Learning Invariant Features through Topographic Filter Maps

Koray Kavukcuoglu Marc'Aurelio Ranzato Rob Fergus Yann LeCun
Courant Institute of Mathematical Sciences
New York University
{koray,ranzato,fergus,yann}@cs.nyu.edu

Abstract

Several recently-proposed architectures for high-performance object recognition are composed of two main stages: a feature extraction stage that extracts locally-invariant feature vectors from regularly spaced image patches, and a somewhat generic supervised classifier. The first stage is often composed of three main modules: (1) a bank of filters (often oriented edge detectors); (2) a non-linear transform, such as a point-wise squashing functions, quantization, or normalization; (3) a spatial pooling operation which combines the outputs of similar filters over neighboring regions. We propose a method that automatically learns such feature extractors in an unsupervised fashion by simultaneously learning the filters and the pooling units that combine multiple filter outputs together. The method automatically generates topographic maps of similar filters that extract features of orientations, scales, and positions. These similar filters are pooled together, producing locally-invariant outputs. The learned feature descriptors give comparable results as SIFT on image recognition tasks for which SIFT is well suited, and better results than SIFT on tasks for which SIFT is less well suited.

1. Introduction

A crucially important component of every recognition system is the feature extractor. Much of the recent proposals for object recognition systems are based on feature descriptors extracted from local patches placed at regularly-spaced grid-points on the image [13, 11, 25, 18, 22]. The most successful and most commonly-used descriptors such as SIFT and HoG [15, 3] are designed to be invariant (or robust) to minor transformations of the input, such as translations, rotations, and other affine transforms and distortions. The present paper proposes a new method to *automatically learn* locally-invariant feature descriptors from data in an unsupervised manner. While good descriptors have been devised for grayscale image recognition, the design of good descriptors for other types of input data is a complex task. The ability to learn the features would allow us to automatically construct good descriptors for new image types (e.g. multispectral images, range images), and for other input modalities (e.g. audio, sonar).

Most existing local descriptors are based on a simple architecture: the patch is convolved with a filter bank (often consisting of oriented edge detectors), the outputs of which are rectified and often normalized and quantized. Then, the outputs of each filter are spatially pooled using a simple addition or a max operator, so as to build local bags of features. The pooling operation makes the descriptor robust to minor changes in the position of individual features. This architecture is somewhat similar (and inspired by) the most commonly accepted model of early areas of the mammalian primary visual cortex: simple cells detect oriented edges at various locations and scales (playing the same role as the filter bank). Highly-active simple cells inhibit other cells at neighboring locations and orientations (similarly to local normalization and/or quantization), while complex cells spatially pool the rectified outputs of complex cells, so as to create a local invariance to small shifts (like the pooling operation) [7, 25, 21]. The method proposed here simultaneously learns the filters and the pooling function, so that filters that fire on similar image patches end up in the same pool. As a result, similar patches will produce similar descriptors.

The problem of learning low-level image features has become a topic of growing interest in recent years. Several authors have proposed unsupervised methods to learn image descriptors based on sparse/overcomplete decomposition [19, 14, 23], but none had explicit provision for local invariance. Supervised learning methods have long been used in conjunction with Convolutional Networks to learn low-level, locally invariant features that are tuned to the task at hand [12, 13], but these methods require large numbers of labeled samples. A number of different proposals have appeared for unsupervised learning of locally-invariant descriptors, which also use sparsity criteria [7, 20, 8, 22].

Our aim is to learn the filter bank stage and the pooling stage simultaneously, in such a way the filters that belong to the same pool extract similar features. Rather than learning descriptors that are merely invariant to small shift (a property that can easily be built by hand), our goal is to learn descriptors that are also invariant to other types of transformations, such as rotations and certain distortions. Our solution is to pre-wire (before learning) which filters outputs are pooled together, and to let the underlying filters learn their coefficients. The main idea, inspired by [8], is to minimize a sparsity criterion on the pooling units. As a result, filters that are pooled together end up extracting similar features.

Several authors have proposed methods to learn pooled features in the context of computational models of the mammalian primary visual cortex. The idea relies on imposing sparsification criteria on small groups of filter outputs [10, 6, 8], which can be related to the Group Lasso method for regularization [27]. When the filters that are pooled together are organized in a regular array (1D or 2D), the filters form *topographic maps* in which nearby filters extract similar features [20, 7], with patterns similar to what is found in the visual cortex.

To the best of our knowledge, the present work is the first time a trainable topographically-organized feature map is used for extracting locally invariant image descriptors for image recognition. The following sections describe the training procedure, and compare the descriptors thereby obtained with a number of standard descriptors such as SIFT. The experiments compare recognition accuracies on Caltech 101, MNIST and Tiny Images datasets using various recognition architectures fed with various descriptors.

2. Algorithm

It is well established that sparse coding algorithms applied to natural images learn basis functions that are localized oriented edges and resemble the receptive fields of simple cells in area V1 of the mammalian visual cortex [19]. These methods produce feature representation that are sparse, but not invariant. If the input pattern is slightly distorted, the representation may change drastically. Moreover, these features represent information about local texture, and hence, are rather inefficient when used to preprocess whole images because they do not exploit the redundancy in adjacent image patches. Finally, most sparse coding algorithms [19, 14, 17, 24, 4] have found limited applications in vision due to the high computational cost of the iterative optimization required to compute the feature descriptor.

In this paper, we introduce an algorithm, named *Invariant Predictive Sparse Decomposition* (IPSD), that: (1) learns features that are invariant to small variations inherent in the data, and (2) produces more efficient representations because they can be compact and directly computed using a feed-forward function, without requiring the use of any iterative optimization procedure.

2.1. Learning an Over-complete Dictionary of Basis Functions

Sparse coding algorithms represent an input signal $x \in \mathcal{R}^m$ using a linear combination of basis functions that are columns of the dictionary matrix $\mathcal{D} \in \mathcal{R}^{m \times n}$, using coefficients $z \in \mathcal{R}^n$, with $n > m$. Since the linear system is under-determined, a sparsity constraint is added that prefers most of the coefficients to be zero for any given input. Many sparse coding algorithms have been proposed in the literature and in this work we focus on the following convex formulation:

$$\mathcal{L} = \min \frac{1}{2}||x - \mathcal{D}z||_2^2 + \lambda \sum_i |z_i| \tag{1}$$

This particular formulation has been extensively studied [19, 4, 2, 14, 17, 24, 16, 9], and it has also been extended to the case when the dictionary $\mathcal{D}$ is learned, thus adapting to the statistics of the input. The basic idea is to minimize the same objective of Eqn. 1 alternatively over coefficients z for a given dictionary $\mathcal{D}$, and then over $\mathcal{D}$ for a given set of z. Note that each column of $\mathcal{D}$ is required to be unit ℓ^2 norm (or bounded norm) in order to avoid trivial solutions that are due to the ambiguity of the linear reconstruction (for instance, the objective can be decreased by respectively dividing and multiplying z and $\mathcal{D}$ by a constant factor).

2.2. Modeling Invariant Representations

Although the sparse coding algorithm outlined above can learn representations that are sparse, they are not invariant: a small change in the input signal x may result in a large change in the coefficients z [24]. We now describe how the sparsity term in Eqn. 1 can be modified to create coefficients that are invariant to perturbations in the input signal.

The overall idea [8] is to arrange the z's into a 2D map (or some other topology) and then pool the squared coefficients of z across overlapping windows. Then, the square of the the filter outputs within each sub-window are summed, and its square root is computed. More formally, let the map of z contain K overlapping neighborhoods P_i. Within each neighborhood i, we sum the squared coefficients z_j (weighted by a fixed Gaussian weighting function centered in the neighborhood) and then take the square root. This gives the activation $v_i = \sqrt{\sum_{j \in P_i} w_j z_j^2}$, where w_j are the Gaussian weights. The overall sparsity penalty is the sum of each neighborhood's activation: $\sum_{i=1}^{K} v_i$. Figure 1(a) illustrates this scheme. Thus, the overall objective function is now:

$$\mathcal{L}_I = \frac{1}{2}||x - \mathcal{D}z||_2^2 + \lambda \sum_{i=1}^{K} \sqrt{\sum_{j \in P_i} w_j z_j^2} \tag{2}$$

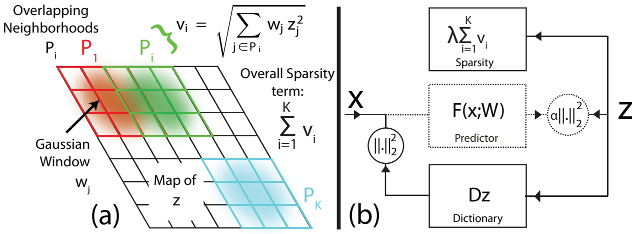

Figure 1. **(a)**: The structure of the block-sparsity term which encourages the basis functions in $\mathcal{D}$ to form a topographic map. See text for details. **(b)**: Overall architecture of the loss function, as defined in Eqn. 4. In the generative model, we seek a feature vector z that simultaneously approximate the input x via a dictionary of basis functions $\mathcal{D}$ and also minimize a sparsity term. Since performing the inference at run-time is slow, we train a prediction function $F(x;W)$ (dashed lines) that directly predicts the optimal z from the input x. At run-time we use only the prediction function to quickly compute z from x, from which the invariant features v_i can computed.

The modified sparsity term has a number of subtle effects on the nature of z that are not immediately obvious:

- The square root in the sum over i encourages sparse activations *across* neighborhoods since a few large activations will have lower overall cost than many small ones.
- *Within* each neighborhood i, the coefficients z_j are encouraged to be similar to one another due to the z_j^2 term (which prefers many small coefficients to a few large ones). This has the effect of encouraging similar basis functions in $\mathcal{D}$ to be spatially close in the 2D map.
- As the neighborhoods overlap, these basis functions will smoothly vary across the map, so that the coefficients z_j in any given neighborhood i will be similar.
- If the size of the pooling regions is reduced to a single z element, then the sparsity term is equivalent to that of Eqn. 1.

The modified sparsity term means that by minimizing the loss function $\mathcal{L}_I$ in Eqn. 2 with respect to both the coefficients z and the dictionary $\mathcal{D}$, the system learns a set of basis functions in $\mathcal{D}$ that are laid out in a *topographic map* on the 2D grid.

Since the nearby basis functions in the topographic map are similar, the coefficients z_j will be similar for a given input x. This also means that if this input is perturbed slightly then the pooled response within a given neighborhood will be minimally affected, since a decrease in the response of one filter will be offset by an increased response in a nearby one. Hence, we can obtain a locally robust representation by taking the pooled activations v_i as features, rather than z as is traditionally done.

Since invariant representations encode similar patterns with the same representation, they can be made more compact. Put another way, this means that the dimensionality of v can be made lower than the dimensionality of z without loss of useful information. This has the triple benefit of requiring less computation to extract the features from an image, requiring less memory to store them, and requiring less computation to exploit them.

The 2D map over z uses circular boundary conditions to ensure that the pooling wraps smoothly around at the edges of the map.

2.3. Code Prediction

The model proposed above is generative, thus at test-time for each input region x, we will have to perform inference by minimizing the energy function $\mathcal{L}_I$ of Eqn. 2 with respect to z. However, this will be impractically slow for real-time applications where we wish to extract thousands of descriptors per image. We therefore propose to *train a non-linear regressor that directly maps input patches x into sparse representations z*, from which the invariant features v_i can easily be computed. At test-time we only need to present x to the regression function which operates in feed-forward fashion to produce z. *No iterative optimization is needed.*

For the regressor, we consider the following parametrized function:

$$F(x;W) = F(x;g,M,b) = g\tanh(Mx+b) \tag{3}$$

where $M \in \mathcal{R}^{m\times n}$ is a filter matrix, $b \in \mathcal{R}^m$ is a vector of biases, $\tanh$ is the hyperbolic tangent non-linearity, and $g \in \mathcal{R}^{m\times m}$ is a diagonal matrix of *gain* coefficients allowing the outputs of F to compensate for the scaling of the input and the limited range of the hyperbolic tangent non-linearity. For convenience, W is used to collectively denote the parameters of the predictor, $W = \{g, M, b\}$.

During training, the goal is to make the prediction of the regressor, $F(x;W)$ as close as possible to the optimal set of coefficients: $z^* = \arg\min_z \mathcal{L}_I$ in Eqn. (2). This optimization can be carried out separately *after* the problem in Eqn. (2) has been solved. However, training becomes much faster by *jointly* optimizing the W and the set of basis functions $\mathcal{D}$ all together. This is achieved by adding another term to the loss function in Eqn. (2), which forces the representation z to be as close as possible to the feed-forward prediction $F(x;W)$:

$$\mathcal{L}_{IP} = \|x - \mathcal{D}z\|_2^2 + \lambda \sum_{i=1}^{K} \sqrt{\sum_{j\in P_i} w_j z_j^2} + \alpha\|z - F(x;W)\|_2^2 \tag{4}$$

The overall structure of this loss function is depicted in Fig. 1(b).

2.4. Learning

The goal of learning is to find the optimal value of the basis functions $\mathcal{D}$, as well as the value of the parameters in the regressor W, thus minimizing $\mathcal{L}_{IP}$ in Eqn. 4. Learning proceeds by an on-line block coordinate gradient descent algorithm, alternating the following two steps for each training sample x.

1. Keeping the parameters W and $\mathcal{D}$ constant, minimize $\mathcal{L}_{IP}$ of Eqn. (4) with respect to z, starting from the initial value provided by the regressor $F(x; W)$.

2. Using the optimal value of the coefficients z provided by the previous step, update the parameters W and $\mathcal{D}$ by one step of stochastic gradient descent. The update is: $U \leftarrow U - \eta\frac{\partial \mathcal{L}_{IP}}{\partial U}$, where U collectively denotes $\{W, \mathcal{D}\}$ and η is the step size. The columns of $\mathcal{D}$ are then re-scaled to unit norm.

We set $\alpha = 1$ for all experiments. We found that training the set of basis functions $\mathcal{D}$ first, then subsequently training the regressor, yields similar performance in terms of recognition accuracy. However, when the regressor is trained afterwards, the approximate representation is usually less sparse and the overall training time is considerably longer.

2.5. Evaluation

Once the parameters are learned, computing the invariant representation v can be performed by a simple feed-forward propagation through the regressor $F(x; W)$, and then propagating z into v through $v_i = \sqrt{\sum_{j \in P_i} w_j z_j^2}$. Note that no reconstruction of x using the set of basis functions $\mathcal{D}$ is necessary any longer. An example of this feed forward recognition architecture is given in Fig. 6.

The addition of this feed-forward module for predicting z, and hence, v is crucial to speeding up the run-time performance, since no optimization needs to be run after training. Experiments reported in a technical report on the non-invariant version of Predictive Sparse Decomposition [9] show that the z produced by this approximate representation gives a slightly superior recognition accuracy to the z produced by optimizing of $\mathcal{L}_I$.

Finally, other families of regressor functions were tested (using different kinds of thresholding non-linearities), but the one chosen here achieves similar performance while having the advantage of being very simple. In fact the filters M learned by the prediction function closely match the basis functions $\mathcal{D}$ used for reconstruction during training, having the same topographic layout.

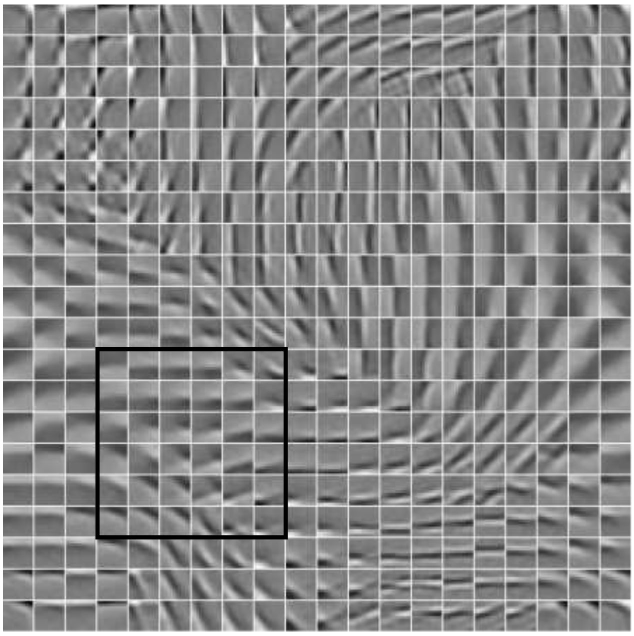

Figure 2. Topographic map of feature detectors learned from natural image patches of size 12x12 pixels by optimizing $\mathcal{L}_{IP}a$ in Eqn. 4. There are 400 filters that are organized in 6x6 neighborhoods. Adjacent neighborhoods overlap by 4 pixels both horizontally and vertically. Notice the smooth variation within a given neighborhood and also the circular boundary conditions.

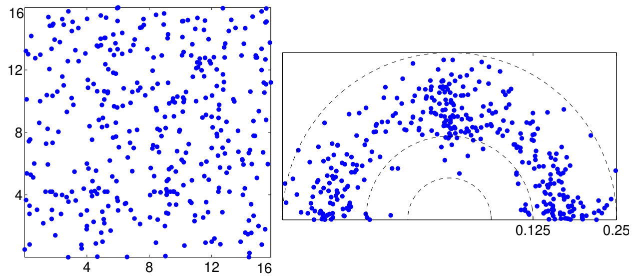

Figure 3. Analysis of learned filters by fitting Gabor functions, each dot corresponding to a feature. Left: Center location of fitted Gabor. Right: Polar map showing the joint distribution of orientation (azimuthally) and frequency (radially) of Gabor fit.

3. Experiments

In the following section, before exploring the properties of the invariant features obtained, we first study the topographic map produced by our training scheme. First, we make an empirical evaluation of the invariance achieved by these representations under translations and rotations of the input image. Second, we assess the discriminative power of these invariant representations on recognition tasks in three different domains: (i) generic object categories using the Caltech 101 dataset; (ii) generic object categories from a dataset of very low resolution images and (iii) classification

Figure 4. Examples from the tiny images. We use grayscale images in our experiments.

of handwriting digits using the MNIST dataset. In these experiments we compare IPSD 's learned representations with the SIFT descriptor [15] that is considered a state-of-the-art descriptor in computer vision. Finally, we examine the computational cost of computing IPSD features on an image.

3.1. Learning the Topographic Map

Fig. 2 shows a typical topographic map learned by the proposed method from natural image patches. Each tile shows a filter in $\mathcal{D}$ corresponding to a particular z_i. In the example shown, the input images are patches of size 12x12 pixels, and there are 400 basis functions, and hence, 400 units z_i arranged in a 20x20 lattice. The neighborhoods over which the squared activities of z_i's are pooled are 6x6 windows, and they overlap by 4 in both the vertical and the horizontal direction. The properties of these filters are analyzed by fitting Gabor functions and are shown in Fig. 3.

By varying the way in which the neighborhoods are pooled, we can change the properties of the map. Larger neighborhoods make the filters in each pool increasingly similar. A larger overlap between windows makes the filters vary more smoothly across different pools. A large sparsity value λ makes the feature detectors learn less localized patterns that look like those produced by k-means clustering because the input has to be reconstructed using a small number of basis functions. On the other hand, a small sparsity value makes the feature detectors look like non-localized random filters because any random overcomplete basis set can produce good reconstructions (effectively, the first term in the loss of Eqn. 4 dominates).

The map in Fig. 2 has been produced with an intermediate sparsity level of $\lambda = 3$. The chosen parameter setting induces the learning algorithm to produce a smooth map with mostly localized edge detectors in different positions, orientations, and scales. These feature detectors are nicely organized in such a way that neighboring units encode similar patterns. A unit v_i, that connects to the sum of the squares of units z_j in a pool is invariant because these units represent similar features, and small distortions applied to the input, while slightly changing the z_j's within a pool, are likely to leave the corresponding v_i unaffected.

While the sparsity level, the size of the pooling windows and their overlap should be set by cross-validation, in practice we found that their exact values do not significantly affect the kind of features learned. In other words, the algorithm is quite robust to the choice of these parameters, probably because of the many constraints enforced during learning.

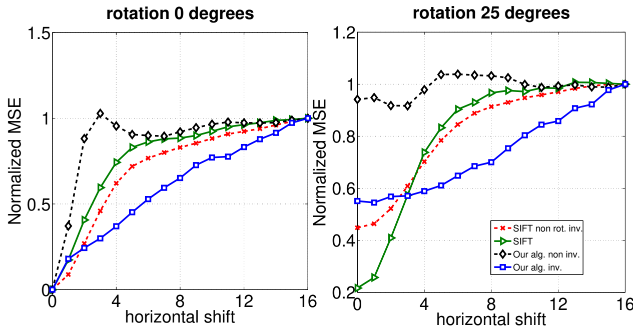

Figure 5. Mean squared error (MSE) between the representation of a patch and its transformed version. On the left panel, the transformed patch is horizontally shifted. On the right panel, the transformed patch is first rotated by 25 degrees and then horizontally shifted. The curves are an average over 100 patches randomly picked from natural images. Since the patches are 16x16 pixels in size, a shift of 16 pixels generates a transformed patch that is quite uncorrelated to the original patch. Hence, the MSE has been normalized so that the MSE at 16 pixels is the same for all methods. This allows us to directly compare different feature extraction algorithms: non-orientation invariant SIFT, SIFT, IPSD trained to produce non-invariant representations (i.e. pools have size 1x1) [9], and IPSD trained to produce invariant representations. All algorithms produce a feature vector with 128 dimensions. IPSD produces representations that are more invariant to transformations than the other approaches.

3.2. Analyzing Invariance to Transformations

In this experiment we study the invariance properties of the learned representation under simple transformations. We have generated a dataset of 16x16 natural image patches under different translations and rotations. Each patch is presented to the predictor function that produces a 128 dimensional descriptor (chosen to be the same size as SIFT) composed of v's. A representation can be considered locally invariant if it does not change significantly under small transformations of the input. Indeed, this is what we observe in Fig. 5. We compare the mean squared difference between the descriptor of the reference patch and the descriptor of the transformed version, averaged over many different image patches. The figure compares proposed descriptor against SIFT with a varying horizontal shift for 0 and 25 degrees initial rotation. Very similar results are found for vertical shifts and other rotation angles.

On the left panel, we can see that the mean squared error (MSE) between the representation of the original patch and its transformation increases linearly as we increase the horizontal shift. The MSE of IPSD representation is generally

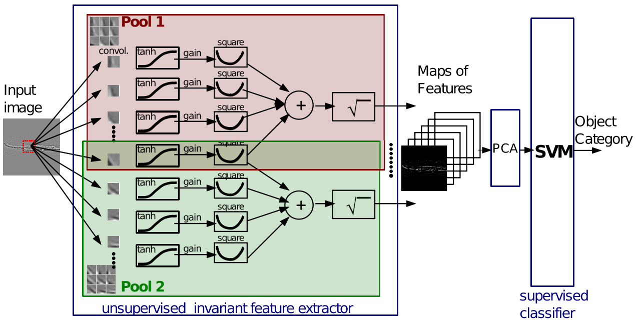

Figure 6. Diagram of the recognition system, which is composed of an invariant feature extractor that has been trained unsupervised, followed by a supervised linear SVM classifier. The feature extractor process the input image through a set of filter banks, where the filters are organized in a two dimensional topographic map. The map defines pools of similar feature detectors whose activations are first non-linearly transformed by a hyperbolic tangent non-linearity, and then, multiplied by a gain. Invariant representations are found by taking the square root of the sum of the squares of those units that belong to the same pool. The output of the feature extractor is a set of feature maps that can be fed as input to the classifier. The filter banks and the set of gains is learned by the algorithm. Recognition is very fast, because it consists of a direct forward propagation through the system.

lower than the MSE produced by features that are computed using SIFT, a non-rotation invariant version of SIFT, and a non-invariant representation produced by the proposed method (that was trained with pools of size 1x1) [9]. A similar behavior is found when the patch is not only shifted, but also rotated. When the shift is small, SIFT has lower MSE, but as soon as the translation becomes large enough that the input pattern falls in a different internal sub-window, the MSE increases considerably. Instead learned representations using IPSD seem to be quite robust to shifts, with an overall lower area under the curve. Note also that traditional sparse coding algorithms are prone to produce unstable representations under small distortions of the input. Because each input has to be encoded with a small number of basis functions, and because the basis functions are highly tuned in orientation and location, a small change in the input can produce drastic changes in the representation. This problem is partly alleviated by our approximate inference procedure that uses a smooth predictor function. However, this experiment shows that this representations is fairly unstable under small distortions, when compared to the invariant representations produced by IPSD and SIFT.

3.3. Generic Object Recognition

We now use IPSD invariant features for object classification on the Caltech 101 dataset [5] of 102 generic object categories including background class. We use 30 training images per class and up to 30 test images per class. The images are randomly picked, and pre-processed in the following way: converted to gray-scale and down-sampled in such a way that the longest side is 151 pixels and then locally normalized and zero padded to 143x143 pixels. The local normalization takes a 3x3 neighborhood around each pixel, subtracts the local mean, then divides the by the local standard deviation if it is greater than the standard deviation of the image. The latter step is a form of divisive normalization, proposed to model the contrast normalization in the retina [21].

We have trained IPSD on 50,000 16x16 patches randomly extracted from the pre-processed images. The topographic map used has size 32x16, with the pooling neighborhoods being 6x6 and an overlap of 4 coefficients between neighborhoods. Hence, there are a total of 512 units that are used in 128 pools to produce a 128-dimensional representation that can be compared to SIFT. After training IPSD in an unsupervised way, we use the predictor function to infer the representation of one whole image by: (i) running the predictor function on 16x16 patches spaced by 4 pixels to produce 128 maps of features of size 32x32; (ii) the feature maps are locally normalized (neighborhood of 5x5) and low-pass filtered with a boxcar filter (5x5) to avoid aliasing; (iii) the maps are then projected along the leading 3060 principal components (equal to the number of training samples), and (iv) a supervised linear SVM[1] is trained to recognize the object in each corresponding image. The overall scheme is shown in Fig. 6.

Table 1 reports the recognition results for this experiment. With a linear classifier similar to [21], IPSD features outperform SIFT and the model proposed by Serre and Poggio [25]. However, if rotation invariance is removed from SIFT its performance becomes comparable to IPSD .

We have also experimented with the more sophisticated Spatial Pyramid Matching (SPM) Kernel SVM classifier of Lazebnik et al. [11]. In this experiment, we again used the same IPSD architecture on 16x16 patches spaced by 3 pixels to produce 42x42x128 dimensional feature maps, followed by local normalization over a 9x9 neighborhood, yielding 128 dimensional features over a uniform 34x34 grid. Using SPM, IPSD features achieve 59.6% average accuracy per class. By decreasing the stepping stride to 1 pixel, thereby producing 120x120 feature maps, IPSD features achieve 65.5% accuracy as shown in Table 1. This is comparable to Lazebnik's 64.6% accuracy on Caltech-101 (without background class) [11]. For comparison, our re-implementation of Lazebnik's SIFT feature extractor, stepped by 4 pixels to produce 34x34 maps, yielded 65% average recognition rate.

With 128 invariant features, each descriptor takes around 4ms to compute from a 16x16 patch. Note that the evaluation time of each region is a linear function of the number

[1]We have used LIBSVM package available at http://www.csie.ntu.edu.tw/ cjlin/libsvm

Method	Av. Accuracy/Class (%)
local norm$_{5\times5}$ + boxcar$_{5\times5}$ + PCA$_{3060}$ + linear SVM	
IPSD (24x24)	**50.9**
SIFT (24x24) (non rot. inv.)	**51.2**
SIFT (24x24) (rot. inv.)	**45.2**
Serre et al. features [25]	**47.1**
local norm$_{9\times9}$ + Spatial Pyramid Match Kernel SVM	
SIFT [11]	**64.6**
IPSD (34x34)	**59.6**
IPSD (56x56)	**62.6**
IPSD (120x120)	**65.5**

Table 1. Recognition accuracy on Caltech 101 dataset using a variety of different feature representations and two different classifiers. The PCA + linear SVM classifier is similar to [21], while the Spatial Pyramid Matching Kernel SVM classifier is that of [11]. IPSD is used to extract features with three different sampling step sizes over an input image to produce 34x34, 56x56 and 120x120 feature maps, where each feature is 128 dimensional to be comparable to SIFT. Local normalization is **not** applied on SIFT features when used with Spatial Pyramid Match Kernel SVM.

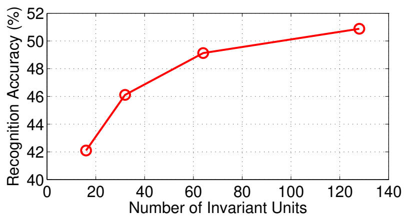

Figure 7. The figure shows the recognition accuracy on Caltech 101 dataset as a function of the number of invariant units. Note that the performance improvement between 64 and 128 units is below 2%, suggesting that for certain applications the more compact descriptor might be preferable.

of features, thus this time can be further reduced if the number of features is reduced. Fig. 7 shows how the recognition performance varies as the number of features is decreased.

3.4. Tiny Images classification

IPSD was compared to SIFT on another recognition task using the tiny images dataset [26]. This dataset was chosen as its extreme low-resolution provides a different setting to the Caltech 101 images. For simplicity, we selected 5 animal nouns (abyssinian cat, angel shark, apatura iris (a type of butterfly), bilby (a type of marsupial), may beetle) and manually labeled 200 examples of each. 160 images of each class were used for training, with the remaining 40 held out for testing. All images are converted to grayscale. Both IPSD with 128 pooled units and SIFT were used to extract features over 16x16 regions, spaced every 4 pixels over the 32x32 images. The resulting 5 by 5 by 128 dimensional feature maps are used with a linear SVM. IPSD features achieve 54% and SIFT features achieve a comparable 53%.

Performance on Tiny Images Dataset	
Method	**Accuracy** (%)
IPSD (5x5)	**54**
SIFT (5x5) (non rot. inv.)	**53**
Performance on MNIST Dataset	
Method	**Error Rate** (%)
IPSD (5x5)	**1.0**
SIFT (5x5) (non rot. inv.)	**1.5**

Table 2. Results of recognition error rate on Tiny Images and MNIST datasets. In both setups, a 128 dimensional feature vector is obtained using either IPSD or SIFT over a regularly spaced 5x5 grid and afterwards a linear SVM is used for classification. For comparison purposes it is worth mentioning that a Gaussian SVM trained on MNIST images without any preprocessing achieves 1.4% error rate.

3.5. Handwriting Recognition

We use a very similar architecture to that used in the experiments above to train on the handwritten digits of the MNIST dataset [1]. This is a dataset of quasi-binary handwritten digits with 60,000 images in the training set, and 10,000 images in the test set. The algorithm was trained using 16x16 windows extracted from the original 28x28 pixel images. For recognition, 128-dimensional feature vectors were extracted at 25 locations regularly spaced over a 5x5 grid. A linear SVM trained on these features yields an error rate of **1.0%**. When 25 SIFT feature vectors are used instead of IPSD features, the error rate increases to **1.5%**. This demonstrates that, while SIFT seems well suited to natural images, IPSD produces features that can adept to the task at hand. In a similar experiment, a *single* 128-dimensional feature vector was extracted using IPSD and SIFT, and fed to a linear SVM. The error rate was 5.6% for IPSD , and 6.4% for SIFT.

4. Summary and Future Work

We presented an architecture and a learning algorithm that can learn locally-invariant feature descriptors. The architecture uses a bank of non-linear filters whose outputs are organized in a topographic fashion, followed by a pooling layer that imposes a sparsity criterion on blocks of filter outputs located within small regions of the topographic map. As a result of learning, filters that are pooled together extract similar features, which results in spontaneous invariance of the pooled outputs to small distortions of the input. During training, the output of the non-linear filter bank is fed to a linear decoder that reconstructs the input patch. The filters and the linear decoder are simultaneously trained to minimize the reconstruction error, together with a sparsity criterion computed as the sum of the pooling units. After training, the linear decoder is discarded, and the pooling unit outputs are used as the invariant feature descriptor of the input patch. Computing the descriptor for a patch is very fast and simple: it merely involves multiplying the patch by a filtering matrix, applying a scaled *tanh* function to the results, and computing the square root of Gaussian-weighted sum-of-squares of filter outputs within each pool window.

Image classification experiments show that the descriptors thereby obtained give comparable performance to SIFT descriptors on tasks for which SIFT was specifically designed (such as Caltech 101), and better performance on tasks for which SIFT is not particularly well suited (MNIST, and Tiny Images).

While other models have learned locally invariant descriptors by explicitly building shift invariance using spatial pooling, our proposal is more general: it can learn local invariances to other transformations than just translations. Our results also show spontaneous local invariance to rotation. To our knowledge, this is the first time such invariant feature descriptors have been learned and tested in an image recognition context with competitive recognition rates.

A long-term goal of this work is to provide a general tool for learning feature descriptors in an unsupervised manner. Future work will involve "stacking" multiple stage of such feature extractors so as to learn multi-level hierarchies of increasingly global and invariant features.

5. Acknowledgments

We thank Karol Gregor, Y-Lan Boureau, Eero Simoncelli, and members of the CIfAR program Neural Computation and Adaptive Perception for helpful discussions. This work was supported in part by ONR grant N00014-07-1-0535, NSF grant EFRI-0835878, and NSF IIS-0535166.

References

[1] http://yann.lecun.com/exdb/mnist/.

[2] M. Aharon, M. Elad, and A. Bruckstein. K-svd and its non-negative variant for dictionary design. volume 5914, 2005.

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. of Computer Vision and Pattern Recognition*, 2005.

[4] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression, 2002,.

[5] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *CVPR Workshop*, 2004.

[6] A. Hyvarinen and P. Hoyer. Emergence of phase- and shift-invariant features by decomposition of natural images into independent feature subspaces. *Neural Comput*, 12(7):1705–1720, 2000 Jul.

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

[8] A. Hyvarinen and U. Koster. Complex cell pooling and the statistics of natural images. *Network*, 18(2):81–100, 2007 Jun.

[9] K. Kavukcuoglu, M. Ranzato, and Y. LeCun. Fast inference in sparse coding algorithms with applications to object recognition. Technical report, CBLL, Courant Institute, NYU, 2008. CBLL-TR-2008-12-01.

[10] T. Kohonen. Emergence of invariant-feature detectors in the adaptive-subspace self-organizing map. *Biol. Cybern.*, 75:281–291, 1996.

[11] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, pages 2169–2178. IEEE, June 2006.

[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[13] Y. LeCun, F.-J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of CVPR'04*. IEEE Press, 2004.

[14] H. Lee, A. Battle, R. Raina, and A. Ng. Efficient sparse coding algorithms. In *NIPS*, 2006.

[15] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[16] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Discriminative learned dictionaries for local image analysis. In *CVPR*, 2008.

[17] J. Murray and K. Kreutz-Delgado. Learning sparse overcomplete codes for images. *The Journal of VLSI Signal Processing*, 45:97–110, 2008.

[18] J. Mutch and D. Lowe. Multiclass object recognition with sparse, localized features. In *CVPR*, 2006.

[19] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: a strategy employed by v1? *Vision Research*, 37:3311–3325, 1997.

[20] S. Osindero, M. Welling, and G. E. Hinton. Topographic product models applied to natural scene statistics. *Neural Comput*, 18(2):381–414, 2006 Feb.

[21] N. Pinto, D. D. Cox, and J. J. DiCarlo. Why is real-world visual object recognition hard? *PLoS Computational Biology*, 4(1), 2008.

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

[23] M. Ranzato, C. Poultney, S. Chopra, and Y. LeCun. Efficient learning of sparse representations with an energy-based model. In *NIPS*, 2006.

[24] C. Rozell, D. Johnson, B. R.G., and B. Olshausen. Sparse coding via thresholding and local competition in neural circuits. *Neural Computation*, 2008.

[25] T. Serre, L. Wolf, and T. Poggio. Object recognition with features inspired by visual cortex. In *CVPR*, 2005.

[26] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: A large dataset for nonparametric object and scene recognition. *IEEE PAMI*, 30(11):1958–1970, 2008.

[27] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. Technical report, Univ. of Winsconsin, 2004.