower-layers and then tries to predict the response over a larger receptive fields.

In order to test this hypothesis, we show the performance of the model over two different tasks. Firstly, we show that using contextual information during inference can lead to a consistent representation of the objects, even in cases where there are large transformations of the object over time. We use the COIL-100 [Nene et al., 1996] dataset for this task. Secondly, we use the model for sequence labeling task, where we assign a class to each frame in sequence before classifying the entire sequence as a whole. The goal is to show the extent of invariance the model can encode, particularly
in cases of corrupted inputs. We test the performance on the Honda/UCSD face video dataset [Lee et al., 2005], on both clean as well as corrupted sequences.

5.4.3.1 Role of contextual information during inference

For this experiment we consider the COIL-100 dataset, which contains 100 different objects (or classes), Figure 5-4 shown some examples. For each object there is a sequence obtained by placing the object on a turn table and taking a picture for every 5 degree turn, resulting in 72 frame long video per object. Each frame is re-sized to be $128 \times 128$ pixel in size and pre-processed as described above. We use the same two-layered network described in Section 5.4.1 and perform inference with top-down connections over each of the sequences. We combine the causes from both the layers for each frame and use it to train a linear SVM for classification. As per the protocol described in Nene et al. [1996], we consider 4 frames per object at viewing angles $0^\circ, 90^\circ, 180^\circ, 270^\circ$ as labeled data used for training the classifier and the rest are used for testing. Notice that we assume that we have access to the test samples during
Table 5-2. Classification performance over COIL-100 dataset with various configurations.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>View-tuned network (VTU) [Wersing and Körner, 2003]</td>
<td>79.10%</td>
</tr>
<tr>
<td>Stacked ISA + temporal [Zou et al., 2012]</td>
<td>87.00%</td>
</tr>
<tr>
<td>ConvNets + temporal [Mobahi et al., 2009]</td>
<td>92.25%</td>
</tr>
<tr>
<td>CDN without context</td>
<td>79.45%</td>
</tr>
<tr>
<td>CDN + temporal (no top down)</td>
<td>94.41%</td>
</tr>
<tr>
<td>CDN + temporal + top down</td>
<td>98.34%</td>
</tr>
</tbody>
</table>

training. This resembles the “transductive” learning setting described by Mobahi et al. [2009].

Here we compare the proposed method with other deep learning models—a two stage hierarchical model built using more biologically possible feature detectors called View-tuned network (VTU) [Wersing and Körner, 2003], stacked independent subspace analysis learned with temporal regularization (Stacked ISA + Temporal) [Zou et al., 2012] and convolutional networks trained with temporal regularization (ConvNet + Temporal) [Mobahi et al., 2009]. While the first two methods do not utilize the contextual information during training the classifier, Mobahi et al. [2009] uses a similar setting as ours where the entire object sequence is considered during training. Also, we consider three different settings during inference in our model: (i) each frame processed independently and does not consider any contextual information, i.e., no temporal or top-down connections, (CDN without context) (ii) with only temporal connections (CDN + temporal (no top down)) and (iii) with both the temporal and top-down connections (CDN + temporal + top down).

As shown in Table 5-2, our method performs much better than the other methods when contextual information is used. While using temporal connections itself proved
Figure 5-5. Part of the face sequence (from left to right) belonging to three different subjects extracted from Honda/UCSD dataset (after histogram equalization).

sufficient to obtain good performance, having top-down connections improved the performance further. On the other hand, not using any contextual information leads to significant drop in performance. Also, we would like to emphasis that the model is learned on video sequences completely unrelated to the task, indicating that the contextual information during inference is more important than using it for just training the classifier as in Mobahi et al. [2009]. The reason for this could be that contextual information might push the representations from each sequence into a well defined attractor, separating it from other classes (more about this in Section 5.4.4).

5.4.3.2 Sequential labeling

While the above experiment shows the role of context during inference, it does not tell much about the discriminability of the model itself. For this, in the following experiment we test the performance of the proposed model for sequence labeling task, where the goal is classify a probe sequence given a set of labeled training sequences.

Here we conduct this experiment for face recognition on the Honda/UCSD dataset [Lee et al., 2005] and the YouTube celebrities dataset [Kim et al., 2008]. The Honda dataset contains 59 videos of 20 different subjects, while the YouTube dataset contains 1910 videos of 47 subjects. We note here that, while the Honda dataset is obtained from a controlled environment, the YouTube dataset is obtained from more natural setting, with very noisy and low-resolution videos, making the task very challenging. Figure 5-6A show some example videos from the YouTube dataset.
Figure 5-6. Example sequences from the Youtube celebrities dataset. (A) Original videos. (B) Extracted face sequences.

First, from every video, the faces from each frame are detected using Voila-Jones face detection [Viola and Jones, 2001] and then re-sized to be $20 \times 20$ pixel for the Honda dataset and $30 \times 30$ pixel for the YouTube dataset. Figure 5-5 and Figure 5-6B shows some example face sequences obtained from the Honda and the YouTube datasets, respectively. Each set of faces detected from a video are then considered as an observation sequence. Next, in addition to the pre-processing described above, we also perform histogram equalization on each frame to remove any illumination variations. Finally, following standard protocol [Lee et al., 2005], for the Honda dataset we consider 20 face sequences for training and the rest 39 sequences for testing. We report results using varying number of frames per sequence (N) as defined in [Hu et al., 2011]: 50, 100 and full length. When the length of the sequence is less than N, then all the frames in the sequence are used. In the YouTube dataset, we first randomly partition the dataset into 10 subsets of 9 videos each, then divide each subset into 3 videos for
<table>
<thead>
<tr>
<th>Sequence Length</th>
<th>50 frames</th>
<th>100 frames</th>
<th>Full length</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDA [Wang and Chen, 2009]</td>
<td>74.36</td>
<td>94.87</td>
<td>97.44</td>
<td>88.89</td>
</tr>
<tr>
<td>AHISD [Cevikalp and Triggs, 2010]</td>
<td>87.18</td>
<td>84.74</td>
<td>89.74</td>
<td>87.18</td>
</tr>
<tr>
<td>CHSID [Cevikalp and Triggs, 2010]</td>
<td>82.05</td>
<td>84.62</td>
<td>92.31</td>
<td>86.33</td>
</tr>
<tr>
<td>SANP [Hu et al., 2011]</td>
<td>84.62</td>
<td>92.31</td>
<td>100</td>
<td>92.31</td>
</tr>
<tr>
<td>DFRV [Chen et al., 2012b]</td>
<td>89.74</td>
<td>97.44</td>
<td>97.44</td>
<td>94.87</td>
</tr>
<tr>
<td>CDN w/o context</td>
<td>89.74</td>
<td>97.44</td>
<td>97.44</td>
<td>94.87</td>
</tr>
<tr>
<td>CDN with context</td>
<td><strong>92.31</strong></td>
<td><strong>100</strong></td>
<td><strong>100</strong></td>
<td><strong>97.43</strong></td>
</tr>
</tbody>
</table>

For the Honda dataset, we use the 20 training sequences to learn a two-layered network, with the first layer made up of 16 states and 48 causes and the second layer is made up of 64 states and 100 causes. All the filters are of \(5 \times 5\) in size and the pooling size in both the layers is \(2 \times 2\). We use a similar architecture for the YouTube dataset as well, but with filter size \(7 \times 7\) and the model parameters are learned by randomly sampling from all the sequences in the dataset. We emphasize that the learning is completely unsupervised. During classification, for the Honda dataset, the inferred causes from both the layers for each frame are concatenated and are used as feature vectors. On the other hand, for the YouTube dataset, we make a 3-level spatial pyramid of the causes from both the layers [Lazebnik et al., 2006] and use it as a feature vector. Any probe sequence is assigned a class based on the maximally polled predicted label across all the frames. All the parameters are set after performing a parameter sweep to find the best performance\(^3\).

\(^3\) On the YouTube dataset, parameter sweep is done a single subset and the same parameters are used for the rest of the subsets.
Table 5-4. Classification performance over YouTube Celebrities dataset.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDA [Wang and Chen, 2009]</td>
<td>65.3%</td>
</tr>
<tr>
<td>SANP [Hu et al., 2011]</td>
<td>68.4%</td>
</tr>
<tr>
<td>COV+PLS [Wang et al., 2012]</td>
<td>70.0%</td>
</tr>
<tr>
<td>COV+KL [Vemulapalli et al., 2013]</td>
<td>73.2%</td>
</tr>
<tr>
<td>Proj.+KL [Vemulapalli et al., 2013]</td>
<td>70.8%</td>
</tr>
<tr>
<td>CDN w/o Context</td>
<td>69.5%</td>
</tr>
<tr>
<td>CDN with Context</td>
<td>71.4%</td>
</tr>
</tbody>
</table>

Table 5-3 summarizes the results obtained on the Honda/UCSD dataset. We compare our method here with manifold discriminant analysis (MDA) [Wang and Chen, 2009], set based face recognition methods (AHISD and CHSID) [Cevikalp and Triggs, 2010], sparse approximated nearest points (SANP) [Hu et al., 2011] and dictionary-based face recognition from video (DFRV) [Chen et al., 2012b]. Our method (CDN with context) clearly outperforms all these methods, across all the sequence lengths considered. We also note that the performance of the proposed model drops when temporal and top-down connections are not considered (CDN w/o context).

On the YouTube dataset, we compare our method, in addition to SANP [Hu et al., 2011] and MDA [Wang and Chen, 2009], with other methods that use covariance features (COV+PLS) [Wang et al., 2012], kernel learning (COV+KL and Proj.+KL) [Vemulapalli et al., 2013]). As shown Table 5-4, the proposed model is competitive with the other state-of-the-art methods. We note here that most of the methods mentioned above (particularly, COV+PLS, Proj.+PLS and COV+KL) consider all the frames in the sequence to extract features before performing classification. On the other hand, we perform a sequential labeling, utilizing knowledge only from the past frames to extract the features. Also, without either the temporal or top-down connections, the performance of the proposed method again drops to around 69.5% (CDN w/o context).
Finally, to evaluate the performance of the model with a noisy observation, we corrupt the Honda/UCSD sequences with some structured noise in the above experiment (but maintain the same parameters learned from clean sequences). We make the noisy sequence as follows: one-half of each frame of all the sequences is corrupted by adding one-half of a randomly chosen frame of random subject. We repeat this a number of times per frame (the number is based on a Possion distribution with mean 2). Figure 5-7 summarizes the classification results (per sequence (Figure 5-7A) and per frame (Figure 5-7B)) obtained with sequence length of 50 frames. While the performance on the proposed model drops in both the cases, i.e., with and without temporal and top-down connections (denoted as CDN with context and CDN w/o context, respectively), the performance drop is steep when contextual information is not used than when it is used. The difference is more prominent in the classification accuracy per frame. For comparison, we also show the performance of SANP, whose performance drops significantly with noise.
Figure 5-8. The performance on noisy Honda/UCSD dataset for various values of $\lambda$ and $\eta$. (A)-(C) shows recognition rates versus temporal connection parameter ($\lambda$), where each color plot indicates a particular value of $\eta$, while (B)-(D) shows recognition rates versus top-down connections parameter ($\eta$), where each color plot indicates a particular value of $\lambda$. Also, (A)-(B) shows the recognition rates per sequence and (C)-(D) shows recognition rates per frame. Note that the higher recognition rates per frame does not always reflect as higher recognition rates per sequence.

5.4.3.3 Analysis of temporal and top-down connections

To understand the extent of influence the temporal and top-down connections have on the representations, we vary the hyper-parameters $\lambda$ and $\eta$ in (5–5), which determine the extent of influence they have, respectively, during inference. We use the
same experimental setup as above with noisy Honda/UCSD sequences and record the classification performance (per sequence and per frame) for different $\lambda$ and $\eta$ values. To make the visualization easier, we use the same set of hyper-parameters for both the layers, with sparsity parameters fixed at $\gamma_0 = 0.3$ and $\beta = 0.05$, which are obtained after performing a parameter sweep for best performance. Figure 5-8 shows the recognition rate on the noisy Honda/UCSD data set, as a function of both temporal connection parameter ($\lambda$) and top-down connection parameter ($\eta$). We observe that the performance is dependent on both the parameters and should be set reasonably (neither too high nor too low) to obtain good results.

While these plots show the effective contribution of temporal and top-down connections, they also show something more interesting. While the performance is better with either temporal or top-down connections, best performance is obtained only when both are available. This indicates that both temporal and top-down connections play an equally important role during inference. This is in accordance with other predictive coding models used for detecting bird songs [Friston and Kiebel, 2009].

5.4.4 Learning Hierarchy of Attractors

In this section, we further analyze the model from a slightly different perspective. The aim here is to visualize and understand the representations learned in the hierarchical model and get some insight into the working of the top-down and temporal connections.

The key assumption in our model is that any visual input sequence unfolds with a well defined spatio-temporal dynamics [George and Hawkins, 2005, Friston and Kiebel, 2009] and these dynamics can be modeled as trajectories in some underlying attractor manifold. In the hierarchical setting, we further assume that the shape of manifold that describes the inputs is itself modulated by the dynamics in an even higher level attractor manifold. From a generative model perspective, this is equivalent to saying that a sequence of causes in a higher layer non-linearly modulate the dynamics of a
lower layer representations, which in turn represent an input sequence. In other words, as succinctly described by Friston and Kiebel [2009], such hierarchical dynamic model represent the inputs as "sequences of sequences".

In the following experiments, we show that the model can learn hierarchy of attractors, such that the complexity of the representation increases with the depth of the model. Also, we show that the temporal and top-down connections (or empirical priors) lead the representations into stable attractors, making them robust to noise.

5.4.4.1 Learning parts of object from unlabeled sequences

We show that the model can learn the hierarchical compositions of the objects from the data itself in a completely unsupervised manner. For this, we consider the VidTIMIT dataset [Sanderson, 2008], where face videos of 16 different people with different facial expressions are used as inputs.

We learn two-layered networks with 16 first layer states, 36 first layer causes, 36 second layer states and 16 second layered causes. We further use 3 x 3 non-overlapping pooling regions in the first layer and 2 x 2 non-overlapping pooling regions for the second layer. We construct the receptive fields of the layer 1 and layer 2 causes using the linear combination of the basis in the layers below and shown in Figure 5-9.

We observe that the model is able to learn a hierarchical structure of the faces. While the first layer states represent primitive features like edges, first layer causes learn parts of the faces. The second layer causes, where the model combines the responses of the first layer causes, are able to represent an entire face. More importantly, we observe that each cause unit in the second layer is specific to a particular face (or object), increasing the discriminability between faces.

5.4.4.2 Denoising videos using top-down information

We next show that the top-down information can be useful to denoise a highly corrupted video by using the information from context. To show this, we use the same model as above on the face video sequences. We corrupt a face video sequence
Figure 5-9. Hierarchical decomposition of object parts learned by the model from face videos of 16 different subjects in the VidTIMIT dataset. (A) The receptive fields of layer 1 causes. (B) The receptive fields of layer 2 causes. Both are constructed as weighted linear combination of filters in the layers below. Layer 1 states have similar receptive fields as shown in Figure 5-3 (Top)
Figure 5-10. Video denoising with temporal and top-down connections. (A)-(B) shows examples with two different video sequences. For each example, (Top) corrupted video sequences where in every frame one-forth of the frame is occluded with an unrelated image, (Middle) the linear projection of layer 2 states onto the image space when inference is performed with temporal and top-down connections and (Bottom) linear projection of layer 2 states when inference is performed without temporal or top-down connections.
Figure 5-11. The PCA projections of layer two causes in the denoising experiment without (a) and with (b) temporal and top-down connections.
(different from the one used to learn the model) with a structured noise, where one-fourth part of each frame is occluded with a completely unrelated image. There is no correlation between the occlusion in two consecutive frames. Figure 5-10 and Figure 5-11 shows the results obtained⁴. In Figure 5-10 we project the response of the layer two states into the input space to understand the underlying representation of the model. Since layer two states get information from the bottom layer as well as the top-down information from the second layer causes, it should be able to resolve the occluded portion of the video sequence using the contextual information over time and space. We observe that with the top-down information the representation over time gets stabilized and the model is able to resolve the occluded part of the input video sequence. On the other hand, without the contextual information the representations did not converge to a stable solution.

Figure 5-11 shows the 2D PCA projections of the layer 2 causes. Again, we observe that representations obtained with temporal and top-down connections for each subject are stable and mapped into a well defined attractors, separated from one another (Figure 5-11B). On the other hand, without these connections the representations are not stable and cannot be well separated (Figure 5-11A).

5.5 Discussion

5.5.1 Relationship with Feed-Forward Networks

Many deep learning methods — like deep belief networks [Hinton et al., 2006], stacked auto-encoders [Vincent et al., 2010], convolutional neural networks [LeCun et al., 1989], etc — encode the inputs as a hierarchical representation. It is observed that increasingly invariant representations can be obtained with the depth of the hierarchical models [Goodfellow et al., 2009]. However, in contrast to our model,

---

⁴ These results are made into videos and they are made available at http://cnel.ufl.edu/~rakesh/face_video_2013.html
these methods neither perform explaining away nor consider temporal and top-down connections, and only focus on feed-forward “rapid” recognition without context.

In fact, the proposed model can also be written as a feed-forward network by performing approximate inference. Starting from initial rest (i.e., all the variables are initialized to zeros) and considering only a single FISTA iteration, the states and the causes can be (approximately) inferred as [Denil and de Freitas, 2012]:

\[
X^{l,k}_t = \frac{1}{L} T_{\gamma_0} \left( \sum_{m=1}^{D_{l-1}} c_{k,m} \ast U^{m,l-1}_t \right) \\
U^{l,d}_t = \frac{1}{L} T_{\beta} \left( \sum_{k=1}^{K_l} B_{k,m} \ast X^{k,l}_t \right)
\]

where \( T_{\gamma}(\cdot) \) is a soft thresholding function and \( L \) determines the step-size. But such representations have only a limited capacity, as there is no competition between the elements to explain the inputs. On the Caltech-101 dataset experiment described in section 5.4.2 such approximate inference only produced a modest recognition rate of 46\% \(^5\) (chance is below 1\%).

### 5.5.2 Comparison with Other Methods

The methods that are closest to our approach are those involving convolutional sparse coding [Zeiler et al., 2010, 2011]. Deconvolutional networks (DN) uses a similar hierarchical sparse coding but does not consider temporal or top-down connections. Our method can be considered as a generalization of DN to a broader class of dynamic systems. Also, pooling with switch setting [Zeiler et al., 2011] can also be incorporated into our model without any significant changes.

---

\(^5\) It should be noted that we do not perform any local contrast normalization between layers, which is reported to produce better performance in feed-forward networks [Boureau et al., 2010].
Deep Boltzmann machine (DBM) [Salakhutdinov and Hinton, 2012] and convolutional deep belief networks (CDBN) [Lee et al., 2009] uses undirected graphical models to construct deep networks. Similar to our model, they also incorporate top-down connections, though they do not consider temporal connections. However, they rely of sampling methods to perform inference and requires several iterations across the layers before a stable solution is obtained. In contrast, in our method we only perform a single top-down and bottom-up pass. Also, learning in these models is slow.

5.6 Summary

In this chapter, we proposed a novel convolutional dynamic model based on predictive coding framework. The curx of our approach is to build a hierarchical generative model by stacking several state-space models with sparse states and causes. The temporal (or recurrent) connections with-in each layer and interaction between top-down and bottom-up connections across the layers, allows the model to incorporate contextual information while extracting features from sequence of sensory signals. We have shown that features extracted are stable and robust to transformations and noise on the objects in the input sequence. Performance of the model in object recognition and sequence labeling datasets show that using contextual information can lead to significant gains in recognition rates.
CHAPTER 6
CONCLUSION

6.1 Summary

In this work, we proposed a new architecture for object recognition in video sequences based on predictive coding framework. The proposed hierarchical dynamic model, learned in a completely unsupervised manner, is able to extract rich structure from the data and is able to self-organize it in a hierarchy. The key component of the proposed model is its ability to better leverage the contextual information and find a good and robust representation of the sensory signals. The experimental results show that proposed model is able to outperform other methods in recognizing objects in video sequences and is competitive in case of static images. We also show the model is able to obtain a robust representation of the corrupt signals, leading to better recognition and denoising of the signals in challenging environments. We emphasis that these are possible only because of four important characteristics of the proposed architecture. Firstly, it models the sensory signals using a dynamic system. This provides a short-term memory to keep track of the time-varying changes in the input and maintain the contextual information during inference. Secondly, the model decomposes the signal in a hierarchical and distributive ways, such that low level features are combined together to form high level abstractions. These re-usability of the low and intermediate level feature extractors leads to compact representations and better generalization to unknown input signals. Thirdly, we impose sparsity on the representations at each level, which helps to obtain better discriminative representations. Lastly, it includes bottom up and top down components to provide context at the lower levels of processing. This bidirectional processing also allows for the use of past information to stabilize internal representations, and potentially the modulate the lower level feature extractors to represent relevant information from the inputs.
In chapter 3, we proposed dynamic sparse coding (DSC) where the input sequences are modeled using a linear dynamical system with appropriate sparsity constraint on the states and the innovations. We further extended this to model invariant representations in the input sequences. Our experiments have showed that the analysis of the inputs using such state-space models leads to more accurate and invariant representation of the inputs.

In chapter 4, we introduced deep predictive coding networks (DPCN) which uses the state-space model proposed in chapter 3 as a basic building block to construct a hierarchical network. In this hierarchical network we consider both bottom-up and top-down influences using the idea of empirical Bayes. We then proposed an efficient approximate inference procedure in this hierarchical network, that combines the bottom-up data driven information coming into the lower layers with the top-down expectation from the higher layers. We have shown that this bidirectional flow of information leads to extracting robust representations from noisy video sequences.

In chapter 5, we proposed a convolutional architecture for DPCN. This convolutional dynamic network allows the inference and the learning algorithms to scale to large images/frames and alleviates the problem of replicating the inference procedure multiple time at each level in DPCN. We have shown that this model is able to learn parts of the object and arrange them in a hierarchy from unlabeled video sequences. We have quantitatively shown the role of temporal and top-down connections have in obtaining robust representations for both recognition and de-noising tasks.

In summary, we designed a hierarchical dynamic mode inspired from predictive coding. This generative model embodies one of the key characteristic of biological vision, namely finding the representations of the sensory signals based on contextual information coming from spatio-temporal relationships (short-term memory) and top-down expectations (long term memory). Such a model is shown to be very useful in several object recognition tasks.
6.2 Avenues for Future Work

While in this work we focused on building an object recognition system inspired from biological vision, we believe it can be further extended to perform other important functions of the human visual system. For example, visual (or spatial) attention plays an important role during perception, allowing one to ignore irrelevant information while attending to important information in the scene [Gilbert and Li, 2013, Carrasco, 2011]. Several computational methods are proposed for modeling such visual attention, either using a bottom-up approach [Borji and Itti, 2013] or using hand-crafted architectures [Chikkerur et al., 2010]; however, autonomously attending to relevant aspect during perception in the scene still remains elusive. We believe that the top-down expectations, along with multiplicative interaction between the states and the causes in the proposed model, allows us to model such attention mechanism.

Also, in its current form, the proposed model has several limitations: (i) In spite of the efficient GPU implementations, the major bottleneck of the proposed is the inference time. However, because of the use of proximal methods, we believe we can build a recurrent network that can efficiently approximate this inference procedure [Gregor and LeCun, 2010]. (ii) The layer-wise learning approach adopted here does not allow the lower layers to adapt to more relevant information encoded in the higher layers. Such two-way interaction even while learning could produce better representations. (iii) While using the state-space models allowed us to model short-term temporal dependencies, encoding long term dependencies is still not possible within the proposed framework.
APPENDIX A
A FAST PROXIMAL METHOD OF CONVOLUTIONAL SPARSE CODING

Generally, sparse coding is based on the idea that an observation, \( y \in \mathbb{R}^p \), can be encoded using an over-complete dictionary of filters, \( C \in \mathbb{R}^{p \times k}; (k > p) \) and a sparse vector \( x \in \mathbb{R}^k \). More formally this can be written as

\[
\hat{x} = \arg \min_x \frac{1}{2} \|y - Cx\|_2^2 + \lambda \|x\|_1
\]  

(A–1)

The \( \ell_1 \)-norm on \( x \) ensures that the latent vector is sparse. Several efficient solvers like coordinate descent (CoD) [Li and Osher, 2009], fast iterative shrinkage thresholding algorithm (FISTA) [Beck and Teboulle, 2009], feature-sign algorithm [Lee et al., 2007b], etc, can be readily applied to solve the above optimization problem. The dictionary \( C \) can also be learned from the data [Lee et al., 2007b, Mairal et al., 2009].

However, in most applications of sparse coding many overlapping patches across the image are processed separately. This is often too slow in practice, making it difficult to scale to large images. Moreover, sparse coding alone is not capable of encoding translations in the observations. Learning the dictionary in this context produces several shifted versions of the same filter, such that each patch can be reconstructed individually [Kavukcuoglu et al., 2010a]. During inference, when performed on all the overlapping patches, this can lead to a very redundant representation. To overcome these limitations, convolutional sparse coding is proposed [Kavukcuoglu et al., 2010a, Zeiler et al., 2010]. Here sparse coding is applied over the entire image and the dictionary is a convolutional filter bank with ‘\( M \)’ kernels such that,

\[
x = \arg \min_x \frac{1}{2} \|I - \sum_{m=1}^{M} C_m * x_m\|_2^2 + \lambda \sum_{m=1}^{M} \|x_m\|_1
\]  

(A–2)
where \( I \) is an image of size \((w \times h)\), \( C_m \) is a filter kernel of size \((s \times s)\) in a dictionary, \( x_m \) is a sparse matrix of size \((w + s - 1) \times (h + s - 1)\), \( \lambda \) is the sparsity parameter and ' * ' represents a 2D convolution operator. 

In this work, we propose an algorithm to solve the optimization problem in (A–2) efficiently and which scales to large images. This is an extension to fast iterative shrinkage thresholding algorithm (FISTA) [Beck and Teboulle, 2009] and is solved using proximal gradient. In addition, we also extend our method to include a feed-forward predictor (predictive sparse decomposition [Kavukcuoglu et al., 2010b]) in the cost for joint optimization during inference and learning.

Convolutional sparse coding is also previously studied in [Kavukcuoglu et al., 2010a, Zeiler et al., 2010]. Zeiler et al. [2010] proposed a method to solve the optimization problem in (A–2) by introducing additional auxiliary variable which demands solving a large linear system (the size of the linear system is proportional to size of the image) at every iteration; although the complexity could be reduced by using conjugate gradient, their approach does not scale to large images. On the other hand, Kavukcuoglu et al. [2010a] proposed a convolutional extension to CoD where the computation per iteration is small compared to the method proposed by Zeiler et al. [2010] but the number of iterations required for convergence becomes large for large images. In the following sections, we compare and contrast convolutional CoD with our method to show the performance improvements achieved.

### A.1 Convolutional FISTA and Dictionary Learning

Since the cost function in (A–2) is not jointly convex in both \( C \) and \( x \), learning the dictionary of filters involves a block coordinate descent kind of optimization [Lee et al., 2007b, Mairal et al., 2009]; where in the objective in (A–2) is alteratively minimized

\[ \text{\textsuperscript{1}} \text{ All the variable henceforth represent a matrix, unless otherwise stated. Also the convolution operators is applied in ‘full’ or ‘valid’ modes, depending on the context.} \]
w.r.t $x$ and $C$ while keeping one of them fixed. This section describes the convolutional
generalization of the popular sparse coding algorithm FISTA for inferring the latent
variable $x$ while $C$ is held fixed. Then, a procedure for updating the parameters in $C$
with fixed $x$ is described. In addition to this, the effectiveness of the proposed method to
generalize to other cost functions is shown using predictive sparse decomposition (PSD)
[Kavukcuoglu et al., 2010a].

A.1.1 Inference

ISTA is a popular first order proximal method to infer sparse codes from a linear
inverse problem in (A–1). That is, it has the advantage of being a simple gradient
based algorithm involving simple computations like matrix multiplications with $C$
and $C^T$ followed by a soft thresholding function. However, it tends to have a very
slow convergence rate. To overcome this, Beck and Teboulle [2009] have proposed
FISTA, which has significantly better global convergence rate while preserving the
computational simplicity of ISTA. In fact, FISTA can be generalized to any non-smooth
convex optimization problem with cost function of the form:

$$F(x) = f(x) + g(x)$$ (A–3)

where $f$, $g$ are convex and $g$ is possibly non-smooth [Beck and Teboulle, 2009]. Most
notable advantage of FISTA is that it keeps ISTA’s number of gradient evaluations
(just one per iteration) but involves finding a point that is smartly chosen for updating
the latent variable, $x$. This additional point is a function of the difference between the
previous two updates of $x$. Such “momentum” term introduced into the update helps to
obtain faster convergence and is shown to have a convergence rate of order $O(1/k^2)$,
where $k$ is the number of iterations.

A straightforward way to use FISTA (or any sparse coding method) to solve (A–2) is
by replacing the convolution operator by a matrix-vector product. The matrix, equivalent
to the dictionary in (A–1), is made by first constructing a “Toeplitz” matrix for each filter
Algorithm 3 Convolutional extension for fast iterative shrinkage thresholding algorithm (FISTA).

Require: Input image: \(I, L^{(0)} > 0, \eta > 1\) and \(x^{(0)} \in \mathbb{R}^{(w+s-1) \times (h+s-1) \times M}\)

1: Initialize \(z^{(1)} = x^{(0)}, t^{(1)} = 1\) and \(k = 0\).
2: while (convergence) do
3: \(k = k + 1\)
4: Line search: Find the smallest non-negative integer \(i^{(k)}\) such that with \(L^{(k)} = \frac{1}{\eta} L^{(k-1)}\)
5: \(F(p_L(z^{(k)})) \leq Q_L(p_L(z^{(k)}), z^{(k)})\)
6: Set \(L^{(k)} = \eta^{i^{(k)}} L^{(k-1)}\)
7: \(x^{(k)} = p_L(z^{(k)})\)
8: \(t^{(k+1)} = \frac{1 + \sqrt{1 + 4t^{(k)^2}}}{2}\)
9: \(z^{(k+1)} = x^{(k)} + \left(\frac{t^{(k)}-1}{4t^{(k)^2}}\right)(x^{(k)} - x^{(k-1)})\)
10: end while
11: return \(x^{(k)}\)

containing all its shifted versions and then concatenating all such matrices. The resulting problem is similar to (A–1) and FISTA is guaranteed to converge for such a problem. However, in practice working on such large matrices can be computationally expensive and might be unnecessary if one can take advantage of the convolutional nature of the formulation.

As discussed before, FISTA involves computing the gradient of the convex part of the cost function in (A–2). So, the key to computational simplicity in this work comes from computing this gradient efficiently and can be obtained as follows: the derivative of \(f\) with respect to \(z_n\) (\(n\)th map corresponding to the \(n\)th filter) is given by

\[
\nabla f(z_n) = C'_n \ast (I - \sum_{m=1}^{M} C_m \ast z_m)
\]

where \(C'_n\) is equivalent to \(180^\circ\) rotation of matrix \(C\). This leads to very efficient computation of the gradient, while avoiding constructing any large matrices.

Equipped with this efficient gradient computation, we can easily extend FISTA for convolutional sparse coding. Algorithm 3 describes the convolutional generalization of
FISTA; where $p_{L_k}(.)$, for $\ell_1$ regularization in the cost function, is given by

$$p_{L_k}(z^{(k)}) = T_{\lambda/L_k}(z^{(k)} - \frac{1}{L_k} \nabla f(z^{(k)}))$$  \hspace{1cm} (A–5)

where $T_\alpha(.)$ is an elementwise soft thresholding function

$$T_\alpha(x_i) = (|x_i| - \alpha)_+ \text{sgn}(x_i)$$  \hspace{1cm} (A–6)

where $x_i$ indicates the $i$th element of a point $x$ in the latent space. $\nabla f(z)$ is the gradient of the quadratic term in (A–2), denoted by $f$, at some point $z$ in the latent space and is computed from (A–4). Also, the line search to find the appropriate step size, $L_k$, requires computing $\mathcal{F}(x)$, the cost function in (A–2), and $Q(p_L(x), x)$ given by

$$Q(p_L(x), x) = f(x) + \langle p_L(x) - x, \nabla f(x) \rangle + \frac{L}{2} \|p_L(x) - x\|_2^2 + |p_L(x)|_1$$

where $f(x)$ is the quadratic loss in (A–2) and $\langle . \rangle$ is the inner product between two vectorized quantities.

Note that, compared to convolutional CoD [Kavukcuoglu et al., 2010a], the computations required per iteration here are more. However, because of the “momentum” term introduced in step 7 of Algorithm 3, the number of iterations required for convergence are much less.

**A.1.2 Dictionary Learning**

With the inferred latent variable $x$ fixed, the dictionary of filters are updated using gradient descent method. For faster convergence limited-memory BFGS [Nocedal, 1980] is used for updating the filters over a batch of images. The gradient w.r.t to a single filter, $C_n$ is

$$\nabla f(C_n) = \sum_{b=1}^{B} x_{b,n}^t * (I_b - \sum_{m=1}^{M} C_m * x_{b,m})$$  \hspace{1cm} (A–7)
where $B$ is the number of images in a batch and $x'_{b,m}$ is computed in a similar manner to $C'_m$ described in (A–4). Each filter is normalized to have a unit norm post update to avoid any redundant solution.

The above described inference and learning procedure can be readily applied for image denoising [Elad and Aharon, 2006] and constructing deep networks like deconvolutional network [Zeiler et al., 2010]. However, unlike convolutional CoD, FISTA can be generalized to other cost functions easily, like predictive sparse decomposition (PSD) described below.

A.1.3 Extension to PSD

The cost function used in PSD [Kavukcuoglu et al., 2010b] contains two parts: a decoder, containing a quadratic cost function along with a sparsity constraint similar to (A–2) and feed-forward, non-linear encoder to approximate the sparse code. The cost function with this additional prediction term looks like:

$$x = \arg\min_x \frac{1}{2}\|I - \sum_{m=1}^{M} C_m * x_m\|_2^2 + \sum_{m=1}^{M} \|x_m - \varphi(W_m * I)\|_2^2 + \lambda \sum_{m=1}^{M} \|x_m\|_1$$

(A–8)

where $W_m \in \mathbb{R}^{s \times s}$ is a feed-forward filter and $\varphi(.)$ is a non-linear function to approximate sparse codes. For further details refer to Kavukcuoglu et al. [2010a].

Now, the first two quadratic terms in (A–8), referred to as $f$ below, are convex and continuously differentiable with respect to $x$ and this can be readily solved using convolutional FISTA described above with gradient w.r.t $x_n$ computed as

$$\nabla f(x_n) = C'_n * \left(I - \sum_{m=1}^{M} C_m * x_m\right) + \left(x_n - \varphi(W_n * I)\right)$$

Note that convolutional CoD cannot deal with this cost function, without losing its efficiency.
A.2 Experiments

In this section, convolutional FISTA (ConvFISTA) is compared with the other existing algorithms. Gray scale images of different sizes are used for testing purposes. All the images are first pre-processed such that their mean is subtracted and then contrast normalized using a $5 \times 5$ average filter. This step ensures that the low-frequency components are removed and high frequency edges are made prominent. This helps to speed up the learning and inference during sparse coding [Olshausen and Field, 1996].

A.2.1 Dictionary Learning

Face images from AT&T database [Samaria and Harter, 1994] are used for learning the dictionary. Each gray scale image is first resized to be $64 \times 64$ pixel and pre-processed as stated above. From here 8, $16 \times 16$ convolutional filters are learned using a batch of 20 images per epoch. Figure A-1 shows the results obtained. It is important to note that taking 8 convolutional filters makes the system that may times
Figure A-2. Comparison of the convergence rate between convolutional CoD (ConvCoD) and convolutional FISTA (ConvFISTA). Two different images sizes, (A) 50 × 50 pixel and (B) 150 × 150 pixel, are used for comparison.

over-complete at every point. This typically leads to a more sparse representation that what would be achieved from patch level sparse coding, because each element in the latent variables competes with the other filters not just to explain one point but a local neighborhood in the image.

A.2.2 Convergence Rate

As discussed before, one of the important advantages of ConvFISTA is its faster convergence rate during inference. To compare the performance of the proposed model with convolutional CoD (ConvCoD) [Kavukcuoglu et al., 2010a], the ability of both the methods to reconstruct the image, with a pre-learned dictionary, is considered as the metric. In other words, the relative error between the true (contrast normalized) image and reconstructed image obtained from the inferred latent representation

\[
\mathcal{M} = \frac{\|I - \hat{I}\|_2^2}{\|I\|_2^2}
\]  

where \(\hat{I}\) is the reconstructed image, is used as the metric.

To make a fair comparison, the same model is used for both the methods, i.e., the sparsity parameter is held fixed (\(\lambda = 0.1\)) and a fixed dictionary of 32, 9 × 9 filters learned
from Berkeley segmentation dataset [Martin et al., 2001] is used. Images from the same dataset of varying size are used to compare the performance with scale. Figure A-2 shows the comparison. It is observed that ConvFISTA is able to outperform ConvCoD in terms of convergence rate. The difference in performance becomes more pronounced with increase in the size of the images because ConvCoD, which updates only one carefully chosen element at every iteration, requires more iterations for convergence.

A.2.3 Predictive Sparse Decomposition

![Encoder weights W](image)

Figure A-3. (A) 36, 11 × 11 encoder filter weights learned using ConvFISTA. They resemble in the learned dictionary filters (not shown). (B) The instantaneous total loss at every epoch for both ConvFISTA and ConvCoD.

As described in Section A.1.3, predictive sparse coding (PSD), used as a basic block for building deep networks, has a decoder (involving $C$) and an feed-forward encoder (involving $W$) in the cost function as in (A–8). However, once the model is trained, the decoder is discarded while the encoder alone is used for approximate inference. Therefore, it is important to minimize the joint cost function, so that the encoder would be able to approximate the decoder function more accurately.

Here ConvFISTA is used for inferring the latent representation from the joint cost function involving both the decoder and the encoder. The non-linear function
used in the encoder is considered as in [Gregor and LeCun, 2010] with: \( \varphi_m(I) = \beta_m(tanh(W_m*I-\theta) + tanh(W_m*I+\theta)) \). The parameters of the encoder, \( \beta \) and \( W \), are also learned along with the convolution filters, \( C \) (each one is updated while all the others are held fixed). On the other hand, for comparison we use the procedure described in [Kavukcuoglu et al., 2010a], where the encoder is learned separately as feed-forward model, similar to convolutional networks [LeCun et al., 1989], after obtaining the sparse representation using ConvCoD. Results obtained a shown in Figure A-3. Similar to the previous experiment, Berkeley segmentation dataset [Martin et al., 2001] is used, with each image re-sized to be 150 \( \times \) 150 pixel and preprocessed. Figure A-3A shows the learned encoder weights corresponding to 36, 11 \( \times \) 11 convolutional filters. We observe (Figure A-3B) that the total loss, the quadratic term plus the prediction term in (A–8), reaches a lower value when using ConvFISTA for jointly optimizing the cost than using ConvCoD for separately training the model.

### A.3 Summary

In this work, we proposed a convolutional extension to the FISTA algorithm for solving the sparse coding problem. The convolutional FISTA proposed here has two major advantages: (1) Unlike ConvCoD, where only one carefully chosen element in updated at every iteration, ConvFISTA updates all the elements in parallel and achieves faster convergence rate and (2) because of simple first order updates, ConvFISTA can be easily generalized to other convex loss functions as well.

With the help of recent advances in using proximal methods for structured sparsity models [Chen et al., 2012a], it might be possible to extend the method proposed here to learn more complex structures within the convolutional filters; for example, introducing tree structured relationship between the filters might better represent the variations between different features within an image. Such extensions still remains to be studied.
**APPENDIX B**

**ADDITIONAL RESULTS FOR MODEL VISUALIZATION**

**B.1 Visualizing Invariance Encoded by Layer 1 Causes**

From Section 3.3.2.1, we here show the invariance encoded by all the 100 layer 1 causes. In addition to the frequency-orientation polar plot and center-orientation scatter plot, we also show here the grouping of the dictionary elements corresponding to each column of the invariance matrix $\mathbf{B}$.

![Figure B-1. Visualizing grouping of the dictionary elements encoded by layer 1 causes.](image)

From left to right and top to bottom, every 5 blocks indicate a group of layer 1 dictionary elements (or states) that are strongly connected to a cause dimension through the invariance matrix $\mathbf{B}$. 
Figure B-2. Visualizing invariance encoded by the layer 1 causes using frequency-orientation polar plot. Each subplot here indicates one column of the invariance matrix $\mathbf{B}$. 
Figure B-3. Visualizing invariance encoded by the layer 1 causes using center-orientation scatter plot. Each subplot here indicates one column of the invariance matrix $B$.

**B.2 Hierarchical Decomposition Obtained from the YouTube dataset**

In Section 5.4.3.2, we discussed classification performance of the convolutional dynamic model on the YouTube celebrities dataset. Here we visualize the model learned
for this task and show that the model learns a hierarchical decomposition of the faces in the dataset.

Figure B-4. Receptive fields in a two-layered network learned from the YouTube celebrities dataset. (Top) Shows the receptive fields of subset of layer 1 causes. (Bottom) Shows the receptive fields of subset of layer 2 causes. Both are constructed as linear combination of lower layer filters.
REFERENCES


BIOGRAPHICAL SKETCH

Rakesh Chalasani was born in Mopidevi, India in 1986. He received his Bachelor of Technology degree in electronics and communication engineering from National Institute of Technology, Nagpur, India in 2008. He received Master of Science and PhD in electrical and computer engineering from University of Florida in 2010 and 2013, respectively. He also worked as research intern at Bosch Research and Technology Center during Summer, 2012. His research interests include machine learning, pattern recognition, unsupervised learning, kernel methods, information theoretic learning and computer vision.