University of California

Los Angeles

Unsupervised Learning of Object Descriptors and Compositions

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Statistics

by

Xingyao Ye

2012

Abstract of the Dissertation

Unsupervised Learning of Object Descriptors and Compositions

by

Xingyao Ye

Doctor of Philosophy in Statistics

University of California, Los Angeles, 2012

Professor Alan L. Yuille, Chair

This thesis presents methods and results to solve the problem of joint object recognition and reconstruction. The proposed solution is a dictionary of deformable image patches and a hierarchical model encoding spatial compositions. Both the dictionary and the composition model are learned from data without supervision. The patch dictionary is shown to achieve state-of-art performance on digit recognition while capable of high-quality reconstruction. The hierarchical model is shown to account for human chunk learning behavior not captured by previous theories. Both learning algorithms are significantly faster and easier to use than previous methods of similar purpose.

Table of Contents

1 Introduction 1

1.1 Part I: Learning a Dictionary of Generative Descriptors 3

1.2 Part II: Learning Spatial Compositions 4

1.3 Previous Works 6

2 A Dictionary of Redundant, Deformable Image Patches 8

2.1 The D-Patch Representation 8

2.1.1 Transformations 9

2.1.2 Redundancy 10

2.2 Matching Patches To An Image 11

2.3 Joint Recognition and Reconstruction 13

2.3.1 Object Classification 13

2.3.2 Reconstruction 13

2.4 Will A Random Dictionary Do The Trick? 14

3 Learning the D-Patch Dictionary 18

3.1 The Learning Algorithm 18

3.1.1 Seed Clustering 18

3.1.2 Applying Transformations 19

3.1.3 Parallel Matching using GPUs 20

3.1.4 Feature Selection 21

3.2 The Learned Dictionaries 21

3.3 State-of-Art Digit Recognition and Reconstruction 24

3.4 Dictionary Transfer 25

3.5 Conclusions 26

4 Modeling Human Visual Chunk Learning Behavior 31

4.1 Background 32

4.2 The Bayesian Chunk Learner (BCL) 33

4.3 The Hierarchical Chunk Learner 34

4.3.1 Model Formulation 34

4.3.2 Maximum-Likelihood Based Inference 36

4.4 Learning The Hierarchical Chunk Model 36

4.5 Predicting Human Chunk Learning Behavior 38

4.5.1 Methods 38

4.5.2 Psychophysical Experiment Settings 39

4.5.3 Results on Previously Reported Experiments 40

4.6 A New Experiment on Human Learning of Hierarchical Visual Chunks 41

4.6.1 Motivations 41

4.6.2 Experiment Design 42

4.6.3 Experiment Results 43

4.7 Conclusions 44

5 Summary and Discussions 51

References 53

List of Figures

2.1 Illustration of the D-Patch representation. The dictionary of deformable patches are generated from a set of seeds plus transformations. Patches are perturbed and matched to local image regions. 9

2.2 Applying transformations to a seed patch. 10

2.3 Two digit images from the MNIST dataset (top left panel) together with their reconstructed versions using our learned dictionary (adjacent panel) and a complete list of patches that are activated by respective images. The patches are sorted in descending order of their matching scores. 17

3.1 Examples of learned seeds and patches. The left half of the figure shows seeds learned from MNIST, as well as the most frequent (up to 5) bases from some of the seeds. The right half shows shows seeds and patches from USPS. 23

3.2 Synthesized images using the learned and transferred D-Patch dictionaries. The top half shows 100 reconstructed MNIST images synthesized from a dictionary learned on the same dataset. The bottom half shows synthesized USPS images by a dictionary learned on USPS. 28

3.3 Using learned bases as features for recognition. For two MNIST images, the activation histograms are shown for some of the fired bases. Observe that some bases are very discriminative, e.g. U-shape for 4, X-junction for 8, etc. 29

3.4 Synthesized images using transferred D-Patch dictionaries. The top half shows 100 reconstructed MNIST images synthesized from a dictionary learned on USPS. The bottom half shows the other way around. 30

4.1 An illustration of the proposed hierarchical model. The training scenes in the left column are generated by the inventory of chunks in the middle column. The right column shows how our hierarchical model and the Bayesian chunk learner model (BCL) may provide different explanations over the same training scenes. 46

4.2 Searching for the model's maximum-likelihood states in two rounds of belief propagation. Black, white, or striped nodes indicate that their ML states are on, off, or undecided respectively. 47

4.3 Learning the model structure in two stages. Top: the model composition stage starting from 4 basic features. Composed chunks are listed in a queue. Rectangles show the likelihood ratio tests performed during the process. Bottom: the model selection stage for the same features. 48

4.4 Illustration of previous experiments on human chunk learning, along with predictions from our model and the BCL model. Top: the general procedure of these experiments. Middle: chunk inventory and results for an experiment in both [FA05] and [OFA08]. Bottom: chunk inventory and results for another experiment in [OFA08]. . 49

4.5 Top: the chunk inventory and example training scenes used in our new experiment. Colors are for illustration purpose only and were not used in the experiment. Bottom: testing trials used in the experiment, along with human performance and our model's prediction. 50

List of Tables

2.1 List of transformation settings for a seed patch with original size (s_0^x, s_0^y) onto an image of size (s_I^x, s_I^y). 10

2.2 Error rates of MNIST digit classification. 15

3.1 Statistics of the learned dictionaries. 22

3.2 Comparison of state-of-art digit recognition methods on benchmark datasets. 25

3.3 Recognition using dictionaries learned on another dataset. Results are compared between transferring both seeds and patches, transferring seeds only, and no transfer at all. 26

Vita

Summer 2007	Research Intern, Microsoft Research Asia, Beijing, China.
2008	B.Eng. in Computer Software, Tsinghua University, Beijing, China.
Summer 2010	Software Engineering Intern, Google Inc., Mountain View, California.
Summer 2011	Software Engineering Intern, Facebook Inc., Palo Alto, California.
2008–2012	Research Assistant, Department of Statistics, UCLA, Los Angeles, California.

Publications

Xingyao Ye, Alan L. Yuille. *Learning a Dictionary of Deformable Patches using GPUs.* ICCV 2011 Workshop on GPU in Computer Vision Applications.

Long Zhu, Yuanhao Chen, Xingyao Ye, Alan L. Yuille. *Structure-Perceptron Learning of a Hierarchical Log-linear Model.* CVPR 2008.

Min Zhang, Xingyao Ye. *A Generation Model to Unify Topic Relevance and Lexicon-based Sentiment for Opinion Retrieval.* SIGIR 2008.

CHAPTER 1

Introduction

Learning effective mid-level visual representations has long been a topic of interest in computer vision research. Such a representation facilitate the information flow of a visual system in both bottom-up and top-down directions. In the bottom-up flow, raw images are mapped to abstract concepts and categories. Core tasks of computer vision, such as object detection and recognition, belong to this category. In the top-down flow, realistic images are generated from specific high-level concepts. The whole field of computer graphics is devoted to this problem.

There are many advantages of using a single representation for all kinds of bottom-up and top-down tasks. First of all, a shared representation is the most economical solution to deal with the sheer size of possible visual tasks. The human brain, the best vision system we naturally possess, is able to handle a variety of vision and graphics tasks effortlessly. It would not be possible for the brain to train a separate module to solve each task it encounters, for instance, when recognizing a new object category. The components of a good representation must be shared and reusable for all possible visual tasks.

A second advantage is that generative features, which possess the capability of reconstructing the input images, are much more intuitive than discriminative features, which are adopted mainly because of their superior classification performance. Intuitiveness helps identify the effectiveness of individual components within the representational framework, and provides better insights on how to improve them. For discriminative features, progress are made on a more trial-

and-error basis, fine-tuning many parameters in the hope of the desired results popping out.

Nevertheless, there are several challenges to learning an effective generative representation. The first one is to compete with discriminative features on recognition performance. In recent years, discriminative features such as SIFT and HOG enjoy massive popularity due to their good performance on object recognition benchmark datasets. No generative features are able to compete directly with them on any widely-accepted datasets. If their recognition performance can level with that of discriminative features, there would be very few reasons not to use generative features because of the advantages discussed above.

The second challenge is the speed of the algorithm. It is difficult to expect a representation so general as the one we are proposing here to be faster than discriminative methods. However we do not want to compromise on this point. Part of the reason for the popularity of discriminative features comes from its relative fast speed over cumbersome generative ones which try to model everything at the same time. And to turn computer vision progress into high-impact consumer products, we have to keep the complexity to a minimum and the algorithm as near real-time as possible. Basically anything more than linear complexity is not good enough for the industry to apply to the massive data they possess. So we made an explicit decision to use linear and sub-linear complexity algorithm all the time.

State-of-art recognition systems are rarely based on generative representations for two reasons. First, learning such representation is complex and takes a long time, and the gain in recognition performance over popular hand-craft features, such as HOG and shape context, is rarely good enough to compensate for this cost. Second, many of these systems restrict themselves to recognition only, and therefore do not require their features to preserve those image details that are useful for reconstruction. We believe, however, that image dictionaries will be increasingly important in the future. Future advanced computer vision systems will need to work on multiple tasks simultaneously, which will encourage them to adopt the more versatile dictionary representation. But, to achieve this future requires us to deal with the learning complexity issue and motivates developing rapid learning algorithms perhaps taking advantage of recent developments in computer hardware.

1.1 Part I: Learning a Dictionary of Generative Descriptors

In this thesis, we present models and algorithms for learning a shared generative representation for multiple object categories. Our approach has two main parts. One is a dictionary of deformable image patches. The dictionary we learn is comprehensive – the patches are of varying sizes, capturing generic patterns as well as object-specific structures. Moreover, we neither provide the category labels nor specify the size of the dictionary during training, making the learning task harder but more realistic. As a result, we explore a search space of image patterns much larger than what is usually framed by interest point detectors or random initializations.

We deal with this challenge by combining two innovations. First, we define a set of common transformations of patches, and use them to guide our search of suitable dictionary elements. Instead of trying to collect and cluster all possible patches in the training images, we construct a preliminary patch set by imposing transformations on a small random set of seed patches. We have found the resulting patch set to be very comprehensive for our needs, capturing almost all useful patterns for discrimination and reconstruction.

Second, we design a parallel matching framework on Graphical Processors (GPUs) for evaluating the preliminary patch set on training images. The basic matching operation computes a similarity measure between a patch and a local image area. And we use the aggregated similarity statistics as the criteria for selecting patches into the dictionary. The challenge is that we have to perform this operation for billions of times during training, because both the preliminary patch set and the training image set are very large. Such scale of computation is impossible for a single CPU workstation to handle. Buying or renting thousands of distributed machines for developing this algorithm is not cost-effective for research labs either. Therefore, we turn to the recently-available commercial GPU cards, which are specifically designed to handle such massively parallel computations.

We demonstrate our algorithm and the effectiveness of the learned dictionary on handwritten digits. The results are preliminary since digits lack the rich texture of real images. Nevertheless, images of digits possess great variability in shapes and plenty of intra-class and inter-class ambiguities. We report state-of-art recognition results on two extensively studied datasets of handwritten digits, outperforming previous dictionary learning methods as well as classic hand-craft features like shape context, without using specific domain knowledge or any complex classifiers. In addition, our trained features on one dataset transfer very well to the other, which proves the generality of our approach. Last but not least, our dictionary is relatively fast to train. And recognition using our dictionary takes much shorter time than competing methods thanks again to the prowess of GPUs.

1.2 Part II: Learning Spatial Compositions

The other crucial component of a generative object representation is the spatial compositions. Descriptors alone can form a naive bag-of-words type model and achieve fairly good results on many vision tasks. But they are not enough to address challenges posted by a lot of common object categories that are articulated. For example, the human limbs are highly articulated subparts of the body. If we want to detect them purely by storing patch templates and matching, there would be too many possible combinations to remember. But this complexity can be greatly reduced if we introduce a compositional model which encodes the probability distribution of relative spatial positions of all subparts.

The second part of this thesis proposes a novel hierarchical chunk learning framework to detect suspicious coincidences of elements during training, and to form efficient representations of complex visual scenes. We choose to demonstrate our model's capability on the problem of predicting human chunk learning behavior in the area of psychophysics. In this area, basic visual features are represented by highly abstract symbols, so we are freed from the ambiguous nature of real image patterns and can focus on evaluating the effectiveness of the spatial composition model we learned.

The most advanced theory for this problem used to be an ideal Bayesian model, which has been shown to account for human performance in learning chunks by passively viewing complex visual scenes. However, this model assumes independence among a single layer of visual chunks, and is very time-consuming to train or test due to the sampling method used to explore the large structure space.

In Chapter 4, we describe a hierarchical visual chunk model coupled with a novel structure learning algorithm, which addresses these problems. Our model captures part-to-whole relations between visual chunks, and is able to learn explicit object representations in an unsupervised manner. The learning algorithm employs data-driven methods to recover an inventory of visual chunks as well as their most probable relations. It takes less than a minute to run, much faster than the several days of sampling required by previous models. Our model is able to predict human performance on not only previously reported experiments, but also a newly designed one that addressed the learning of spatial configurations and the subparts of visual chunks, respectively.

1.3 Previous Works

There has long been interest in describing images in terms of generative dictionaries to provide adaptive representations for a variety of vision tasks. Examples include modeling receptive fields [OF96], texture [EF01], appearances [BPP08], and object categories [FJ03]. In this thesis, we are particularly interested in the use of dictionaries for object recognition and reconstruction.

Various types of generative descriptors have been designed and tested over recent years. Ullman and collaborators have advocated the use of intensity patches in segmentation [BU02] as well as object recognition [EU05]. But their patches were not deformable. Wu *et al* [WSF07] introduced local deformation in their active basis model, but the bases were pre-specified to be Gabor functions. Zhu *et al* [ZCT10] and Fidler *et al* [FL07] proposed methods to learn hierarchical object models based on edgelets. However, their recognition performance were inferior on most object categories including handwritten digits to purely discriminative systems.

Recent progress in deep learning by Hinton *et al* [HOT06] has led to a series of works for learning unsupervised generative dictionaries [RHB07, LGR09, JKR09] using hierarchical convolutional networks. However, as discussed in [BBL10, CLN10], a lot of parameters needs to be carefully selected for the network to perform well. But neither the meaning of the network units nor their relationships to the parameters were intuitive enough to guide the tuning. By contrast, our approach opts for a very intuitive dictionary with a small set of easy-to-understand parameters.

Another notable approach to dictionary learning is reported by Mairal *et al* [MBP08], who trained supervised dictionaries for objects and texture classes. But their approach is not scalable since dictionaries of each class are trained separately, requiring additional training images and labels whenever a new class is added. On the contrary, our unsupervisedly trained dictionary enables features to be shared among different categories.

Previous theories on human's chunk learning behavior are mainly discussed in the relevant chapter in Section 4.1. Though the idea of compositionality has not played a role in this area before, it has long been advocated by vision researchers in related fields. For example, Geman *et al.* [BGP98] argued for its economical representation of shared parts between objects, and robustness to occlusion and deformation in visual cognition. Considerable success has been achieved by applying this principle to learn hierarchies of features and parts for the computer vision application of object recognition [ZCY07, EU05, FL07]. Compositions of shape fragments, interest points, and grayscale patches in these papers are good examples of grouping visual features into a hierarchy of chunks.

CHAPTER 2

A Dictionary of Redundant, Deformable Image Patches

In this chapter, we present a novel mid-level representation – the D-Patch dictionary, for the problem of simultaneous object recognition and reconstruction. We first give an overview of this representation framework in Section 2.1, highlighting its two key aspects: redundancy and transformation. The problem of inference, i.e. how to map an input image to an instance of this representation, is tackled in Section 2.2. Finally, we present methods to utilize the dictionary for object recognition and reconstruction in Section 2.3.

2.1 The D-Patch Representation

We choose to use the most basic image descriptor - patches in their raw intensity values - as features in our generative dictionary. They are simple, easy to interpret, and are suitable for both discriminative and generative tasks.

Note that features like HOG, SIFT, and shape context are based on histograms, and so are not well suited for reconstruction (despite their successes for recognition). Filter-banks of Gabor or wavelet functions are alternative descriptors, but have restricted forms and cannot adapt flexibly to different appearances.

The patches in the dictionary are designed to be flexible. They are rectangular areas cropped directly from any training images. They can be of any sizes larger than 5×5. They can come in any orientations, covering non-rectangular local

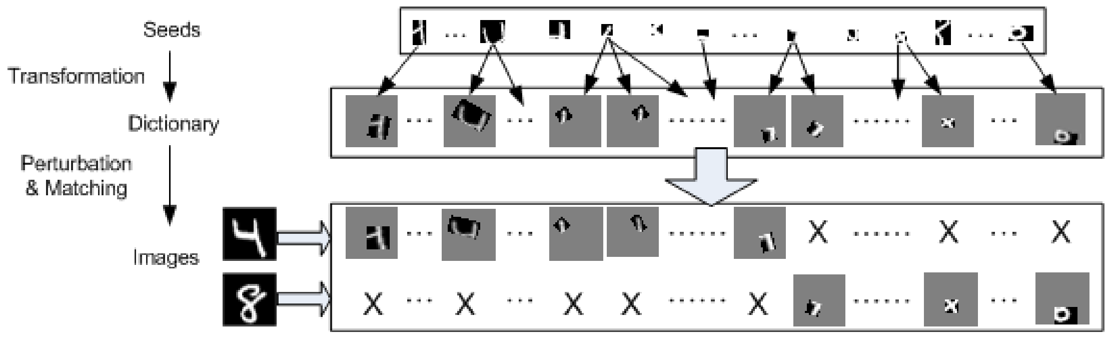

Figure 2.1: Illustration of the D-Patch representation. The dictionary of deformable patches are generated from a set of seeds plus transformations. Patches are perturbed and matched to local image regions.

image patterns. We describe a D-Patch P in terms of a seed patch S (always in upright orientation) and its transformation parameters T.

$$P = (S, T), \qquad T = (s_x, s_y, \theta, x, y) \tag{2.1}$$

2.1.1 Transformations

The transformations are controlled by five parameters $T = (s^x, s^y, \theta, x, y)$. (s^x, s^y) are the width and height of the patch in its upright form. θ is the rotation angle. And (x, y) specifies the image position of the rotated patch. See Figure 2.2 for a graphical illustration on how these 2D transformation is applied to a seed patch.

Table 2.1 summarizes all the supported transformation configurations. As a result of these pre-defined transformations, our dictionary is capable of covering a wide spectrum of local image patterns from smaller, generic ones (edgelets, strokes, corners) to larger, object-specific ones (T-junctions, X-junctions, rings). See Figure 2.1 for an illustration.

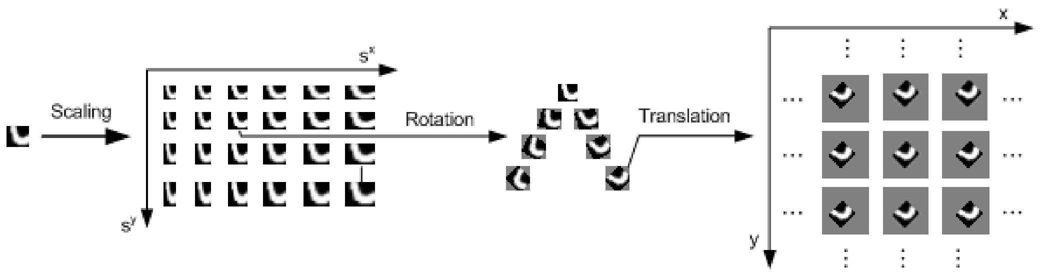

Figure 2.2: Applying transformations to a seed patch.

Table 2.1: List of transformation settings for a seed patch with original size (s_0^x, s_0^y) onto an image of size (s_I^x, s_I^y).

Type	Parameter	Min Value	Max Value	Stride
Scale	s^x	$\max(5, 0.5s_0^x)$	$\min(s_I^x, 2s_0^x)$	2 pixels
	s^y	$\max(5, 0.5s_0^y)$	$\min(s_I^y, 2s_0^y)$	2 pixels
Orientation	θ	$-45°$	$45°$	$15°$
Position	x	1	$s_I^x - s_0^x + 1$	1 pixel
	y	1	$s_I^y - s_0^y + 1$	1 pixel

2.1.2 Redundancy

Another crucial characteristics of this dictionary is its redundancy. We do not pursue to cluster semantically-equivalent structures into one dictionary element. Strokes and junctions can have many different actual configurations and our dictionary should cover all frequent instantiations by including all these different templates. The capability of neatly grouping these concepts is of secondary importance to us. This deliberation ensures the quality of pattern reconstruction from the dictionary without modeling the interactions between the dictionary elements.

Neither do we aim to segment image areas into non-overlapping regions, each explained by a different patch template. Such an approach is adopted widely in the image modeling works based on local basis functions. We elaborately keep the activated patches for one single image as redundant as possible. For example, a short stroke can simultaneously be part of a long stroke patch, part of a corner, and part of parallel strokes. All three larger components can contribute to the comprehension of the whole image. See Figure 2.3 for a visual illustration of this idea. Consequently we have plenty of features to aid the recognition task, explained in the last section of this chapter.

2.2 Matching Patches To An Image

To obtain a D-Patch representation of an image, we match the whole dictionary to the image and record a list of activated patch templates. The core decision here is the similarity measure between a patch and its corresponding image region. Following the example of Ullman *et al*, we use the normalized correlation between the patch P and the image I:

$$r(P, I) = \sum_{x \in P} \sum_{y \in I} \frac{(x - \bar{x})(y - \bar{y})}{\sigma_x \sigma_y} \in [-1, 1] \tag{2.2}$$

In order to make the matching robust to local variations, we allow the patch to perturb locally in terms of its transformation parameters. The best normalized correlation value among all perturbed versions is adopted as our actual matching score.

$$\tilde{P} = (S, T + \Delta_T) \qquad \Delta_T \leq |(2, 2, 15^\circ, 2, 2)| \tag{2.3}$$

$$\tilde{r}(I, P) = \max_{\tilde{P}} r(I, \tilde{P}) \tag{2.4}$$

We represent an image by the set of patches that *fire* on the image. A patch P_j fires on an image I_i if its matching score after perturbation is higher than a threshold.

However, a fixed threshold is not good enough for our flexible D-Patch dictionary, since we allow patches of different sizes. The normalized correlation value is by itself a rather inconsistent indicator of visual similarity across different vector sizes. Two small patches tend to yield a high correlation easily, while it is very difficult for two large patches to do so even when they are visually more similar.

To solve this problem, we introduce the t-statistics of the normalized correlation value. When we consider the correlation as a statistics computed from two samples, the correlation value follows the t-distribution of a freedom degree determined by the sample size N, which is exactly the size of the patch.

$$t = \tilde{r}\sqrt{\frac{N-2}{1-\tilde{r}^2}} \sim T(N-2) \tag{2.5}$$

$$p = 1 - tcdf(t, N-2) \tag{2.6}$$

After this calibration step, we can compute the activation of patches as follows using a fixed τ. Picking the threshold value is very easy because this is a widely-used p-value statistics. Normally people use 0.01 or 0.005 to indicate a significant trend, which in our case corresponds to a match between a patch and the input image.

$$F_{i,j} = \begin{cases} 1 & \text{if } p(I_i, P_j) > \tau \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

The firing states of all patches in the dictionary on an image forms a binary vector, which we use as features for training recognition models and classifying test images. Figure 2.3 shows the fired patches among our learned dictionary for two digit images.

2.3 Joint Recognition and Reconstruction

In this section we introduce ways to use the same D-Patch dictionary for both object recognition and reconstruction.

2.3.1 Object Classification

The recognition method is straightforward and standard in computer vision research. We treat each firing vector as the feature vector of the image. Then we feed these vectors to a standard classifier such as SVM or boosting trees.

In the case of multi-class classification ($M > 2$), we train C_M^2 one-versus-one classifiers, each trained from a binary classification sub-problem. For example, to classify 10 digits, we train 45 1-vs-1 classifiers. The final prediction is obtained from majority voting of these classifiers.

Throughout experiments reported in this thesis, we utilize the standard SVM-Light package [Joa] and employed two most basic kernels - linear and radial basis function. For linear kernels, the default parameter set by SVM-Light is used. For RBF kernel, we use a validation set of 10,000 images to choose the best values of a pair of parameters (γ, C).

2.3.2 Reconstruction

The reconstruction of an image from the dictionary is carried out on a per-pixel basis. We synthesize each pixel in the image from the corresponding pixel of the best matching patch in the dictionary that covers this pixel. This non-linear image model essentially generate each pixel from the patch in the dictionary that provides the closest match to its neighboring image area.

$$I(x) = P^*(x) \qquad P^*(x) = \underset{P_j : x \in P_j \wedge F_j = 1}{\operatorname{argmax}} p(I, P_j) \tag{2.8}$$

Another way to understand this reconstruction process is through the analogy of a piling up process. We sort all matched templates by their score from top to bottom, pile them all together, layer over layer, and finally look over the pile from the very top. What we see is the highest-scored patch, followed by the second-best patch excluding its overlapping area with the first, and so on.

Such a process is quite commonly used in the field of computer graphics. The phenomenal work on textual synthesis by Efros and Freeman [EF01] used a similar approach of merging overlapping patches with consistency constraint to get a full reconstruction of the desired texture.

2.4 Will A Random Dictionary Do The Trick?

To assess the effectiveness of our representational framework, we experiment with a baseline system where patches are sampled at random. No parameterized transformations or local perturbations are allowed. Elements in such a dictionary are not very flexible individually. The effectiveness relies solely on the redundancy of the random seed patches.

To obtain a set of seeds, we randomly extract 100 rectangular patches from each training image. For our handwritten digit dataset, the images are of size 28×28. We extract patches at random sizes from 7×7 up to 14×14 at all possible pixel locations. This dense multi-scale sampling strategy ensures the completeness of the seed set.

Then we carry out a few edge-based shape cleaning to the sampled patches. First we use the Canny edge detector to track edges in each patch. Patches whose longest edge chain is shorter than the width or height of the patch, and whose shortest edge chain is less than 3 pixels long, are discarded. Second, we crop out empty rows and columns on the patch boundaries, provided that the entire row or column is at least 2 pixels away from the nearest edge pixel. Other than these no further pruning or deliberate selection is applied.

Such a naive representation leads to surprisingly good digit recognition performances. The results are summarized in Table 2.2. In particular, a random dictionary sampled from $10,000$ images achieves much better performance than using $60,000$ raw images, which demonstrates the effectiveness of the patch representation framework. Furthermore, the patch dictionaries work very well with linear classifiers. This is an important achievement because linear classifier is much faster than K-NN or SVMs with polynomial kernels, and can scale to large amount of data fairly easily.

Table 2.2: Error rates of MNIST digit classification.

		Training Set Size		
Dictionary	Learning Method	1,000	10,000	60,000
Random Patches	Linear Classifier	7.73%	2.87%	
Learned Patches	Linear Classifier	2.73%	1.40%	0.70%
Raw Image	Linear Classifier			12.0%
Raw Image	K-NN Classfier			2.83%

The biggest problem with the random dictionary is that it scales badly with the size of the training set. Randomly-cropped patches are overly redundant when the training set gets big, wasting unnecessary storage and processing time. As a result, we were unable to apply this random dictionary to the complete MNIST training set. On the contrary, a learning process serves to discard less useful patches and maintain a more compact set of patch which is good enough for the task at hand. This will enable us to exploit the complete training set fairly quickly, reaping better recognition results than the baseline system.

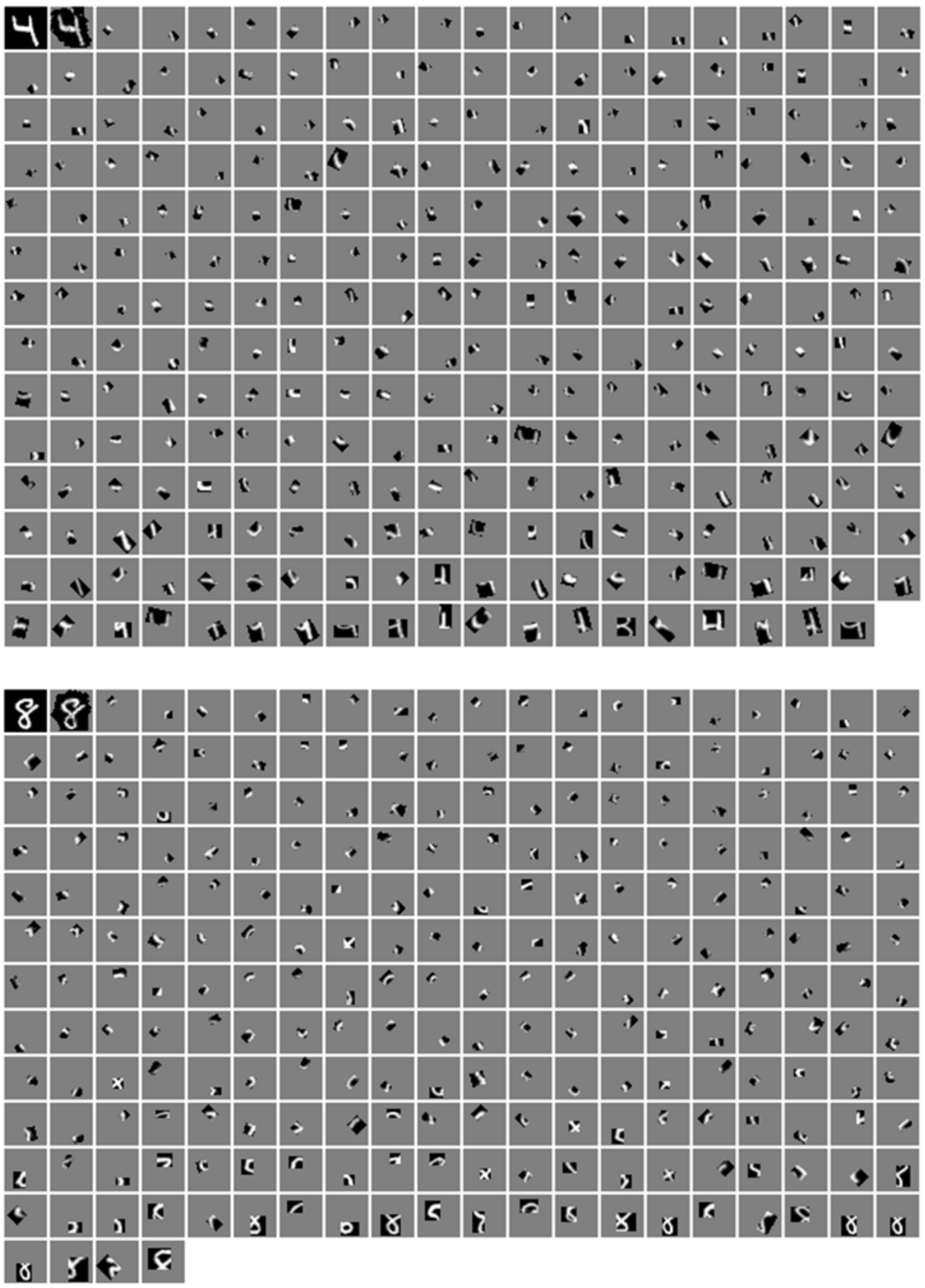

Figure 2.3: Two digit images from the MNIST dataset (top left panel) together with their reconstructed versions using our learned dictionary (adjacent panel) and a complete list of patches that are activated by respective images. The patches are sorted in descending order of their matching scores.

CHAPTER 3

Learning the D-Patch Dictionary

In this chapter, we present a method to learn a D-Patch dictionary given a set of training images. The images contain objects from different classes but we don't have the label information. The size of the dictionary is not specified either. Our learning algorithm is able to discover a reasonable set of image patches that can fulfill both the recognition and the reconstruction tasks outlined in the previous chapter.

3.1 The Learning Algorithm

The D-Patch dictionary are learned through four stages after the initial seed set is extracted from the training images, as described in the previous section. We describe them in detail below, along with an estimate of the complexities involved at each step.

3.1.1 Seed Clustering

First, a clustering algorithm is employed to reduce the redundant shape patterns. We used a density-based clustering method described in [EKS96]. The algorithm scans through the list of patches and assigns a patch to an existing cluster if the similarity between the patch and the center of the cluster is above a threshold. Here we used the same similarity measure and threshold presented in Eqn. 2.7. If no good match can be found, a new cluster centered around the current patch is created. Given the threshold, the number of clusters are determined automatically. This clustering algorithm has a complexity of $O(NM)$ where N and M are the number of patches before and after clustering.

The resulting cluster centers form our set of seed patches. The outcome may vary depending on different orderings of the patches being fed to the clustering algorithm. But in our experiments this has negligible effect on the final dictionary's performance in recognition and reconstruction.

We intentionally limit our seed candidate to be sampled from a relatively small set of training images, because the cost of dense-sampling and subsequent clustering is high. Therefore, the next stage aims to enrich this compact set of seeds so that we can have a set of patches flexible enough to cover most shape variations in the training images.

3.1.2 Applying Transformations

We have found that applying a set of transformations on each seed suits this purpose well. This is because a lot of the shape variations can be attributed to parameterizable transformations – scaling, rotation, and translation. After we factor them out, the remaining variations are relatively small and seem to have been captured by our seed set.

For each seed, we apply all possible transformations defined in Table 2.1, as illustrated by 2.2. There are roughly 10,000 transformation settings in total. Our last clustering step results in around 1,000 centers. Therefore we end up with around 10 million preliminary patches to select our final dictionary from. In order to evaluate them, we have to match them to all the training images, which is 60,000 for MNIST. A problem of this scale is out of the reach of normal computer hardware. Fortunately, an emerging generation of massively parallel Graphical Processors (GPUs) makes such computation possible.

3.1.3 Parallel Matching using GPUs

The matching operation between a patch and an image, as described in Section 2.2, needs to be performed billions of times during the training of D-Patch dictionary. However, a lot of these operations is parallelizable on GPUs. We make use of two key observations on this problem. First, the result of all these matching operations are independent of each other, meaning the matching can be computed by separate hardware units. Second, many operations, if carried out at the same time, access the same piece of data. In particular, accesses to image pixels by neighboring matching operations exhibit a coalesced pattern, which is ideal for the GPU architecture.

To match all transformed copies of a seed to the entire set of training images, we first stitch together all training images as one giant one. Then scaling and rotation is performed on the seed. Finally, we match each of these semi-transformed patches to all pixels of the stitched image simultaneously on parallel cores. Under this strategy, all matching operations between the same scaled and rotated patch to every possible image regions in the training set are parallelized. Tens of millions of matching operations can be performed at the same time.

On the memory access front, all the parallelized operations read from the same semi-transformed patch, which is a small matrix and can be stored in constant memory for fastest access. Matching operations centered on neighboring pixels always read neighboring image values and write back to neighboring output addresses, which enable coalesced global memory access almost everywhere. Pre-loading data into shared memory may bring further speedup but we have not adopted this strategy yet.

Using a single 240-core Tesla C1060 GPU card, we are able to reduce the time for computing normalized correlations between 10 million preliminary patches and 60,000 training images to about 4 hours. The same parallel matching strategy is used for classifying a new image using a learned dictionary of about 10,000 patches. And we are able to process more than 100 testing images in parallel in less than a second, achieving real-time performance.

3.1.4 Feature Selection

After matching all preliminary bases to the images and obtaining their firing statistics, we then select the final D-Patch dictionary elements in a greedy manner. The selection is mainly based on the firing frequency. First, non-maximal suppression is applied to firings between the same patch and multiple neighboring positions of the same image to remove duplicate counts. Then, we sort the preliminary patches from the most to the least frequent, and select them in this order. Every time a patch is selected, we suppress similar versions of it by eliminating patches generated from the same seed under similar transformation settings (the suppression range is twice the perturbation range described by Eqn. 2.3). The selection stops when the firing frequency falls below a threshold. We set the threshold to be 1% in this paper.

We list in Table 3.1 the number of patches we are dealing with at each stage of the learning process. Though the transformation stage significantly increased the number of patches we consider, only about 0.1% of them were picked in the final selection stage. From another perspective, on average 10 transformation settings of each seed were accepted as useful patches.

3.2 The Learned Dictionaries

In this section we present qualitative properties of the learned dictionaries. In particular, we evaluate the quality of the learned dictionary from four aspects – intuitiveness, completeness, sparseness, and discriminative ability.

First, we show in Figure 3.1 a subset of the learned dictionary. The dictionary

Table 3.1: Statistics of the learned dictionaries.

Dataset	MNIST			USPS
Training Size	1,000	10,000	60,000	7,291
Seeds	1,140			603
Prelim Patches	15,427,836			9,685,896
Learned Patches	10,264	9,714	9,727	6,343
Firing Freq. (Train)	5.08%	5.23%	5.20%	8.03%
Firing Freq. (Test)	5.15%	5.36%	5.32%	8.30%

elements are very intuitive. Our seeds set captures a wide range of structures, from elementary ones (edgelets, corners) and complex ones (T-junctions, X-junctions, rings). Most seeds contain structures that can be shared between multiple digits, while some are more digit-specific. The D-Patch dictionary successfully captures the frequent modes of transformations. Interestingly, different transformations of a seed patch may correspond to different digits. For example, a ring pattern could become part of 2, 6, 8 and 9 when appearing at different locations.

For completeness, we show a collection of synthesized images using the learned dictionary in the left half of Figure 3.2. The synthesized images are obtained by the method described at the end of Section 2.3. We observe that our dictionary is capable of covering all possible shape variations, with no details left behind, although we do not explicitly minimize reconstructive errors in our learning algorithm.

Our learning algorithm does not impose sparsity either, but sparseness of the firing vectors comes out naturally after training. Table 3.1 shows around 5% of the learned patches fire on average for an MNIST image. For USPS, the proportion is 8%, a bit higher but still quite sparse. This sparsity behavior can be mainly attributed to the many digit-specific mid-level structures contained in the dictionary for 10 different digits.

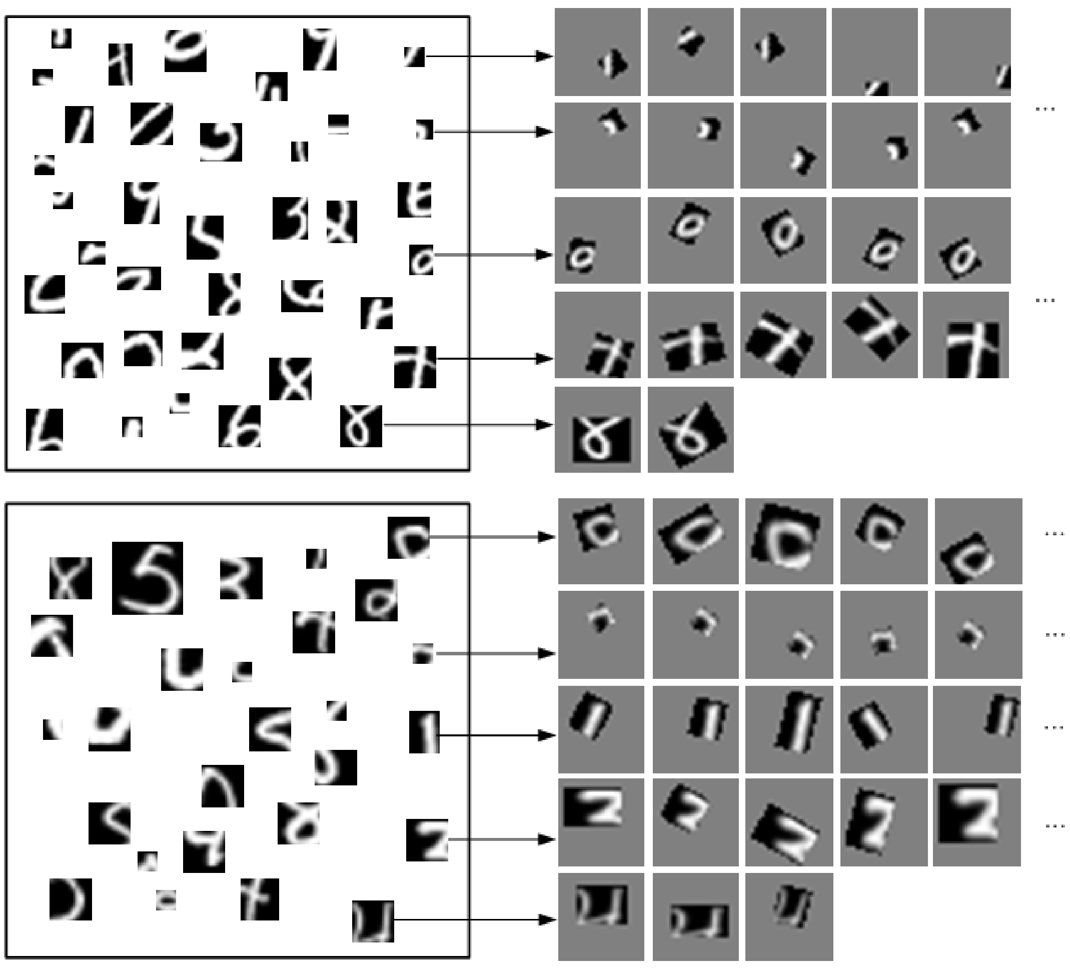

Figure 3.1: Examples of learned seeds and patches. The left half of the figure shows seeds learned from MNIST, as well as the most frequent (up to 5) bases from some of the seeds. The right half shows shows seeds and patches from USPS.

Finally, we demonstrate the discriminative power of the patches in Figure 3.3. For each D-Patch, we visualize its firing rates on the 10 digit categories in an activation histogram, for training and testing images respectively. We find that some of the learned patches are very good weak classifiers. For example, an X-junction near the center of the image is a strong indicator for an 8, a weak indicator for a 2, and a negative indicator for other digits. It is the presence of many patches like this that leads to impressive digit recognition performance, which is reported in the next subsection.

3.3 State-of-Art Digit Recognition and Reconstruction

We achieved state-of-art handwritten digit recognition results on both highly competitive benchmark datasets, the MNIST and USPS handwritten digits. Both sets have been extensively studied in the past and near-human level performance have been reported already. We are able to match the state-of-art performance with a much simplified representation and learning framework.

For the purpose of a fair and easy-to-reproduce comparison, we use the standard SVM-Light package [Joa] and employed two most basic kernels - linear and radial basis function for our recognition tasks. For linear kernels, the default parameter set by SVM-Light is used. For RBF kernel, we use a validation set of 10,000 images to choose the best values of a pair of parameters (γ, C).

Note that the above choice of classifiers is much less elaborate than previous methods. Most deep learning works employ a specially-trained Support Vector Machine with 5 to 9 degree polynomial kernel. Such training takes too long to complete and have a high tendency towards over-fitting.

Table 3.2 shows that our classification error rates are better than the best competing methods from supervised dictionary learning [MBP08], which also reported performance on both MNIST and USPS. Our results are also better than the deep learning based method when training on a smaller set of images [LGR09]. Our recognition performance is very close to the individual state-of-art on either MNIST [JKR09] or USPS [HK02]. And we perform better than the classic shape context feature [BMP02] without having to store all training images in testing.

Our approach delivers good performance even with simple linear classifiers. This is important because linear classifiers are very fast to train and therefore can be applied to large datasets easily. This merit is usually downplayed in the academia but is a must-have quality for developing exciting vision products in the industry.

Table 3.2: Comparison of state-of-art digit recognition methods on benchmark datasets.

Dataset	MNIST			USPS
Training Size	1,000	10,000	60,000	7,291
Testing Size	10,000			2,007
D-Patch Dictionary + Linear SVM	2.73%	1.40%	0.70%	3.04%
D-Patch Dictionary + RBF SVM	2.56%	1.26%	0.60%	2.84%
Shape Context + kNN [BMP02]			0.62%	
Supervised Dictionary [MBP08]			1.05%	3.54%
Deep Belief Network [LGR09]	2.62%		0.82%	
Convolution Network [JKR09]			0.52%	
USPS-specific Kernel Learning [HK02]				2.40%

3.4 Dictionary Transfer

Finally we demonstrate the transfer capability of our dictionaries. Few prior works have addressed the issue of using features learned on one dataset to encode another dataset, as generality of the learned dictionary are usually considered secondary to classification performance. However, we feel that a useful dictionary must be general enough to transfer its discriminative ability across datasets. We demonstrated such ability in our learned dictionaries.

We tested two different transfer settings. One is to transfer the complete D-Patch dictionary trained on one digit database to the other. The other setting transfers the seed candidate set only and learns a new D-Patch dictionary on the target dataset. Results are summarized in Table 3.3 for recognition and Fig 3.4 for reconstruction.

In general, transferred dictionaries work effectively for our applications. Seeds appear to be a better choice to transfer since they encode invariant structures,

Table 3.3: Recognition using dictionaries learned on another dataset. Results are compared between transferring both seeds and patches, transferring seeds only, and no transfer at all.

Seeds	Bases	Test	Linear SVM	RBF SVM
MNIST	MNIST	USPS	3.39%	3.14%
MNIST	USPS		3.03%	2.89%
USPS	USPS		3.04%	2.84%
USPS	USPS	MNIST	0.96%	0.87%
USPS	MNIST		0.78%	0.66%
MNIST	MNIST		0.70%	0.60%

while patches can be affected by different digit writing styles in different datasets. For instance, USPS digits are often wider and thicker than MNIST counterparts. In addition, seeds learned from MNIST seems to encompass richer structures than seeds from USPS and gives better results, probably due to a more comprehensive training set.

3.5 Conclusions

This chapter presents a method for learning deformable dictionaries of image patches for representing shapes. Our learning algorithm imposes a large set of transformation on a randomly sampled set of seed patches to enrich the search space of dictionary elements. The evaluation of this space is made possible by using Graphical Processors and designing a massively-parallel template matching framework. We evaluated our approach quantitatively on the MNIST and USPS databases and obtained results comparable to the state of the art. We also demonstrated the possibility of transferring the learned dictionaries from one dataset to the other.

The key insight of the learning procedure is that the emergence of parallel computing hardware such as GPUs enables the aggregation of certain computation (deformable template matching by normalized correlation in our case) on a unprecedented scale. We are then able to opt for simpler techniques (intensity patches as features, greedy feature selection based on frequency, and linear classifier) without losing performance as compared to more complex counterparts. Besides, we get much faster speed and more intuitive intermediate representations at each step of the whole system.

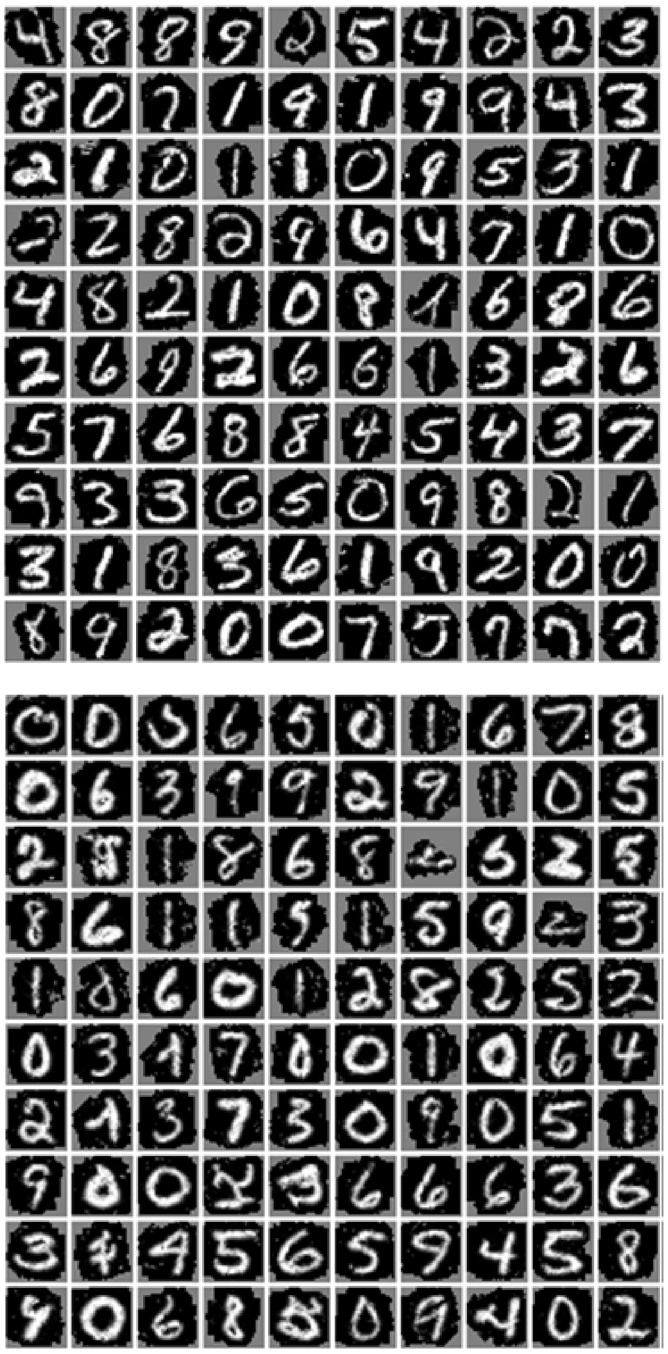

Figure 3.2: Synthesized images using the learned and transferred D-Patch dictionaries. The top half shows 100 reconstructed MNIST images synthesized from a dictionary learned on the same dataset. The bottom half shows synthesized USPS images by a dictionary learned on USPS.

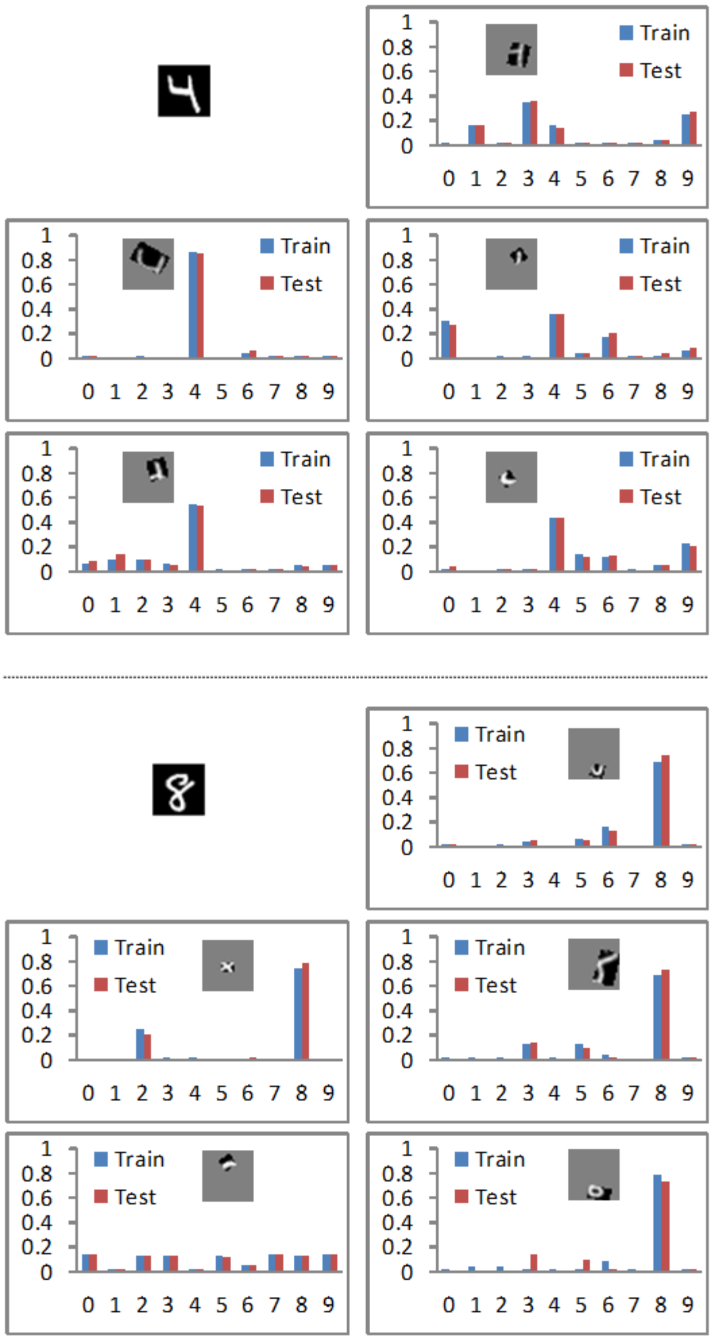

Figure 3.3: Using learned bases as features for recognition. For two MNIST images, the activation histograms are shown for some of the fired bases. Observe that some bases are very discriminative, e.g. U-shape for 4, X-junction for 8, etc.

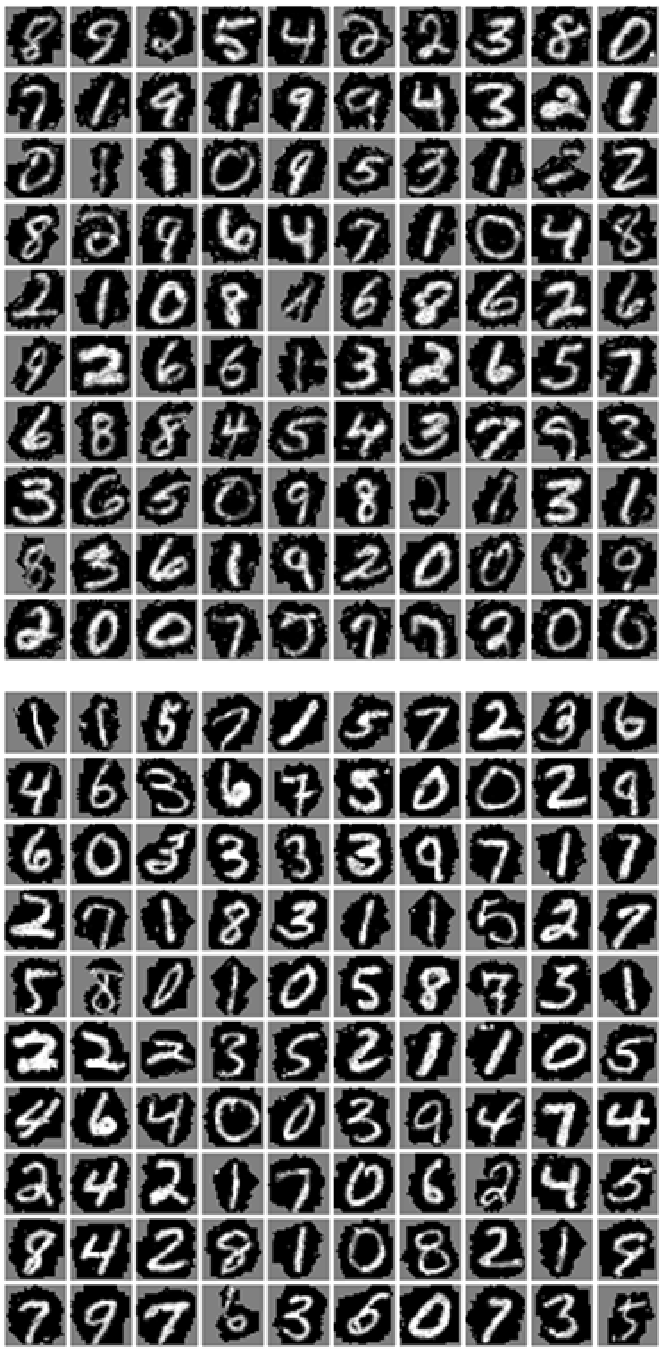

Figure 3.4: Synthesized images using transferred D-Patch dictionaries. The top half shows 100 reconstructed MNIST images synthesized from a dictionary learned on USPS. The bottom half shows the other way around.

CHAPTER 4

Modeling Human Visual Chunk Learning Behavior

Though the patch descriptors introduced in the previous chapters are very effective in various tasks, it is lacking several key ingredients for representing real-world objects. Most visual objects are compositional, and many exhibit flexible spatial configurations. Therefore a well-rounded object representation must be able to deal with hierarchical compositions of local descriptors. In this chapter, we provide a solution for this problem and show that it accounts for human compositional learning behavior under a novel experiment.

The chapter is organized as follows. We first summarize existing theories on the topic of human compositional learning in Section 4.1. Among them, we pick out the latest state-of-art model, the Bayesian Chunk Learner, and provide a detailed analysis in Section 4.2. Then we present our new model, the Hierarchical Chunk Learner in Section 4.3, followed by a novel inference algorithm in Section 4.3.2 and a learning procedure in Section 4.4, highlighting the reduction of computational complexity. In Section 4.6, we report a new psychophysical experiment, where human observers learned compositional concepts that are not supported by BCL but our model.

4.1 Background

Ever since Helmholtz [Hel25] characterized perception as a process of unconscious inference, a fundamental question in perception research has been how the visual system forms efficient representations of complex scenes, allowing seemingly effortless inferences to be made. The classic study by Miller [Mil56] on human short-term memory demonstrated that *chunks* can be formed in order to increase the number of items that the human mind can code and represent at one time.

Later research showed that, as early as 14 months of age, human infants are able to bind individuals into sets and thereby increase their representational capacity [FH03]. Furthermore, Fiser and Aslin [FA01] reported that after a brief period of passive viewing of complex visual scenes, human adults can readily encode and remember shape conjunctions, even though observers were not instructed to attend to any particular features of the displays. More generally, the ability to form chunks from complex inputs is a hallmark of human perception and cognition.

However, the mechanisms by which a human learner can form chunks remain poorly understood. Two general explanations have been offered in the literature. One is that high-level schema processing may provide top-down information, which is combined with low-level information to form chunks [CS73]. Although this is an appealing proposal, the origin of the high-level schema knowledge remains unclear (i.e., whether the schemas are innate or acquired via some learning mechanisms). The other proposal is that humans rely on bottom-up statistical learning based on repeated exposures to inputs and detecting suspicious coincidences [ZCY07] of elements during learning, thereby extracting chunks from low-level information [GLC01] via a bootstrapping approach.

4.2 The Bayesian Chunk Learner (BCL)

Recent work by Orban *et al.* [OFA08] provides support for the second mechanism of statistical learning. They found that a Bayesian model based on extracting chunks from complex visual patterns accounted for human performance much better than did a simple associative learning model that encodes the full correlational structure of the visual inputs.

Although this Bayesian chunk learner model (denoted BCL) advanced our understanding of the underlying principles in visual chunk learning, two critical issues prevent this model from being a general computational framework for human chunk learning. Below we elaborate on these limitations and present our approach to overcome these issues.

The first limitation stems from BCL's representation of visual chunks as a single layer of hidden units that never interact with each other. As a consequence, the appearance of any two chunks constitute independent events. However, it is apparently advantageous for the human visual system to represent complete objects and their parts as separate but interacting chunks.

This idea, termed the compositionality principle, has long been advocated by vision researchers in related fields. For example, Geman *et al.* argued for the economical representation of shared parts between objects, which can yield robustness to occlusion and deformation in visual cognition [BGP98]. Considerable success has been achieved in computer vision by applying this principle to learn hierarchies of features and parts for the application of object recognition [ZCY07, EU05, FL07].

Accordingly, we propose using hierarchical models to represent visual chunks. The hierarchical structure allows large chunks to be composed of other smaller ones, leading to more accurate probability distributions for modeling part-based objects. In a new psychophysical experiment reported in this paper, we show that human performance agrees with the prediction of our hierarchical approach for the task of picking up an untrained visual chunk that is part of a larger trained chunk. Such an effect could not be captured by BCL, which predicts chance performance on the untrained chunk.

The second limitation of BCL is the efficiency of their algorithm. BCL adopted a fully Bayesian computation using MCMC sampling over the entire model space. Given the high dimensionality of visual scenes, the algorithm typically takes several days of computation. We demonstrate in this paper that the maximum-likelihood (denoted ML) estimates of model states, parameters, and structure are sufficient to account for human performance. The proposed ML method can estimate model states via an efficient inference algorithm, and assess structure by a data-driven composition and validation procedure based on a series of likelihood ratio tests. As a result, we are able to dramatically reduce model training time to less than one minute.

4.3 The Hierarchical Chunk Learner

In this thesis, we aim to model a set of scenes where a number of elementary shapes appear over a range of discrete locations. The presence of shapes is determined by a hidden inventory of visual chunks. Knowledge about these hidden chunks can only be inferred from a set of training scenes without explicit supervision. This setting is standard in related literature.

4.3.1 Model Formulation

We develop a probabilistic model to explain the scenes using a hierarchy of visual chunks (see Figure 4.1). The leaf nodes in the model correspond to directly observed elementary shapes. The hidden nodes encode the configurations of visual chunks. One critical difference between this model and BCL is that we allow hidden nodes to be parents of other hidden nodes.

$$P(x, u|\theta_S, S) = \prod_i P(x_i, u_i|x_{Par(i)}, u_{Par(i)}, \theta_S, S) \tag{4.1}$$

$$\begin{aligned} P(x_i, u_i|x_{Par(i)}, u_{Par(i)}, \theta_S, S) = \text{Bernoulli}(x_i; \text{Sigmoid}(w_i + x_j w_{ij}))\cdot \\ \left(\frac{1}{Z}\text{DNormal}(u_i; c_i, \sigma_1 \cdot I)(\text{DNormal}(u_i; u_j + c_{ij}, \sigma_2 \cdot I))^{x_j}\right)^{x_i} \\ j = \operatorname*{argmax}_{j \in Par(i)} P(x_i, u_i|\theta_S, S) \end{aligned} \tag{4.2}$$

Each node i in our hierarchical model, whether observed or hidden, can be present $x_i = 1$ with a discrete 2D position u_i, or absent $x_i = 0$, in a scene. Given a configuration of model structure S and parameters θ_S, the probability of a joint model state of all nodes (x, u) is given by Eqn. 4.1 as the product of individual nodes' conditional probabilities on their parents (Eqn. 4.2).

w_i gives the spontaneous appearance weight of node i and $(c_i, \sigma_1 \cdot I)$ gives its prior spatial distribution. When a parent is present ($x_j = 1$), w_{ij} quantifies its influence on the appearance probability of x_i, and $(c_{ij}, \sigma_2 \cdot I)$ denotes the influence of parent position u_j on u_i. DNormal$(c, \sigma \cdot I)$ is a 2D Gaussian kernel defined on discrete positions. Z is a suitable normalizing factor for the spatial term. We describe how to estimate these parameters from data in Section 4.4.

The status of a single node can be affected by multiple parent nodes. Since our model may have more than two layers of nodes, this setting leads to difficulty in inference due to possible loops in the model structure [Pea88]. Learning also becomes more challenging due to the explaining-away effect caused by the existence of loops [HOT06]. Therefore we restrict each node to have at most one active parent, reducing the graph structure to a tree. The parent is chosen at inference time to maximize the marginal probability of the child node.

4.3.2 Maximum-Likelihood Based Inference

In both training and testing, we need to find out the maximum-likelihood state of the model under a given scene. We use (y, v) for the states of the observed nodes and (x, u) for those of the hidden nodes. The ML states of the hidden nodes can be written as:

$$(x^*, u^*) = \underset{x,u}{\operatorname{argmax}} P(y, v|x, u, \theta_S, S)P(x, u|\theta_S, S) \tag{4.3}$$

We search for (x^*, u^*) using a "node-splitting" technique proposed by Choi *et al.* [3], which is a variation of belief propagation in its max-product version. Figure 4.2 gives an illustration of the inference process. In the first bottom-up pass, we propagate messages from a child node to all of its parents in parallel as if each of them is the only parent. Then in the first top-down pass, we obtain a set of marginal distributions for each node (computed by Eqn 4.2). We select the parent and the node state that provides the maximum marginal probability. After this step, the graph is reduced to a set of trees. A second round of belief propagation is performed on the reduced model. This second round is necessary because the ML states of some nodes might change when they become disconnected from their children.

4.4 Learning The Hierarchical Chunk Model

Given a set of training scenes $D = (y^k, v^k), k = 1, ..., N$, we aim to learn the single best model structure $S*$ and its associated parameters θ_S^* under the maximum likelihood principle.

$$(\theta_S^*, S^*) = \underset{\theta_S,S}{\operatorname{argmax}} P(D|\theta_S, S) = \underset{\theta_S,S}{\operatorname{argmax}} \prod_k P(y^k, v^k|\theta_S, S) \tag{4.4}$$

The ML solution is much easier to obtain than the entire posterior distribution $P(\theta_S, S|D)$ required in a Bayesian framework. In Section 4.5.3, we show that this solution is capable of correctly predicting human chunk learning performance in previous experiments.

We learn the hierarchical model in two stages, as illustrated by Figure 4.3. In the model composition stage, we adopt a data-driven approach to search for possible visual chunks in a highly probable subspace. We compose elementary shapes into visual chunks in a progressive manner, each time trying to combine two shapes or chunks together. If a combination appears frequently, we propose the formation of a new visual chunk. The proposal is validated by a likelihood ratio test (Eqn. 4.5) between this single-chunk model and a default model with independent shapes. Accepted proposals are subject to further composition and later model selection.

$$\frac{P(D|\theta_S^1, S^1)}{P(D|\theta_S^2, S^2)} \approx \frac{\max_{x,u} P(D|x, u, \theta_S^1, S^1)P(x, u|\theta_S^1, S^1)}{\max_{x,u} P(D|x, u, \theta_S^2, S^2)P(x, u|\theta_S^2, S^2)} \tag{4.5}$$

Note that Eqn. 4.5 is using the likelihood under the ML model states as an approximation of the full likelihood. This saves the learning algorithm from having to sum over all hidden states (x, u). Instead, we are able to utilize the efficient inference procedure for (x^*, u^*), as described in section 4.3.2.

In the model selection stage, we start with the default model, and at each step greedily add to the model a visual chunk that maximizes the likelihood of training images. Edges are added between chunks whose shape combinations are a subset or a superset of each other. The iteration stops when no additional visual chunks can improve the likelihood. The bottom half of Figure 4.3 illustrates the model selection process.

Whenever a new visual chunk is added to the model, we need to estimate the parameters associated with it. Among the six types of parameters $\theta =$

$\{w_i, w_{ij}, c_i, c_{ij}, \sigma_1, \sigma_2\}$, spatial variance σ_1 and σ_2 are hand-specified and fixed in all experiments. c_i's are fixed at the center of each scene. c_{ij}'s are estimated as the mean values of all observed relative positions in scenes where both child node i and parent node j are present. w_i's and w_{ij}'s are estimated from empirical statistics with minimum smoothing, as given by the following equations:

$$w_i = \text{Sigmoid}^{-1}\left(\frac{\#\text{Scene}_{i\wedge\neg Par(i)} + 1}{\#\text{Scene}_{\neg Par(i)} + 2}\right) \qquad w_{ij} = \text{Sigmoid}^{-1}\left(\frac{\#\text{Scene}_{i\wedge j} + 1}{\#\text{Scene}_{j} + 2}\right) - w_i \tag{4.6}$$

4.5 Predicting Human Chunk Learning Behavior

4.5.1 Methods

After the model is learned, we need a way to assess how well the model simulates actual human chunk learning behaviors. Since the standard way to measure human learning performance in the literature is to ask human observers to choose the more familiar chunk between a trained one T_1 and a random distractor T_2, our model simulates this choice by computing the probability of picking the trained visual chunk as:

$$P(\text{choose } T_1) = \text{Sigmoid}\left(\beta \log \frac{P(y^{T_1}, v^{T_1} | \theta_S^*, S^*)}{P(y^{T_2}, v^{T_2} | \theta_S^*, S^*)}\right) \tag{4.7}$$

The deciding factor in Eqn. 4.7 is the likelihood ratio between the two test scenes. Note that we use the likelihood of ML states under the ML model structure and parameters to approximate the full likelihood. Following the setting of BCL, β is a parameter used to fit human performance.

4.5.2 Psychophysical Experiment Settings

Previous psychophysical experiments [FA01, FA05, OFA08] have investigated how humans learn structures of visual chunks via passive viewing, a paradigm known as visual statistical learning. The top panel in Figure 4.4 shows the general procedure used in these studies (adopted from [FA01]). First, an inventory of visual chunks was generated by randomly grouping shapes together. Each chunk defined a fixed spatial relationship among its constituent shapes. Then, training scenes were generated by randomly placing chunks next to each other without overlaps. Participants passively viewed these scenes during the training session. Lastly, learning performance was measured in a testing session consisting of several trials. In each trial, the participant was presented with one true chunk that was taken from the inventory, and one false chunk (distractor) that was generated by randomly putting shapes together in the same spatial layout as the true chunk. Participants were asked to judge which chunk looked more familiar to them. Learning performance was taken as the proportion of trials in which participants chose the true chunk.

An interesting finding from these experiments is that, while humans were able to learn larger, complex chunks, they were unable to learn ”sub-chunks” that were embedded within the learned complex chunks. For example, the middle panel of Figure 4.4 shows an experiment designed by Fiser and Aslin [FA05], and later repeated by Orban *et al.* [OFA08]. Their inventory contained two 4-shape chunks (quadruple) and two 2-shape chunks (pairs). Each training scene contained one quadruple and one pair adjacent to each other. They found that learning performance for the quadruples and pairs inside the inventory was significantly better than chance. On the contrary, for pairs that were embedded within quadruples, observers showed chance level performance despite the pure frequency of displaying the embedded pairs were the same as chunks in the inventory.

A similar pattern of result was found in a later experiment in [OFA08], as shown in the bottom panel of Figure 4.4. The inventory included four 3-shape chunks (triplets), two pairs, and a square quadruple. Each training scene consisted of either the quadruple and a pair, or a triplet plus a pair plus a single shape from the quadruple. Their results showed that human learning performance of the true triplets was better than that of a triplet embedded within the quadruple. Recognition performance of the latter was not significantly different from chance level.

4.5.3 Results on Previously Reported Experiments

We include our model predictions for the above experiments in Figure 4.4, side-by-side with the predictions by BCL. As shown in the figure, our model achieved comparable accuracies to BCL in predicting human learning performance for both experiments. All predictions by our model were made using the same β value. Note that although we advocate the modeling of part-to-whole relations, our learning algorithm will not pick up embedded subparts if they are not distinctive themselves.

At the same time, our model takes less than one minute to train and less than one second to test for each of these experiments, whereas BCL needs several days to obtain the millions of MCMC samples required for approximating the posterior distribution of model configurations. These results demonstrate the significant improvement in efficiency achieved by our learning algorithm.

4.6 A New Experiment on Human Learning of Hierarchical Visual Chunks

4.6.1 Motivations

Experiments discussed in the last section indicate that an embedded part within a larger configuration are not encoded as explicit components in internal representations. If this were the case, a flat representation with disconnected hidden units, which was used by BCL, would be adequate. However, as suggested by the compositionality principle, explicitly encoding parts provides a more robust and economical representation for common visual perception tasks. For example, face, arms, and legs are all parts of the human body, but due to frequent occlusions, these parts are not always visible together in a scene. Yet humans are able to recognize them both separately and as a whole. For many part-based objects such as the human body, it is more natural to model their various visual chunks using a hierarchical structure.

One possible reason why previous experiment results failed to capture the importance of encoding visual parts might be that observers never viewed an embedded part as a stand-alone visual chunk during training. As a consequence, the gain in data explanation ability from forming sub-chunks did not compensate for the increase in model complexity, leading human observers to favor a flat representation. To test this hypothesis, we designed a new experiment in which a shape combination could appear both as part of a large configuration and as a stand-alone chunk. If human observers are able to form explicit representation of subparts, we expect that they could readily recognize the trained embedded shape combination, and, more importantly, induce the complementary shape combination as a part, despite the fact that the latter combination was never displayed alone during training.

4.6.2 Experiment Design

The basic settings of the new experiment followed those discussed in the last section. 20 observers in total participated in the experiment. We used the same 12 elementary shapes and randomized the assignment of shapes to chunks for each human observer. Without loss of generality, letters from A to L are used here to represent these shapes for easier reference. As shown in the top panel of Figure 4.5, our inventory included two quadruples (ABCD and EFGH) and two pairs (IJ and KL). We then introduced the critical manipulation: for each observer, one of the quadruples (e.g., EFGH) was chosen to be the *target quadruple*, and an embedded pair within it (e.g., EG) was chosen to be the *trained embedded pair* (denoted TEP). Choices of both target quadruple and TEP were counterbalanced across observers. This design of the inventory would allow us to test whether recognition of the complementary pair (FH) embedded in the target quadruple would be better recognized than that of the untrained embedded pairs (AC or BD) in the non-target quadruple.

Training was carried out in two phases. Phase 1 aimed to let observers form a solid representation of TEP before seeing the quadruple that enclosed it. Observers first viewed the TEP for 30s. Then, they were asked to detect the TEP in 96 scenes (48 distinct scenes with 2 repetitions, 2 seconds per scene). Each scene consisted of 4 adjacent shapes, which may or may not include the TEP. Afterwards, observers were given a facilitation test block consisting of 12 trials, with 4 trials for each of the three true pairs in the inventory, i.e. the TEP and the two stand-alone pairs (IJ and KL). In each trial they judged whether the true pair or a randomly generated false pair looked more familiar. Observers were 92% accurate on the TEP detection task, and 98% accurate on the familiarity judgment for TEP, indicating successful formation of representation for TEP after phase 1.

In phase 2, observers passively viewed 104 distinct training scenes, one after another, without doing any task. Each training scene consisted of one quadruple (ABCD or EFGH) and one pair (EG, IJ or KL). As all shapes in a scene must be distinct, there were five possible types of training scenes: ABCD+IJ, ABCD+KL, ABCD+EG, EFGH+IJ and EFGH+KL, with respective frequencies of 0.125, 0.125, 0.25, 0.25, and 0.25. This setting ensured that the frequencies for all the untrained embedded pairs (FH, AC, and BD) were equal, allowing us to have a fair comparison among their recognition performance. The final testing session consisted of 12 trials, with 2 trials for each of the following 6 conditions: Target quadruple (EFGH), Non-target quadruple (ABCD), Pair (IJ or KL), Complimentary pair embedded in the target quadruple (FH), Untrained embedded pair in the non-target quadruple (AC or BD), and the Trained embedded pair (EG). See Figure 4.5 for visual illustrations on these training and testing scenes.

4.6.3 Experiment Results

Human performance for the TEP (EG) was rather good as expected (0.95), indicating that learning of this substructure in phase 1 was retained in phase 2. For the other five test conditions, human performance is depicted in Figure 4.5. Observers recognized the non-target quadruple (ABCD) much more accurately than identification of its embedded pair (AC/BD) (accuracy 0.75 versus 0.55), replicating the previous results reported by Fiser and Aslin [FA05]. However for the target quadruple (EFGH), after learning the TEP (EG), observers were able to recognize the untrained complementary embedded pair (FH) with a better-than-chance performance level of 0.68, despite the fact that the complementary pair had never been displayed alone (just like AC/BD). In other words, observers induced the complementary pair as a component of a larger structure after acquiring knowledge of the trained embedded pair. This result supports the existence of a hierarchical representation for the target quadruple; i.e., embedded pairs were explicitly represented as a component of the target quadruple.

Given such training sequences in our new experiment, BCL preferred a family of models containing two independent visual chunks for the target quadruple (EFGH) and the TEP (EG). None of the untrained embedded pairs (FH, AC, and BD) was picked up as an explicit chunk. Therefore, FH was assigned the same probability as AC/BD and any other random pairings of shapes. As a result, the model predicted similar performance on all these untrained pairs at chance level.

Our hierarchical chunk model, on the other hand, placed EFGH as the parent of EG, which in effect forms a weak representation of the complimentary pair FH. The resulting hierarchical model assigned more probability to FH compared to AC/BD, and increased the recognition performance of FH to above chance level.

The human performance in this experiment clearly agreed with the prediction of the hierarchical model. As discussed in the last subsection, the recognition accuracy of FH was significantly higher than that of AC/BD and chance level, but not as good as specifically trained EG and other stand-alone visual chunks. Apart from FH, human performance on other tested visual chunks roughly agreed with the predictions of our hierarchical model. Overall, the experiment and simulation results demonstrated that our hierarchical chunk model is a better fit for modeling human chunk learning behavior than previous flat models, especially in complex scenarios where one visual chunk might be a component of other large configurations.

4.7 Conclusions

In this chapter, we have presented a novel hierarchical approach for modeling human's visual chunk learning behavior. The framework includes a hierarchical chunk model, a fast inference algorithm, and an unsupervised learning algorithm. There are two main novelties. First, we utilized the hierarchical model structure to capture part-to-whole relations between visual chunks. Second, we adopt a data-driven maximum-likelihood approach to learn this model, reducing running time from several days to less than one minute. We reported a new experiment where chunks and their components were trained at the same time. Human observers were able to learn a special type of untrained embedded parts under this scenario. This result is in accordance with our model's prediction but not BCL's, due to the latter's flat model structure.

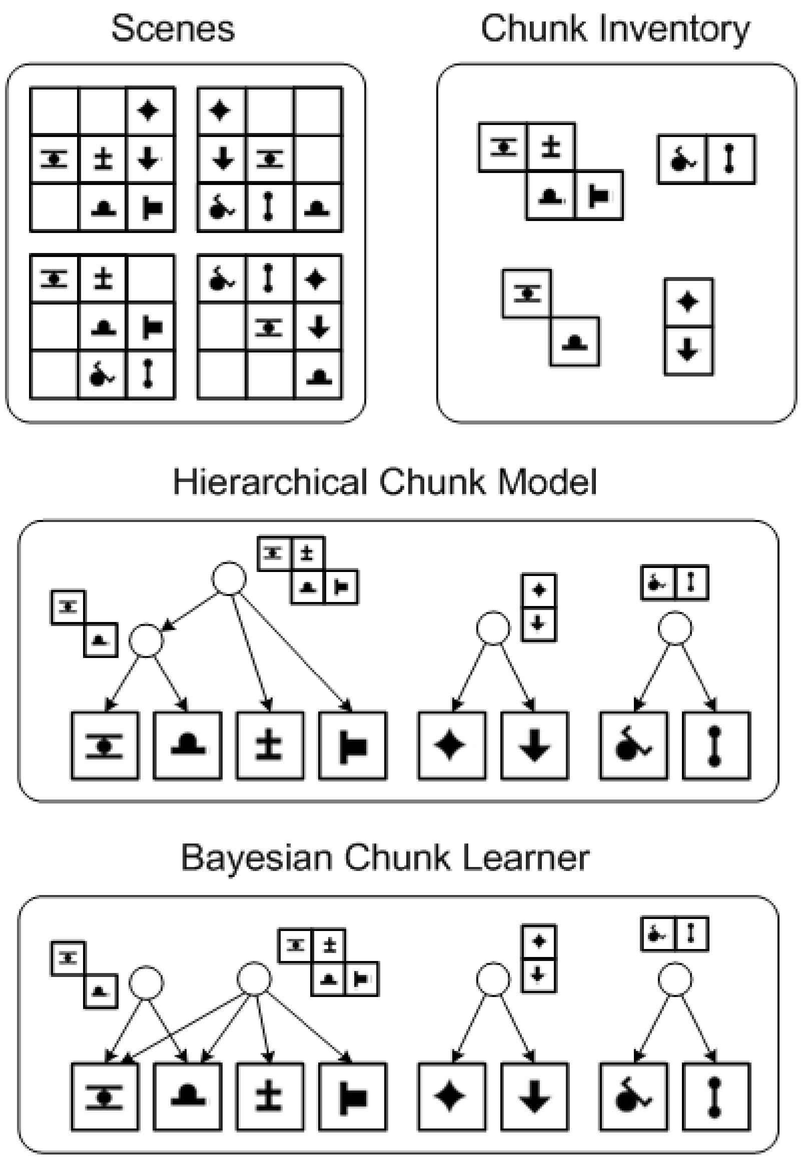

Figure 4.1: An illustration of the proposed hierarchical model. The training scenes in the left column are generated by the inventory of chunks in the middle column. The right column shows how our hierarchical model and the Bayesian chunk learner model (BCL) may provide different explanations over the same training scenes.

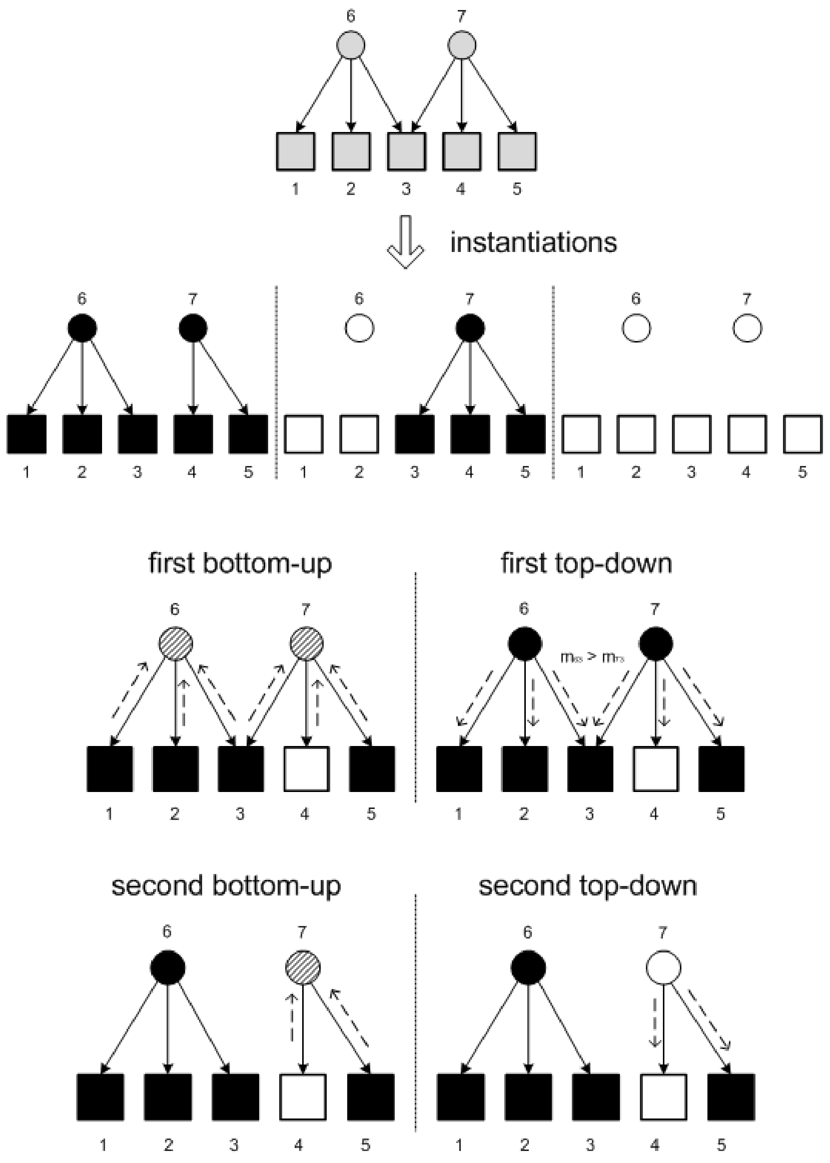

Figure 4.2: Searching for the model's maximum-likelihood states in two rounds of belief propagation. Black, white, or striped nodes indicate that their ML states are on, off, or undecided respectively.

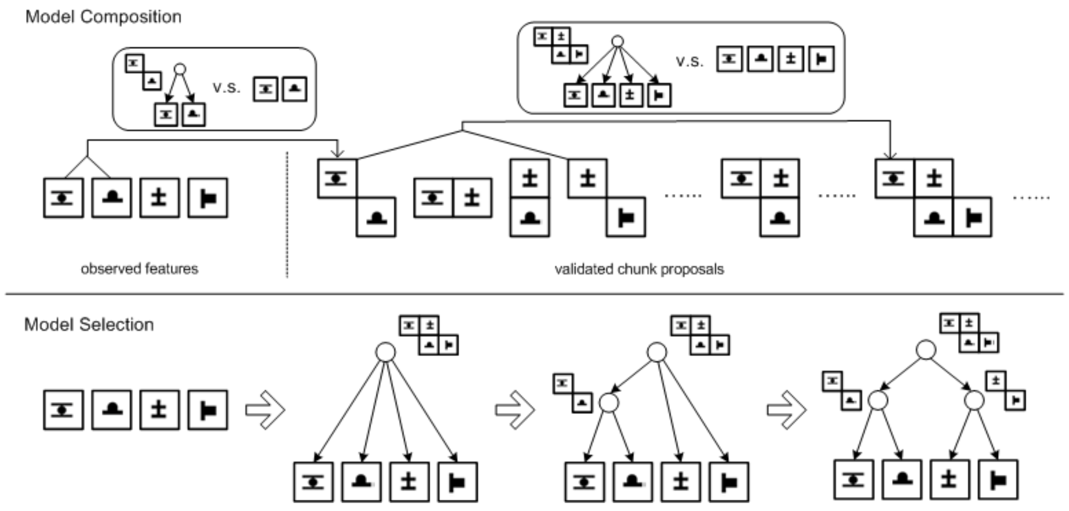

Figure 4.3: Learning the model structure in two stages. Top: the model composition stage starting from 4 basic features. Composed chunks are listed in a queue. Rectangles show the likelihood ratio tests performed during the process. Bottom: the model selection stage for the same features.

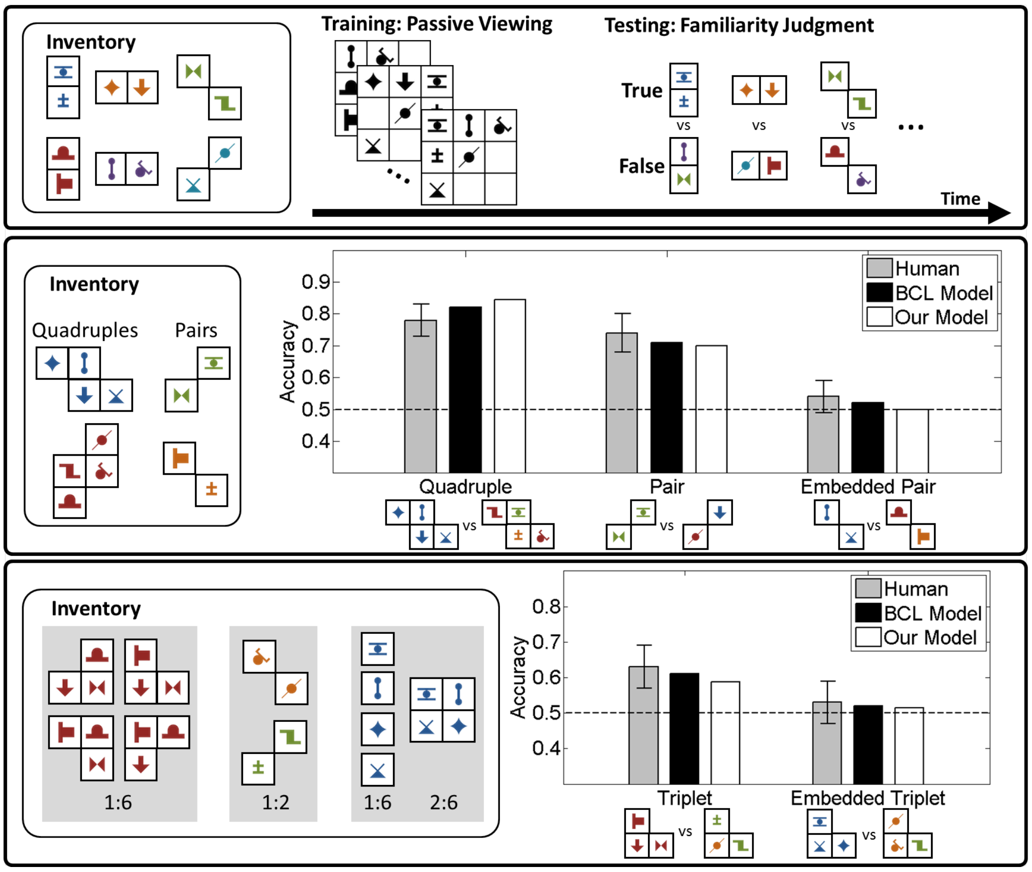

Figure 4.4: Illustration of previous experiments on human chunk learning, along with predictions from our model and the BCL model. Top: the general procedure of these experiments. Middle: chunk inventory and results for an experiment in both [FA05] and [OFA08]. Bottom: chunk inventory and results for another experiment in [OFA08].

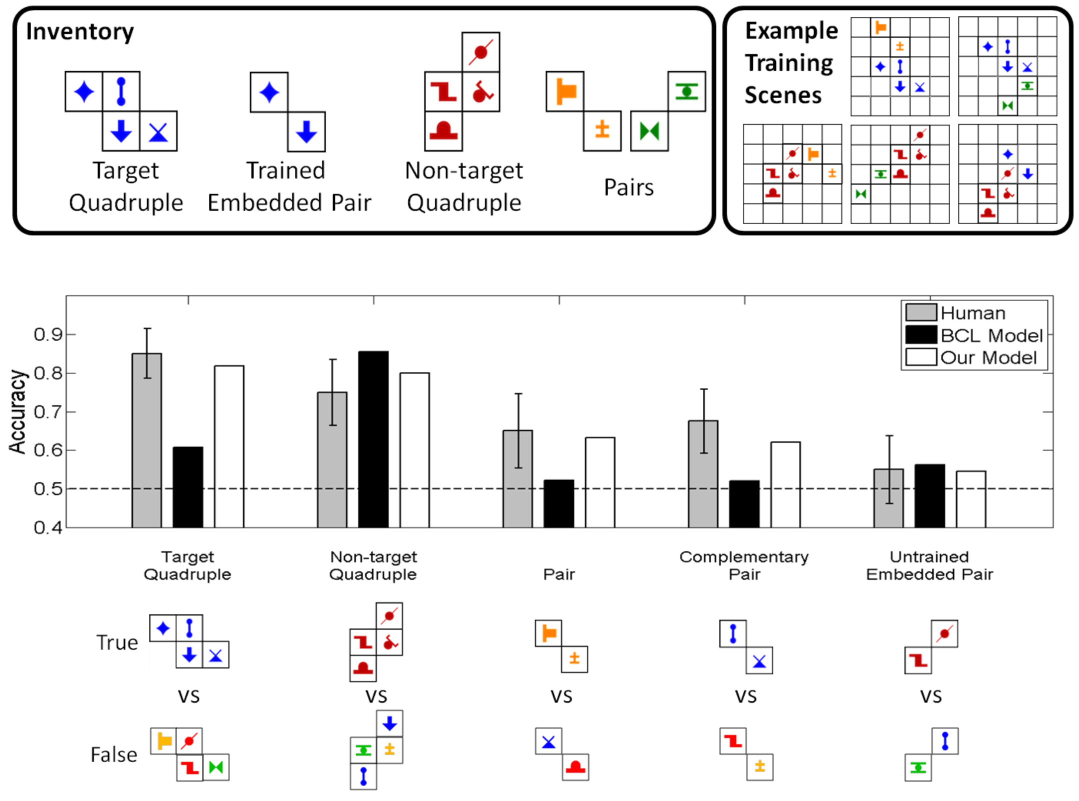

Figure 4.5: Top: the chunk inventory and example training scenes used in our new experiment. Colors are for illustration purpose only and were not used in the experiment. Bottom: testing trials used in the experiment, along with human performance and our model's prediction.

CHAPTER 5

Summary and Discussions

In this thesis, we tackle the problem of learning a generative representation of multiple object categories for the application of simultaneous recognition and reconstruction. We learn such a representation in two separate and complimentary modules.

The first module is a dictionary of deformable patches. Our approach relies on two key innovations – introducing a pre-defined set of transformations on patches to enrich the search space, and designing a parallel framework on Graphical Processors (GPUs) for matching a large number of deformable templates to a large set of images efficiently. We illustrate our method on two handwritten digit databases – MNIST and USPS, and report state-of-art recognition performance without using any domain-specific knowledge on digits. We briefly show that our dictionary has many desirable properties: it includes intuitive low- and mid-level structures, it is sufficient to synthesize digits, it gives sparse representations of digits, and contains elements which are useful for discrimination. In addition, we are the first dictionary learning method to report good results when transferring the learned dictionary between different datasets.

The key insight of this work is that the emergence of parallel computing hardware such as GPUs enable the aggregation of certain computation (deformable template matching by normalized correlation in our case) on a unprecedented scale. We are then able to opt for simpler techniques (intensity patches as features, greedy feature selection based on frequency, and linear classifier) without losing performance as compared to more complex counterparts. Besides, we get much faster speed and more intuitive intermediate representations at each step of the whole system.

The second module is a hierarchical model encoding spatial contributions of low-level descriptors. We presented a novel hierarchical visual chunk model, along with a fast unsupervised algorithm to learn the structure and parameters of this model. We demonstrated that the learned model is able to predict previously reported human chunk learning behavior, and that the training takes less than a thousandth of the time required by an ideal Bayesian chunk learner. Our model captures dependence between hidden visual chunks, especially the relation between objects and parts, which are essential to visual processing but absent from previous models. In addition, we reported a new experiments that yielded human results consistent with our model's predictions concerning learning of chunks together with their parts, the latter of which cannot be predicted by previous single-layer chunk models.

References

[Bar89] H.B. Barlow. "Unsupervised learning." *Neural Computation*, **1**:295–311, 1989.

[BBL10] Y. Boureau, F. Bach, Yann LeCun, and Jean Ponce. "Learning mid-level features for recognition." In *CVPR*, 2010.

[BGP98] E. Bienenstock, S. Geman, and D. Potter. "Compositionality, MDL Priors, and Object Recognition." In *Advances in Neural Information Processing Systems 10 (NIPS-97)*, 1998.

[BMP02] S. Belongie, J. Malik, and J. Puzicha. "Shape Matching and Object Recognition Using Shape Contexts." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **24**(4):509–522, April 2002.

[BPP08] E. Bart, I. Porteous, P. Perona, and M. Welling. "Unsupervised Learning of Visual Taxonomies." In *CVPR*, 2008.

[BU02] E. Borenstein and S. Ullman. "Class specific top-down segmentation." In *ECCV*, 2002.

[CCD07] A. Choi, M. Chavira, and A. Darwiche. "Node Splitting: A Scheme for Generating Upper Bounds in Bayesian Networks." In *Proceedings of the Twenty-Third Annual Conference on Uncertainty in Artificial Intelligence (UAI-07)*, pp. 57–66, 2007.

[CLN10] A. Coates, H. Lee, and A. Ng. "An analysis of single-layer networks in unsupervised feature learning." In *NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.

[CS73] W.G. Chase and H.A. Simon. "Perception in chess." *Cognitive Psychology*, **4**:55–81, 1973.

[EF01] A. A. Efros and W. T. Freeman. "Image Quilting for Texture Synthesis and Transfer." In *SIGGRAPH*, 2001.

[EKS96] M. Ester, H. Kriegel, J. Sander, and X. Xu. "A density-based algorithm for discovering clusters in large spatial databases with noise." In *KDD*, p. 226 231, 1996.

[EU05] B. Epshtein and S. Ullman. "Hierarchical features for object classification." In *ICCV*, 2005.

[FA01] J. Fiser and R.N. Aslin. "Unsupervised statistical learning of higher-order spatial structures from visual scenes." *Psychological Science*, **6**:499–504, 2001.

[FA05] J. Fiser and R.N. Aslin. “Encoding multi-element scenes: Statistical learning of visual feature hierarchies.” *Journal of Experimental Psychology: General*, **134**:521–537, 2005.

[FH03] L. Feigenson and J. Halberda. “Infants chunk object arrays into sets of individuals.” *Cognition*, **91**:173–190, 2003.

[FJ03] B. J. Frey and N. Jojic. “Transformation-Invariant Clustering and Dimensionality Reduction Using the EM Algorithm.” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, **25**(1), 2003.

[FL07] S. Fidler and A. Leonardis. “Towards Scalable Representation of Object Categories: Learning a Hierarchy of Parts.” In *CVPR*, 2007.

[GLC01] F. Gobet, P.C.R. Lane, S. Croker, P.C.H. Cheng, G. Jones, I. Oliver, and J. M. Pine. “Chunking mechanisms in human learning.” *Trends in Cognitive Sciences*, **5**:236–243, 2001.

[Hel25] H. von Helmholtz. *Treatise on physiological optics.* Optical Society of America, Washington, DC, 1925.

[HK02] B. Haasdonk and D. Keysers. “Tangent distant kernels for support vector machines.” In *ICPR*, 2002.

[HOT06] G. E. Hinton, S. Osindero, and Y. W. Teh. “A fast learning algorithm for deep belief nets.” *Neural Computation*, **18**(7):1527–1554, 2006.

[JG06] Y. Jin and S. Geman. “Context and hierarchy in a probablistic image model.” In *CVPR*, 2006.

[JKR09] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. “What is the best multi-stage architecture for object recognition?” In *ICCV*, 2009.

[Joa] T. Joachims. “SVM-Light package.” `http://svmlight.joachims.org`.

[LGR09] H. Lee, R. Grosse, R. Ranganath, and A. Ng. “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations.” In *ICML*, 2009.

[MBP08] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. “Supervised Dictionary Learning.” In *NIPS*, 2008.

[Mil56] G.A. Miller. “The Magical Number Seven, Plus or Minus Two: Some Limits on our Capacity for Processing Information.” *Psychological Review*, **63**:81–97, 1956.

[MNI] “MNIST handwritten digits database.” `http://yann.lecun.com/exdb/mnist/`.

[OF96] B. A. Olshausen and D. J. Field. “Emergence of simple-cell receptive field properties by learning a sparse code for natural images.” *Nature*, **381**:607 – 609, 1996.

[OFA08] G. Orban, J. Fiser, R.N. Aslin, and M. Lengyel. “Bayesian Learning of visual chunks by human observers.” *Proceedings of the National Academy of Sciences (PNAS)*, **105**(7):2745–2750, 2008.

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference (2nd ed.)*. Morgan Kaufmann, San Fracisco, CA, 1988.

[RBL07] R. Raina, A. Battle, H. Lee, B. Packer, and A. Ng. “Self-taught learning: Transfer learning from unlabeled data.” In *ICML*, 2007.

[RHB07] M. Ranzato, F. Huang, Y. Boureau, and Y. LeCun. “Unsupervised learning of invariant feature hierarchies with applications to object recognition.” In *CVPR*, 2007.

[USP] “USPS handwritten digits database.” `http://www-stat.stanford.edu/~tibs/ElemStatLearn/`.

[WSF07] Y. N. Wu, Z. Si, C. Fleming, and S.C. Zhu. “Deformable template as active basis.” In *ICCV*, 2007.

[ZCT10] L. Zhu, Y. Chen, A. Torralba, W. Freeman, and A. Yuille. “Part and Appearance Sharing: Recursive Compositional Models for Multi-View Multi-Object Detection.” In *CVPR*, 2010.

[ZCY07] L. Zhu, Y. Chen, and A.L. Yuille. “Unsupervised Learning of a Probabilistic Grammar for Object Detection and Parsing.” In *Advances in Neural Information Processing Systems 19 (NIPS-06)*, 2007.