Looking around the Backyard Helps to Recognize Handwritten Digits

Anonymous CVPR submission

Paper ID 2611

Abstract

*Human being has the ability to learn to recognize a new visual category based on only one or few training examples. Part of this ability might come from the use of knowledge from previous visual experiences. We show that such knowledge can be expressed as a set of “universal” visual features, which are learned from randomly collected natural scene images. Using these visual features, we have obtained state-of-the-art performance on several classification tasks.*

1. Introduction

One notable difference between the human visual system and most computer vision systems is that we human beings can learn a new visual category based on few or even one example of the new category, while most computer systems require a large number of training samples to work reasonably well. For example, we can recognize the euro symbol (shown in Figure 1) under a wide variety of visual conditions after having seen a single example. Such an ability is necessary for the wide application of computer vision, as collecting training examples is generally expensive. In fact, due to the curse of dimensionality, almost all the real-world pattern recognition problems can be regarded as problems of learning with few examples.

![Figure 1. A handwritten euro (the European currency) symbol.](image)

The problem of learning with scarce training samples has been addressed in different ways. Semi-supervised learning algorithms [6] deal with the situation when labeled training samples are scarce by utilizing information from unlabeled samples. It alleviates the pain of building a working system, since generally unlabeled data abound and are much cheaper to collect. It does not help, however, when unlabeled samples are also rare or expensive to collect, whereas human beings still exhibit reasonable performance in these cases. Miller et al. [13] adopted an alternative approach to address this problem. They learned a set of transforms from handwritten English letters and then applied them to handwritten digits to regularize the input patterns, which allowed them to build a handwritten digit classifier using only several training examples per class. The limitations of this method, however, are that the forms of the transforms need to be predefined and that the visual knowledge can only be shared between highly similar visual categories.

These two types of techniques share one common characteristic—they both use knowledge obtained from a second information source to help building the classifier. They differ in how they define the knowledge and how the knowledge is learned and transferred, but they both pick a secondary information source which share some common property with the labeled data at hand, either unlabeled data belonging to the same category with the labeled data, or labeled data belonging to a different but visually similar category, so that a knowledge transfer is plausible.

The human visual system seems to adopt a similar approach. Low-level visual layers, such as retina, LGN (the lateral geniculate nucleus) and V1 (the primary visual cortex), are shared components that process all visual information we perceive. These layers develop and mature during the childhood, and provide the basis for all the visual tasks encountered in the rest of the life. If we loosely define the term “visual feature” as any function of the image pixel values, the above phenomenon can be interpreted as learning a set of *universal* visual features from the scenes encountered during the childhood. These visual features are later applied to various visual stimuli and help to fulfill all kinds of visual tasks. Presumably these visual features provide one way to transfer the knowledge obtained from previous visual experiences and should help to build a classifier when the labeled data is rare.

The idea about the “universal visual features” might at first sight appear astonishing, since this concept suggests that all visual stimuli share some characteristic in common
such that knowledge obtained from one set of stimuli can be applied to a completely different set of visual stimuli. What is the common property shared by the appearance of your husband/wife’s face and the sight of your backyard that would allow you to better recognize the first by simply browsing the latter?

One observation is that they share similar local statistical structures. For example, if we take all the 255,025 $8 \times 8$ image patches from each of the two images shown on the top row of Figure 1, subtract the local mean from each image patch, and apply PCA on them, the resulting basis functions (i.e., eigenvectors) all resemble the DCT filters. In fact, if we apply the PCA projection learned from image patches extracted from one image to the image patches from the other, and calculate the covariance matrix of the projected features, we will see that most off-diagonal cells have much smaller values than the diagonal cells. That is, although the projection matrix is calculated to capture the second-order local structure of one image, it also approximately captures the second-order local structure of the other image. The observation here, is that although the two images display very different visual content, they share very similar local/lower-level statistical structures.

This is plausible because the visual stimuli we perceive share similar local statistical structures. And presumably, these visual features will be especially helpful in building a classifier with few training examples, because by doing so, we implicitly exploit the visual knowledge obtained from previous visual experiences.

The following sections are contributed to instantiating these ideas. We briefly review neuroscience theories about low-level human vision in Section 2, mostly from a computational perspective. Then we show in Section 3 how to learn a set of visual features from ten randomly collected natural images by simulating visual information processing up to the simple cells, the first layer of visual information processing in the cerebral cortex. With images being represented by these visual features, we have achieved recognition performance comparable to recently proposed computer vision techniques, even if we just use a linear classifier. In the same section, we also show through an example when the technique will seriously fail, as well as a solution to deal with such cases. We end this paper by discussing possible approaches to further improve this technique.

2. Theory of Low-level Human Vision

2.1. The Efficient Coding Hypothesis

What is the utility of unsupervised visual feature extraction from images? A similar question has been considered in the neuroscience community for years: what is the functional role of low-level visual layers in the human visual pathway which appear to receive little top-down (i.e. task driven) influence?

One hypothesis is that they capture the statistical structures of sensory inputs so that corresponding high-level decisions can be made accordingly (see [3] for a brief review). This hypothesis engenders two questions: (1) how to (quantitatively) define the statistical structure; and (2) how to capture the statistical structure. In 1954, Attenave [1] pointed out that whether we perceive structures in an image depends on how well we can predict a missing part of the image by its remaining parts. This insight suggests that we can use the dependency among the input features (i.e. dimensions), or the redundancy of the inputs, as a quantitative measurement of the statistical structures provided by the sensory inputs. Based on this observation, Barlow [2] stated that one plausible way for a neural system to capture the statistical structure of its inputs was to remove the redundancy in its outputs, because to do so, the neural system must have a complete knowledge about statistical structures contained in its inputs. This hypothesis is later referred to as the redundancy reduction principle or the efficient coding principle. Although the name appears to suggest pursuing an economic coding, its essence is still about capturing the statistical structure of the sensory inputs.

Figure 2. Applying PCA on image patches. For each $512 \times 512$ image on the top row, we sample 255,025 $8 \times 8$ image patches and apply PCA on them. The top 49 eigenvectors with the largest eigenvalues are shown at the bottom row.

It is not a coincidence that the most popular theory about low-level human vision happens to suggest that the functional role of low-level vision is to capture the statistical structure of the visual sensory inputs. Combining all the facts together, it appears that the human visual system learns a set of universal visual features during development, and uses these features for all the visual tasks encountered later.
2.2. Linear Efficient Coding

Linear implementations of the efficient coding hypothesis, such as independent component analysis [5] and sparse coding [14], have been used to explain the functional role of simple cells in the primary visual cortex, the first layer of visual information processing in the cerebral cortex. These algorithms are best described by a generative model, in which the observed data $\mathbf{x} \in \mathbb{R}^d$ is assumed to be generated by linearly mixing underlying signal source $\mathbf{s} \in \mathbb{R}^D$:

$$\mathbf{x} = A\mathbf{s} + \mathbf{\epsilon}$$  \hspace{1cm} (1)

where $A \in \mathbb{R}^{d \times D}$ is the linear mixing matrix, $\mathbf{\epsilon} \in \mathbb{R}^d$ denotes additive gaussian noises. The signal sources $s_j$'s are assumed to be statistically independent, which incorporates the desire of capturing statistical structures in $\mathbf{x}$. For natural image statistic studies, a sparse marginal distribution for each $s_j$ is assumed, which is characterized by a peak at zero and two heavy tails symmetrically residing on both sides of zero, such as the Student-t distribution or the Laplacian distribution. It was argued [14] that such a distribution incorporates the need to transfer more information with minimum energy cost, which is very important for a biological system.

There are two optimization problems associated with this model. One is to learn the most probable $A$ given $n$ observations $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ (see [12] on the learning algorithm). Once the optimal $A$ is learned, the inference problem aims to infer the most probable signal source $\mathbf{s}$ given the mixing matrix $A$ and one observation $\mathbf{x}$. When the marginal distribution is assumed to be a Laplacian distribution, the inference problem is a convex optimization problem and can be solved efficiently.

If we apply the linear efficient coding algorithms on natural image patches, the resulting basis functions (i.e., columns of $A$) resemble the simple cells receptive fields, as shown in Figure 4. It is widely observed [5, 11, 14] that the basis functions learned from different image datasets are qualitatively similar, which suggests even though these images display very different global contents they share similar local statistical structures.

2.3. Before Linear Efficient Coding

The visual information passes through retina and LGN before reaching the primary visual cortex. What happens there and why?

The classical theory about what happens before V1 is the whitening theory [8], which states that retina and LGN serve to flatten the magnitudes of the images on the frequency domain. It was observed that natural images approximately follow the $1/f$ law in the frequency domain. That is, if we apply 2D Fourier transform on a natural image, most of the time we will observe that the magnitude of each component decreases with the increase of the frequency. Supposedly the visual system removes the redundant information by flattening the magnitudes on the frequency domains. Later it was pointed out that this operation also approximately makes pixel values uncorrelated [9]. A recent paper [17] also noted that this operation regulates the distribution of the inputs to V1 so that the linear efficient coding algorithm could work more efficiently.

In Section 3, we will show the classification performance with and without this whitening operation before linear efficient coding.

2.4. After Linear Efficient Coding

Recently exploring the functional role of higher visual layers in the human visual pathway (i.e., beyond simple cells) has become a hot topic. We have found one recent work [17] pretty interesting and the algorithm proposed in that paper greatly improves classification performance in our experiments.

Their work was based on a well-known observation that different parts of the cerebral cortex share similar anatomical structure. They hypothesized that higher visual layers might also be working under similar computational principles as the primary visual cortex. They proposed a hierarchical model to study natural image statistics, in which a second layer of linear efficient coding (denoted as layer-2) was put on top of the first layer (layer-1) of linear efficient coding model. They applied their model on natural image patches and the layer-2 basis functions seemed to correspond to texture boundary and corner detectors.

To allow the second layer of linear efficient coding work efficiently, they derived a coordinate-wise nonlinear activation function that transformed the layer-1 outputs as the layer-2 inputs. For each dimension of the layer-1 outputs $s_j$, they first took its absolute value $\|s_j\|_1$. They estimated the cdf function $\mathcal{F}$ of $\|s_j\|_1$, and the coordinate-wise activation function $G$ was defined as:

$$G(s_j) = \mathcal{G}(\mathcal{F}(\|s_j\|_1))$$  \hspace{1cm} (2)

where $\mathcal{G}$ denotes the inverse cdf function of a standard normal distribution.

Here is an intuitive explanation of their activation functions. The layer-1 basis function can be roughly considered as edge/bar detectors. Taking the absolute values from a small range on the $y$-axis. This segment serves to increase the distance between two $\|s_j\|$ values and may help the classifier to distinguish their differences. When the $\|s_j\|$ value is bigger than where the red dot indicates, the...
activation function serves the stretch the distance between $s_j$‘s.

3. Practices in Computer Vision

3.1. Learning Visual Features

We apply the sparse coding algorithm [15] on ten 512 × 512 natural images available from Olshausen’s homepage. The images are whitened using the whitening filter described in [14], whose matlab code is also available from Olshausen’s homepage. As discussed earlier, this whitening process is supposed to simulate the processing in retina and LGN. We normalize each image to have zero mean and unit variance. After whitening, six pixels off the boundary are discarded to avoid the boundary effect. So each image is now 500 × 500 in size. We extract all the 2,430,490 8 × 8 image patches, subtract the local mean from each image patch, then calculate the PCA projection matrix. Now each image patch is represented as a 63 dimensional vector $\vec{x}$. During the PCA projection, we make each dimension of $\vec{x}$ to have unit variance.

The sparse coding algorithm is applied on these image patches. It corresponds to the linear efficient coding model described in Equation 1, with the following marginal prior:

$$p(s_j) \propto \frac{1}{(1 + (s_j/\sigma))^\beta}$$

In our experiments, we set $\sigma = 1$ and $\beta = 0.4$. The variances of noises $\vec{e}$ (see Equation 1) are also set to 1.

We initialize the linear mixing matrix $A$ with gaussian random variables. Then in each batch, we randomly pick 100 image patches, calculate their PCA projections $s$.

Described in Equation 1, with the following marginal prior:

$$\mathcal{F}_s(||s_j||) \propto \Gamma(||s_j||/\tau)^0, 1/\theta$$

where $\theta$ denotes the incomplete Gamma function. Figure 5 displays the empirical cdf function of $\mathcal{F}_s$ as well as the fitted cdf function, when the dimensionality of $s$ equals 128.

3.2. Experiments on Yale Face Dataset

After the linear mixing matrix $A$ and the nonlinear activation functions $G_j = \mathcal{G}(\mathcal{F}_j)$ are learned from the ten natural images, we apply them on various classification tasks without any adjustment of the parameters. And it turns
out that classification performance based on these features outperforms many recently proposed computer vision techniques, even if we just use a linear classifier.

We first test the features on the Yale dataset [4], which contains 165 gray-scale images of 15 individuals. Each individual has 11 images. The manually aligned and cropped images can be downloaded from the homepage of the first author of reference [7]. We downloaded the $64 \times 64$ processed images, and then downsampled the images to $32 \times 32$ using the imresize matlab function. Then we whitened each image using the whitening filter discussed in Section 3.1 and normalize each image to have zero mean and unit variance. For each image, we extract all the $625 \times 8 \times 8$ image patches, and infer the most probable $\hat{s}$ for each image patch. After that, we apply the nonlinear activation $G_j$ on each dimension of $s_j$. When the dimensionality of $\hat{s}$ is 64 (1 times overcomplete), each facial image is represented by a $625 \times 64 = 40,000$ dimensional vector. When it comes to 2 times over completeness, each image is represented by a $80,000$ dimensional vector.

We followed the method in [7] to randomly divide the images into the training and the testing sets. For each experiment, we randomly select $M = 2, 3, \ldots, 8$ images from each individual as the training images, and use the rest images as testing images. For each number of $M$, we tested 50 random splits.

When the training set and the testing set are selected, we calculate the PCA projection based on the training examples. The number of principal components are chosen so that 95% of the variances are captured. Then we use the projected training examples to train a one layer network using the softmax activation function. The network is updated for 1,000 epochs or until the network weights have converged, using the scaled conjugate gradient algorithm (the glm + netopt function in the netlib library implement this algorithm). The test images are projected using the PCA projection matrix trained on the training images, and fed to the one layer network.

Table 1 lists the recognition performance on the test images, with different overcompleteness and different numbers of training images from each individual. The results are extremely good given the small number of training examples. For example, in CVPR2007, the best result reported in [7] using 2, 3, 4, 5 training examples per individual was 42.4%, 27.7%, 22.2%, 18.3% on the same dataset. In another paper [10] from the same conference, the authors reported a 13.2% error rate using 5 training examples per individual. We noticed that on the homepage of the first author of reference [7], they reported new results with newly tuned parameters: 37.5%, 25.5%, 19.3%, 14.7%, 12.3%, 10.3%, 8.7%, using 2, 3, \ldots, 8 training examples per class. However, there is still an obvious gap between their results and our results. And our results are achieved without tuning any parameters.

### 3.3. Experiments on ORL Face Dataset

The Olivetti Research Laboratory (ORL) database[16] contains 400 face images of 40 persons, with 10 per person, taken at different time, under different lighting conditions, and with different facial expressions. We download the manually aligned and cropped $64 \times 64$ images from the homepage of the first author of reference [7], and then downsampled them to $32 \times 32$ size using the imresize matlab function. We implemented the same experiments as on the Yale datasets. We randomly select $M = 2, 3, \ldots, 8$ images per person for training and the rest for testing. The average recognition error rates are reported in Table 2.

The authors of [10] reported their best result as 3.0% with 5 training examples per person. The authors of [7] reported the error rates 14.8%, 7.7%, 4.2%, 2.8% with 2, 3, 4, 5 training examples per person. Our results are comparable to their results. Still, the results in Table 2 are achieved without tuning any parameters.

### Table 1. Recognition error rates (in percentage) on the Yale dataset

<table>
<thead>
<tr>
<th>M</th>
<th>Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>28.86</td>
</tr>
<tr>
<td>3</td>
<td>19.55</td>
</tr>
<tr>
<td>4</td>
<td>13.71</td>
</tr>
<tr>
<td>5</td>
<td>9.96</td>
</tr>
<tr>
<td>6</td>
<td>7.71</td>
</tr>
<tr>
<td>7</td>
<td>6.43</td>
</tr>
<tr>
<td>8</td>
<td>4.93</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>M</th>
<th>Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>27.13</td>
</tr>
<tr>
<td>192</td>
<td>27.11</td>
</tr>
</tbody>
</table>

The number of principal components are chosen so that 95% of the variances are captured. Then we use the projected training examples to train a one layer network using the softmax activation function. The network is updated for 1,000 epochs or until the network weights have converged, using the scaled conjugate gradient algorithm (the glm + netopt function in the netlib library implement this algorithm). The test images are projected using the PCA projection matrix trained on the training images, and fed to the one layer network.

Table 1 lists the recognition performance on the test images, with different overcompleteness and different numbers of training images from each individual. The results are extremely good given the small number of training examples. For example, in CVPR2007, the best result reported in [7] using 2, 3, 4, 5 training examples per individual was 42.4%, 27.7%, 22.2%, 18.3% on the same dataset. In another paper [10] from the same conference, the authors reported a 13.2% error rate using 5 training examples per individual. We noticed that on the homepage of the first author of reference [7], they reported new results with newly tuned parameters: 37.5%, 25.5%, 19.3%, 14.7%, 12.3%, 10.3%, 8.7%, using 2, 3, \ldots, 8 training examples per class. However, there is still an obvious gap between their results and our results. And our results are achieved without tuning any parameters.

### Table 3. Recognition error rates (in percentage) on the ORL dataset

<table>
<thead>
<tr>
<th>M</th>
<th>Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>30.3%</td>
</tr>
<tr>
<td>3</td>
<td>12.3%</td>
</tr>
<tr>
<td>4</td>
<td>9.96%</td>
</tr>
<tr>
<td>5</td>
<td>7.71%</td>
</tr>
<tr>
<td>6</td>
<td>6.43%</td>
</tr>
<tr>
<td>7</td>
<td>4.93%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>M</th>
<th>Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>27.13%</td>
</tr>
<tr>
<td>192</td>
<td>27.11%</td>
</tr>
</tbody>
</table>
### 3.4. Experiments on MNIST Digit Dataset

In this experiment, we apply the features on the MNIST handwritten digits dataset. The MNIST dataset has a training set of 60,000 examples and a test set of 10,000 examples. The digits have been size-normalized and centered in a \(28 \times 28\) image. We downsampled each image to \(18 \times 18\) size by the imresize matlab function. Each image is whitened and normalized to have zero mean and unit variance. \(121\) \(8 \times 8\) image patches are extracted from each image.

We randomly select \(M = 10, 20, 30, 40, 50\) training examples for each digit:

<table>
<thead>
<tr>
<th>(D)</th>
<th>(M=2)</th>
<th>(M=3)</th>
<th>(M=4)</th>
<th>(M=5)</th>
<th>(M=6)</th>
<th>(M=7)</th>
<th>(M=8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>15.99</td>
<td>8.64</td>
<td>5.08</td>
<td>2.91</td>
<td>2.14</td>
<td>1.40</td>
<td>1.10</td>
</tr>
<tr>
<td>128</td>
<td>15.46</td>
<td>8.20</td>
<td>4.75</td>
<td>2.70</td>
<td>1.78</td>
<td>1.20</td>
<td>0.90</td>
</tr>
<tr>
<td>192</td>
<td>15.11</td>
<td>8.02</td>
<td>4.46</td>
<td>2.43</td>
<td>1.78</td>
<td>1.12</td>
<td>0.90</td>
</tr>
</tbody>
</table>

Table 2. Recognition error rates (in percentage) on the ORL dataset, using \(G(\|s\|)\) as features. \(D\) denotes the dimensionality of \(s\). \(M\) denotes the number of training images selected from each individual.

### 3.5. Why This Works, & When It Will Fail

Why unsupervised learning helps supervised learning? Or, why overcomplete sparse coding representation helps for classification tasks?

First, information of the original image is preserved in the sense that the original image can be reconstructed.

Second, the similarity structure is preserved.

\[
(As_1 - As_2)^T (As_1 - As_2) = (s_1 - s_2)^T A^T A (s_1 - s_2)
\]

However, the reverse is not true. Two very close points \(x_1\) and \(x_2\) might be mapped to two quite far away points in the high-dimensional space.

As discussed at the beginning of this paper, the whole idea about the universal visual features is based on the observation that visually quite different images share similar local statistical structures. Hence, the method will (greatly?) fail when the local statistics of the images under consideration is very different from the images on which the visual features are learned. This may occur when some artificial statistics is introduced during the image capturing process.

In some sense, this section is not about “when it (this technique) will fail”, it is rather about “when we (human beings) also will fail”.

How could the computer vision system learn to know when it should “step several steps back”?

This, however, raises an interesting question – how to avoid/remove artificial statistics introduced during image capturing?

### 4. Discussion

Linear efficient coding (ICA) has not interpreted in a right way in computer vision. first, FFA is a higher visual layer beyond V1, and we know little yet about the computational principle happening there. second, mathematically, complete ica after whitening is no more than a rotation. third, ica is itself a learning problem. \(300 \times 300 = 90,000\) parameters to learn, while with only \(300\) examples.

We feel that the right way to interpret ica is that it is trying to provide a set of universal visual features. This is plausible because globally different visual stimuli in fact share similar local statistical structures. This is beneficial because all we need is a bunch of natural images. These images do not need to be labeled. And unlike semisupervised learning or the method proposed in , the images do not need to share similar property as.

How to ensure the image sets have the same statistical properties? i.e., how to avoid introducing some artificial statistics? This is not a problem for human vision, because the sensor.

Will high overcompleteness help? Is this why we have 100 times overcompleteness in human vision V1? It was estimated that 100 times overcompleteness. Does this really help? Will the performance improvement stop after certain number of overcompleteness?

This does not provide shift/orientation invariance. Higher visual layers, or high-order image statistics. Caltech 101?

We hope our paper will advocate the interest from the computer vision toward human vision research.

works well for faces, cars, motobikes, template based. variations.... structural constraints, ...

This is

### References


[2] H. B. Barlow. Possible principles underlying the transformation of sensory messages. In W. A. Rosenblith, editor,


