Learning a Dictionary of Deformable Patches using GPUs

Xingyao Ye
Dept of Statistics, UCLA
yexy@stat.ucla.edu

Alan Yuille
Dept of Statistics, Computer Science, and Psychology, UCLA
yuille@stat.ucla.edu

Abstract

We propose a simple method for learning a dictionary of deformable patches for simultaneous shape recognition and reconstruction. 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.

1. Introduction

Learning dictionaries of generative image features has long been a topic of interest in computer vision research. The learned dictionaries provide an intuitive and economical mid-level representation for visual processing systems. They facilitate the information flow of the system both bottom-up and top-down. In bottom-up recognition tasks, the features can be used as input to discriminative models of objects and scenes. And in top-down reconstruction tasks, they serve as bases from which images can be generated.

However, state-of-art object recognition systems rarely employ these dictionaries for two reasons. First, learning these dictionaries is complex and takes a long time, and the gain in recognition performance over simple 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.

In this paper, we tackle the problem of learning a dictionary of image patches for simultaneous recognition and reconstruction of object shapes. For concreteness, we work on the MNIST and USPS handwritten digit databases. The dictionary we learn is comprehensive – the patches are of varying sizes, some corresponding to generic patterns such as bars and corners, others capturing 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 work-

station 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 is 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 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.

The structure of this paper is as follows. Related works are reviewed in Section 2. The definition and use of deformable patch dictionary is described in Section 3. Section 4 presents the learning algorithm. In Section 5 we report results for dictionary learning and handwritten digit recognition. Finally we discuss conclusions and future work in Section 6.

2. Related 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 [17], texture [7], appearances [3], and object categories [11]. In this paper, 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 [4] as well as object recognition [8]. But their patches were not deformable. Wu *et al* [19] introduced local deformation in their active basis model, but the bases were pre-specified to be Gabor functions. Zhu *et al* [20] and Fidler *et al* [10] 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* [12] has led to a series of works for learning unsupervised generative dictionaries [18, 15, 13] using hierarchical convolutional networks. However, as discussed in [5, 6], 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* [16], 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.

Very few previous works have addressed the issue of transferability of the learned dictionaries between different datasets. Raina *et al* reported modest performance in classifying English characters using features learned from digits [**?**]. We are the first to show good performance in dictionary transfer on handwritten digit recognition.

3. The D-Patch Representation

In this section, we propose a D-Patch (deformable patches) representation for the problem of simultaneous object recognition and reconstruction. We describe three key aspects of this representation: 1) the definition of the D-Patch dictionary (see Section 3.1), 2) how images are represented in terms of the dictionary, and 3) how the dictionary facilitates recognition and reconstruction (see Section 3.2).

3.1. The D-Patch Dictionary

We choose to use the most basic form of image descriptor - patches in raw intensity values - as features in our generative dictionary. These are simple, easy to interpret, and are suitable both for discrimination and generative (e.g., reconstruction) tasks. We notes that features like HOG, SIFT, shape context are based on histograms, and so are not well suited for reconstruction (despite their successes for discrimination). Filterbanks of Gabor or wavelet functions are alternative descriptors, but seem less intuitive than intensity patches and have restricted transformations.

We design our patches to be flexible. They can be rectangles of any sizes larger than 5×5. They can come in any orientations, covering non-rectangular areas. More specifically, we describe a D-Patch P in terms of a seed patch S (always in upright orientation) and its transformation parameters T.

$$P = (S, T), \qquad T = (s_x, s_y, \theta, x, y) \tag{1}$$

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.

Our dictionary is designed to cover a wide spectrum of local image patterns from smaller, generic ones (edgelets,

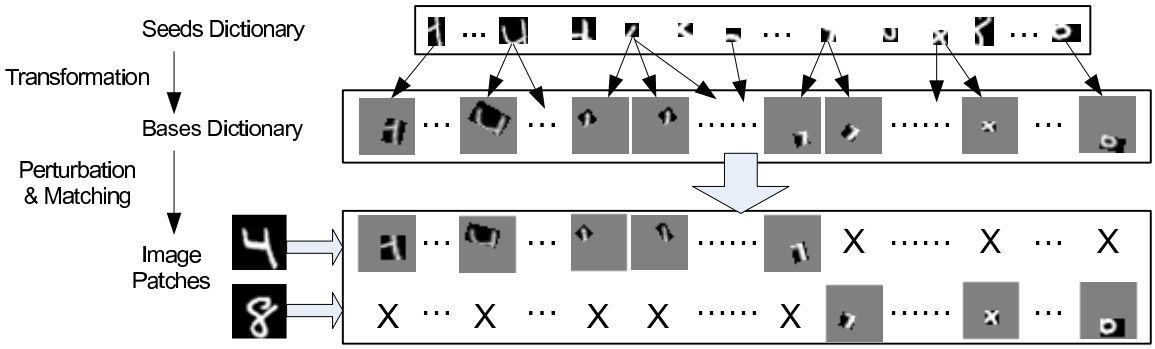

Figure 1. Illustration of the D-Patch representation. The dictionary of deformable patches are generated from a set of seeds plus transformations. Patches are perturbed and matched to local image regions.

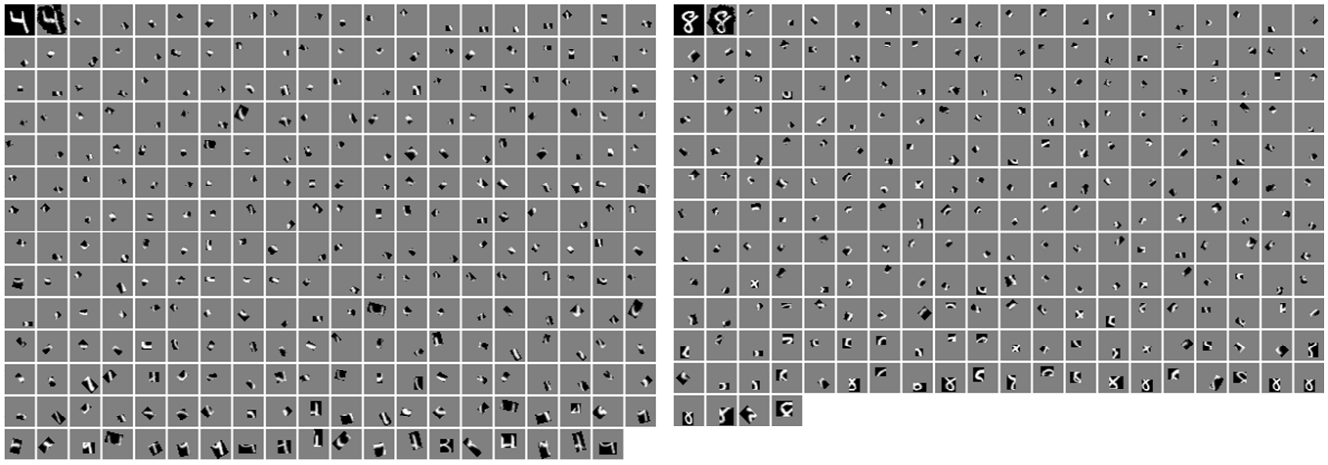

Figure 2. Two images from the MNIST dataset (top left panel) together with their reconstructed versions using our learned dictionary (adjacent panel) and a complete list of fired patches. The patches are sorted in descending order of their matching score.

strokes, corners) to larger, object-specific ones (T-junctions, X-junctions, rings). See Figure 1 for an illustration.

3.2. The Image Model

To obtain a D-Patch representation of an image, we first match the whole dictionary to the image. The measure of similarity between a patch and its corresponding image region is the normalized correlation between two vectors:

$$NC(X,Y) = \sum_{x\in X}\sum_{y\in Y}\frac{(x-\bar{x})(y-\bar{y})}{\sigma_x\sigma_y} \in [-1,1] \quad (2)$$

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.

$$\tilde{P} = (S, T+\Delta_T) \qquad \Delta_T \leq |(2,2,15^\circ,2,2)| \quad (3)$$

$$pNC(I,P) = max(I,\tilde{P}) \quad (4)$$

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. We use $\tau = 0.8$ in this paper.

$$F_{i,j} = \begin{cases} 1 & \text{if } pNC(I_i,P_j) > \tau \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

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 shows the fired patches among our learned dictionary for two digit images.

The reconstruction of an image from its 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.

$$I(x) = P^*(x) \qquad P^*(x) = \underset{P_j : x \in P_j \wedge F_j = 1}{\operatorname{argmax}} pNC(I, P_j) \tag{6}$$

4. Learning the D-Patch Dictionary

In this section, 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 supposed to discover a reasonable set of image patches that can fulfill both the recognition and the reconstruction tasks outlined in the previous section.

The algorithm proceeds in four stages, which are described in the four following subsections. First, a set of seed patches are obtained from random sampling and clustering (Section 4.1. Second, a pre-defined set of transformations are applied to all seeds, resulting in a very large set of preliminary patches (Section 4.2). Third, we match these patches to all training images in a parallel framework developed on GPUs (Section 4.3). Finally, the dictionary is constructed through a greedy feature selection process (Section 4.4).

4.1. Obtaining Seed Patches

To obtain a set of seeds, we first extract rectangular patches from 100 randomly selected training images. In this paper, the images are of size 28×28. We extract patches at three different sizes, 28×28, 14×14 and 7×7, at all possible pixel locations. This dense multi-scale sampling strategy ensures the completeness of the initial 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. After this stage, about 10,000 patches of various sizes remain.

Finally, a clustering algorithm is employed to reduce the redundancy of shape patterns. We used a density-based clustering method described in [9]. 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. 5. 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.

4.2. Transformations

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.

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.

We list the possible transformations in Table 1. The transformation process, from scaling to rotation, then translation, is illustrated in Figure 3. In all figures of this paper, image areas not covered by a patch is marked in grey.

For each seed, there are roughly 10,000 transformation settings in total. 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.

4.3. Parallel Matching using GPUs

The matching operation between a patch and an image, as described by Eqn 5, 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.

Table 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

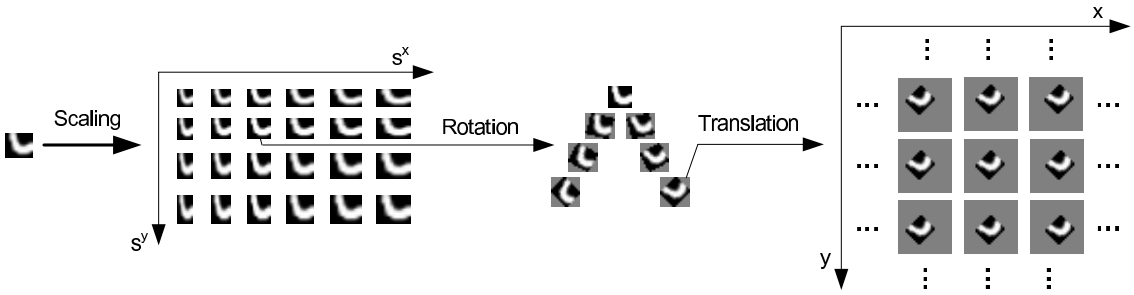

Figure 3. Applying transformations to a seed patch.

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. Preloading 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.

4.4. 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. 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 2 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.

5. Experimental Results

We tested our methods on two widely-used handwritten digits datasets. The MNIST dataset contains 60,000 training images and 10,000 testing images [1]. The USPS dataset contains 7,291 training images and 2,007 testing images [2]. We upsampled the original 16×16 USPS images to 20×20 and centered them in a 28×28 bounding box, following the preprocessing steps of MNIST.

Table 2. Statistics of the learned dictionaries.

Dataset	Seeds	Prelim Patches	Training Images	Learned Patches	Firing Freq. (Train)	Firing Freq. (Test)
MNIST	1,140	15,427,836	1,000	10,264	521.7 (5.08%)	528.7 (5.15%)
			10,000	9,714	507.9 (5.23%)	520.4 (5.36%)
			60,000	9,727	505.8 (5.20%)	517.5 (5.32%)
USPS	603	9,685,896	7,291	6,343	509.1 (8.03%)	526.2 (8.30%)

5.1. The Learned Dictionary

In this subsection 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 4 a subset of the learned dictionary. The dictionary 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 5. The synthesized images are obtained by the method described at the end of Section 3.2. 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 2 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.

Finally, we demonstrate the discriminative power of the patches in Figure 6. 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.

5.2. Handwritten Digits Recognition

We achieved state-of-art handwritten digit recognition results on both highly competitive benchmark datasets. We follow the standard machine learning approach on this 10-way classification problem, which is to train 45 1-vs-1 classifiers and predict categories based on majority voting of these classifiers. We used the standard SVM-Light package [14] 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).

Table 3 shows that our classification error rates are better than the best competing methods from supervised dictionary learning [16], which also reported performance on both MNIST and USPS. Our results are also better than deep convolutional networks method when training on a smaller set of images [15]. Our recognition performance is very close to the individual state-of-art on either MNIST [13] or USPS [**?**]. Most significantly, 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.

5.3. Dictionary Transfer

Finally we test 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 D-Patch dictionary. For example, we use the patches learned from MNIST to represent both the USPS training and testing images, and train SVMs based on these transferred features. The other setting transfers the seed candidate set only and learns a new D-Patch dictionary on the target dataset. For example, we use the seeds obtained from 100 MNIST images to learn patches on the USPS training set. The new bases dictionary are used as features to train and test SVMs on USPS. Results are summarized in Table 4.

In general, transferred dictionaries work effectively on another dataset. Seeds appear to be a better choice to trans-

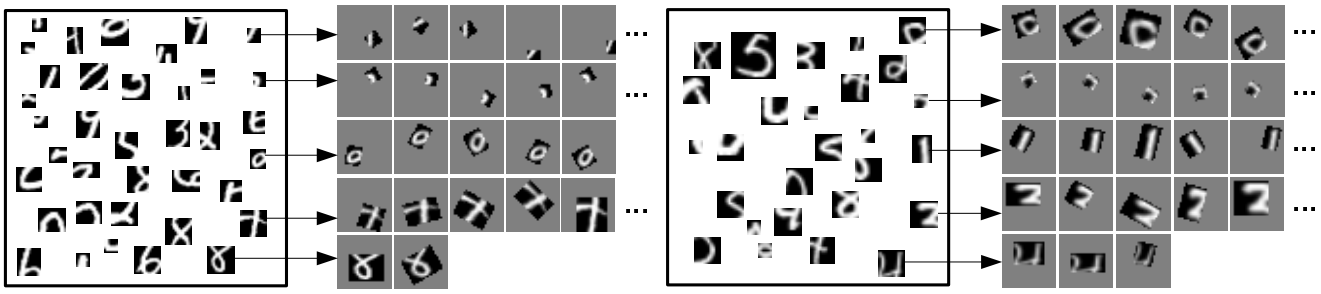

Figure 4. Examples of learned seeds and patches. The left half of the figure shows seeds learned from MNIST, as well as the most frequent (up to 5) bases from some of the seeds. The right half shows shows seeds and patches from USPS.

Figure 5. Synthesized images using the learned and transferred D-Patch dictionaries. Each of the four grid shows 100 images of the same dataset synthesized from a dictionary learned on the same or a different dataset. From left to right, the images come from MNIST, USPS, MNIST, USPS, and the dictionary are learned from MNIST, USPS, USPS, MNIST.

Table 4. 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		3.39%	3.14%
MNIST	USPS	USPS	3.03%	2.89%
USPS	USPS		3.04%	2.84%
USPS	USPS		0.96%	0.87%
USPS	MNIST	MNIST	0.78%	0.66%
MNIST	MNIST		0.70%	0.60%

fer since they encode invariant structures, 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 larger training set.

We also show synthesis results of transferring USPS dictionary to MNIST testing set in Fig 5. This transfer direction is more difficult than the other way around. We can observe a few less-than-perfect synthesis in the figure, but still the majority of the structures are captured.

6. Discussions

This paper presented 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 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.

In future work, we plan to extend our approach and the emphasis on simplicity and parallel computation to more complex shapes and images and use the dictionary to learn

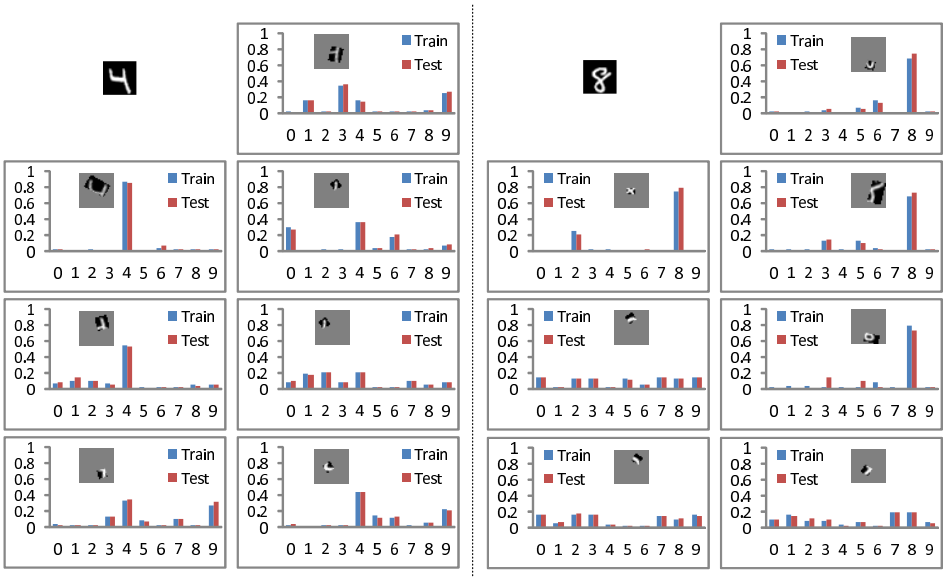

Figure 6. Using learned bases as features for recognition. For two MNIST images, the activation histograms are shown for some of the fired bases. Observe that some bases are very discriminative, e.g. U-shape for 4, X-junction for 8, etc.

Table 3. Classification error rates

Dataset	Training	Linear SVM	RBF SVM	[16]	[15]	[13]	[?]
MNIST	1,000	2.73%	2.56%		2.62%		
	10,000	1.40%	1.26%				
	60,000	0.70%	0.60%	1.05%	0.82%	0.52%	
USPS	7,291	3.04%	2.84%	3.54%			2.40%

sophisticated object models that solves multiple vision tasks (e.g. detection and segmentation) simultaneously.

References

[1] MNIST handwritten digits database. http://yann.lecun.com/exdb/mnist/. 5

[2] USPS handwritten digits database. http://www-stat.stanford.edu/~tibs/ElemStatLearn/. 5

[3] E. Bart, I. Porteous, P. Perona, and M. Welling. Unsupervised learning of visual taxonomies. In *CVPR*, 2008. 2

[4] E. Borenstein and S. Ullman. Class specific top-down segmentation. In *ECCV*, 2002. 2

[5] Y. Boureau, F. Bach, Y. LeCun, and J. Ponce. Learning mid-level features for recognition. In *CVPR*, 2010. 2

[6] 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*. 2

[7] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *SIGGRAPH*, 2001. 2

[8] B. Epshtein and S. Ullman. Hierarchical features for object classification. In *ICCV*, 2005. 2

[9] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, page 226 231, 1996. 4

[10] S. Fidler and A. Leonardis. Towards scalable representation of object categories: Learning a hierarchy of parts. In *CVPR*, 2007. 2

[11] 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. 2

[12] G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006. 2

[13] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *ICCV*, 2009. 2, 6, 8

[14] T. Joachims. SVM-Light package. http://svmlight.joachims.org. 6

[15] H. Lee, R. Grosse, R. Ranganath, and A. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*, 2009. 2, 6, 8

[16] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Supervised dictionary learning. In *NIPS*, 2008. 2, 6, 8

[17] 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. 2

[18] M. Ranzato, F. Huang, Y. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *CVPR*, 2007. 2

[19] Y. N. Wu, Z. Si, C. Fleming, and S. Zhu. Deformable template as active basis. In *ICCV*, 2007. 2

[20] 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. 2