Reconstructive Sparse Code Transfer for Contour Detection and Semantic Labeling

Michael Maire[1,2], Stella X. Yu[3], and Pietro Perona[2]

[1]TTI Chicago [2]California Institute of Technology
[3]University of California at Berkeley / ICSI
mmaire@ttic.edu, stellayu@berkeley.edu, perona@caltech.edu

Abstract. We frame the task of predicting a semantic labeling as a sparse reconstruction procedure that applies a target-specific learned transfer function to a generic deep sparse code representation of an image. This strategy partitions training into two distinct stages. First, in an unsupervised manner, we learn a set of dictionaries optimized for sparse coding of image patches. These generic dictionaries minimize error with respect to representing image appearance and are independent of any particular target task. We train a multilayer representation via recursive sparse dictionary learning on pooled codes output by earlier layers. Second, we encode all training images with the generic dictionaries and learn a transfer function that optimizes reconstruction of patches extracted from annotated ground-truth given the sparse codes of their corresponding image patches. At test time, we encode a novel image using the generic dictionaries and then reconstruct using the transfer function. The output reconstruction is a semantic labeling of the test image.

Applying this strategy to the task of contour detection, we demonstrate performance competitive with state-of-the-art systems. Unlike almost all prior work, our approach obviates the need for any form of hand-designed features or filters. Our model is entirely learned from image and ground-truth patches, with only patch sizes, dictionary sizes and sparsity levels, and depth of the network as chosen parameters. To illustrate the general applicability of our approach, we also show initial results on the task of semantic part labeling of human faces.

The effectiveness of our data-driven approach opens new avenues for research on deep sparse representations. Our classifiers utilize this representation in a novel manner. Rather than acting on nodes in the deepest layer, they attach to nodes along a slice through multiple layers of the network in order to make predictions about local patches. Our flexible combination of a generatively learned sparse representation with discriminatively trained transfer classifiers extends the notion of sparse reconstruction to encompass arbitrary semantic labeling tasks.

1 Introduction

A multitude of recent work establishes the power of learning hierarchical representations for visual recognition tasks. Noteworthy examples include deep autoencoders [1], deep convolutional networks [2, 3], deconvolutional networks [4],

hierarchical sparse coding [5], and multipath sparse coding [6]. Though modeling choices and learning techniques vary, these architectures share the overall strategy of concatenating coding (or convolution) operations followed by pooling operations in a repeating series of layers. Typically, the representation at the topmost layer (or pooled codes from multiple layers [6]) serves as an input feature vector for an auxiliary classifier, such as a support vector machine (SVM), tasked with assigning a category label to the image.

Our work is motivated by exploration of the information content of the representation constructed by the rest of the network. While the topmost or pooled features robustly encode object category, what semantics can be extracted from the spatially distributed activations in the earlier network layers? Previous work attacks this question through development of tools for visualizing and probing network behavior [7]. We provide a direct result: a multilayer slice above a particular spatial location contains sufficient information for semantic labeling of a local patch. Combining predicted labels across overlapping patches yields a semantic segmentation of the entire image.

In the case of contour detection (regarded as a binary labeling problem), we show that a single layer sparse representation (albeit over multiple image scales and patch sizes) suffices to recover most edges, while a second layer adds the ability to differentiate (and suppress) texture edges. This suggests that contour detection (and its dual problem, image segmentation [8]) emerge implicitly as byproducts of deep representations.

Moreover, our reconstruction algorithm is not specific to contours. It is a recipe for transforming a generic sparse representation into a task-specific semantic labeling. We are able to reuse the same multilayer network structure for contours in order to train a system for semantic segmentation of human faces.

We make these claims in the specific context of the multipath sparse coding architecture of Bo *et al.* [6]. We learn sparse codes for different patch resolutions on image input, and, for deeper layers, on pooled and subsampled sparse representations of earlier layers. However, instead of a final step that pools codes into a single feature vector for the entire image, we use the distributed encoding in the setting of sparse reconstruction. This encoding associates a high-dimensional sparse feature vector with each pixel. For the traditional image denoising reconstruction task, convolving these vectors with the patch dictionary from the encoding stage and averaging overlapping areas yields a denoised version of the original image [9].

Our strategy is to instead swap in an entirely different dictionary for use in reconstruction. Here we generalize the notion of "dictionary" to include any function which takes a sparse feature vector as input and outputs predicted labels for a patch. Throughout the paper, these transfer dictionaries take the form of a set of logistic regression functions: one function for predicting the label of each pixel in the output patch. For a simplified toy example, Figure 1 illustrates the reconstruction obtained with such a dictionary learned for the contour detection task. Figure 2 diagrams the much larger multipath sparse coding network that our actual system uses to generate high-dimensional sparse representations. The

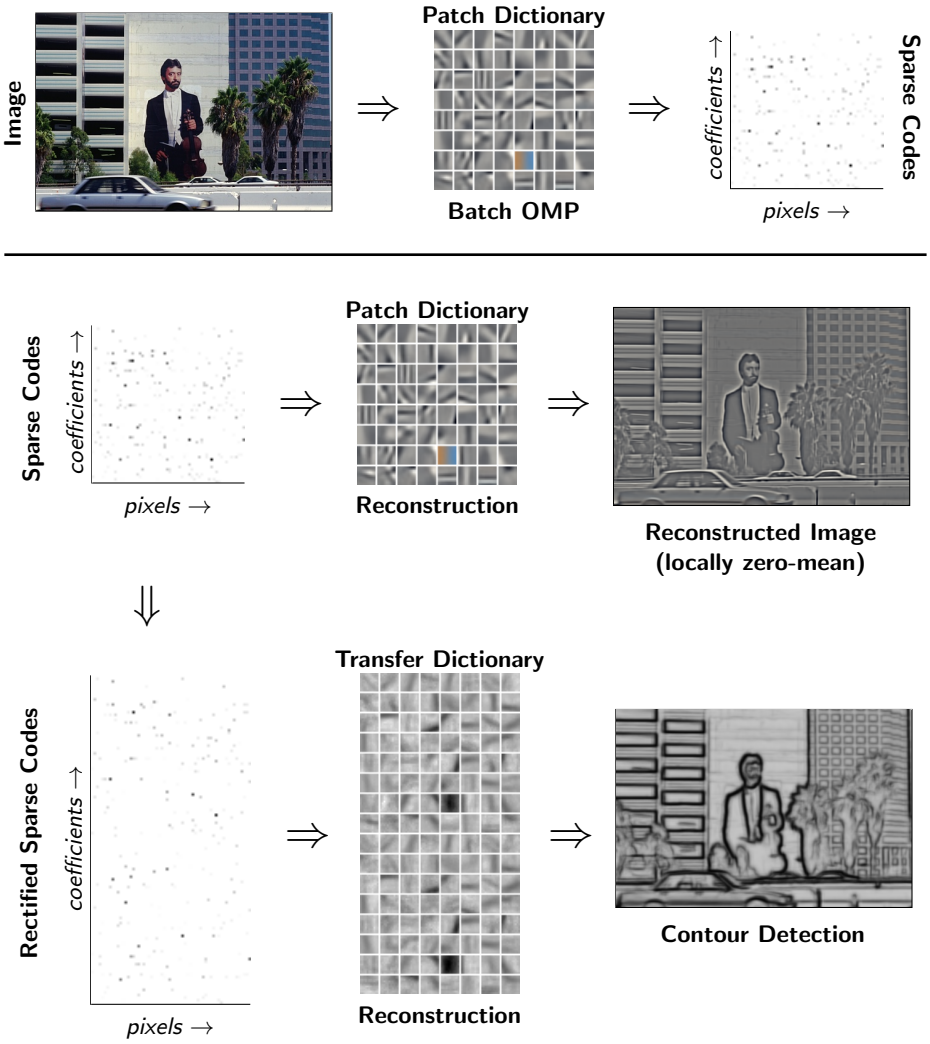

Fig. 1. Reconstructive sparse code transfer. *Top:* Applying batch orthogonal matching pursuit (BOMP) [10, 11] against a learned appearance dictionary determines a sparse code for each patch in the image. We subtract means of patch RGB channels prior to encoding. *Bottom:* Convolving the sparse code representation with the same dictionary reconstructs a locally zero-mean version of the input image. Alternatively, rectifying the representation and applying a transfer function (learned for contour detection) reconstructs an edge map. For illustrative purposes, we show a small single-layer dictionary (64 11x11 patches) and simple transfer functions (logistic classifiers) whose coefficients are rendered as a corresponding dictionary. Our deeper representations (Figure 2) are much higher-dimensional and yield better performing, but not easily visualized, transfer functions.

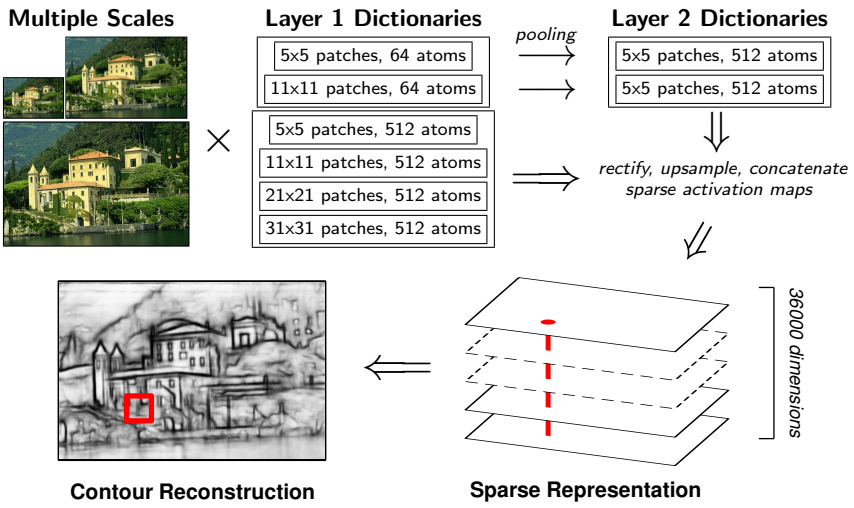

Fig. 2. Multipath sparse coding and reconstruction network. We resize an image to 6 different scales (3 shown) and encode each using dictionaries learned for different patch sizes and atom counts. Encoding sparsity is 2 and 4 nonzero coefficients for 64 and 512 atom dictionaries, respectively. The output representation of the smaller dictionaries is pooled, subsampled, and fed to a second sparse coding layer. We then rectify all sparse activation maps, upsample them to the original grid size, and concatenate to form a 36000-dimensional sparse representation for each pixel on the image grid. A set of logistic classifiers then transform the sparse vector associated with each pixel (red vertical slice) into a predicted labeling of the surrounding patch (red box).

structural similarity to the multipath network of Bo *et al.* [6] is by design. They tap part of such a network for object recognition; we tap a different part of the network for semantic segmentation. This suggests that it may be possible to use an underlying shared representation for both tasks.

In addition to being an implicit aspect of deep representations used for object recognition, our approach to contour detection is entirely free of reliance on hand-crafted features. As Section 2 reviews, this characteristic is unique amongst competing contour detection algorithms. Sections 3, 4, and 5 describe the technical details behind our two-stage approach of sparse coding and reconstructive transfer. Section 6 visualizes and benchmarks results for our primary application of contour detection on the Berkeley segmentation dataset (BSDS) [12]. We also show results for a secondary application of semantic part labeling on the Labeled Faces in the Wild (LFW) dataset [13, 14]. Section 7 concludes.

2 Related Work

Contour detection has long been a major research focus in computer vision. Arbeláez *et al.* [8] catalogue a vast set of historical and modern algorithms. Three

different approaches [8, 15, 16] appear competitive for state-of-the-art accuracy. Arbeláez *et al.* [8] derive pairwise pixel affinities from local color and texture gradients [17] and apply spectral clustering [18] followed by morphological operations to obtain a global boundary map.

Ren and Bo [15] adopt the same pipeline, but use gradients of sparse codes instead of the color and texture gradients developed by Martin *et al.* [17]. Note that this is completely different from the manner in which we propose to use sparse coding for contour detection. In [15], sparse codes from a dictionary of small 5×5 patches serve as replacement for the textons [19] used in previous work [17, 8]. Borrowing the hand-designed filtering scheme of [17], half-discs at multiple orientations act as regions over which codes are pooled into feature vectors and then classified using an SVM. In contrast, we use a range of patch resolutions, from 5×5 to 31×31, without hand-designed gradient operations, in a reconstructive setting through application of a learned transfer dictionary. Our sparse codes assume a role different than that of serving as glorified textons.

Dollár and Zitnick [16] learn a random decision forest on feature channels consisting of image color, gradient magnitude at multiple orientations, and pairwise patch differences. They cluster ground-truth edge patches by similarity and train the random forest to predict structured output. The emphasis on describing local edge structure in both [16] and previous work [20, 21] matches our intuition. However, sparse coding offers a more flexible methodology for achieving this goal. Unlike [16], we learn directly from image data (not predefined features), in an unsupervised manner, a generic (not contour-specific) representation, which can then be ported to many tasks via a second stage of supervised transfer learning.

Mairal *et al.* [22] use sparse models as the foundation for developing an edge detector. However, they focus on discriminative dictionary training and per-pixel labeling using a linear classifier on feature vectors derived from error residuals during sparse coding of patches. This scheme does not benefit from the spatial averaging of overlapping predictions that occurs in structured output paradigms such as [16] and our proposed algorithm. It also does not incorporate deeper layers of coding, an aspect we find to be crucial for capturing texture characteristics in the sparse representation.

Yang *et al.* [23] study the problem of learning dictionaries for coupled feature spaces with image super-resolution as an application. We share their motivation of utilizing sparse coding in a transfer learning context. As the following sections detail, we differ in our choice of a modular training procedure split into distinct unsupervised (generic) and supervised (transfer) phases. We are unique in targeting contour detection and face part labeling as applications.

3 Sparse Representation

Given image I consisting of c channels ($c = 3$ for an RGB color image) defined over a 2-dimension grid, our sparse coding problem is to represent each $m \times m \times c$ patch $x \in I$ as a sparse linear combination z of elements from a dictionary $D = [d_0, d_1, \ldots, d_{L-1}] \in \Re^{(m \cdot m \cdot c) \times L}$. From a collection of patches $X = [x_0, x_1, ...]$

randomly sampled from a set of training images, we learn the corresponding sparse representations $Z = [z_0, z_1, ...]$ as well as the dictionary D using the MI-KSVD algorithm proposed by Bo *et al.* [6]. MI-KSVD finds an approximate solution to the following optimization problem:

$$\begin{aligned} &\underset{D,\ Z}{\operatorname{argmin}} \left[||X - DZ||_F^2 + \lambda \sum_{i=0}^{L-1} \sum_{j=0, j\neq i}^{L-1} |d_i^T d_j| \right] \\ &s.t.\ \forall i,\ ||d_i||_2 = 1 \text{ and } \forall n,\ ||z_n||_0 \leq K \end{aligned} \tag{1}$$

where $||\cdot||_F$ denotes Frobenius norm and K is the desired sparsity level. MI-KSVD adapts KSVD [24] by balancing reconstruction error with mutual incoherence of the dictionary. This unsupervised training stage is blind to any task-specific uses of the sparse representation.

Once the dictionary is fixed, the desired encoding $z \in \Re^L$ of a novel patch $x \in \Re^{m \cdot m \cdot c}$ is:

$$\underset{z}{\operatorname{argmin}} ||x - Dz||^2 \quad s.t.\ ||z||_0 \leq K \tag{2}$$

Obtaining the exact optimal z is NP-hard, but the orthogonal matching pursuit (OMP) algorithm [10] is a greedy iterative routine that works well in practice. Over each of K rounds, it selects the dictionary atom (codeword) best correlated with the residual after orthogonal projection onto the span of previously selected codewords. Batch orthogonal matching pursuit [11] precomputes correlations between codewords to significantly speed the process of coding many signals against the same dictionary. We extract the $m \times m$ patch surrounding each pixel in an image and encode all patches using batch orthogonal matching pursuit.

4 Dictionary Transfer

Coding an image I as described in the previous section produces a sparse matrix $Z \in \Re^{L \times N}$, where N is the number of pixels in the image and each column of Z has at most K nonzeros. Reshaping each of the L rows of Z into a 2-dimensional grid matching the image size, convolving with the corresponding codeword from D, and summing the results approximately reconstructs the original image. Figure 1 (middle) shows an example with the caveat that we drop patch means from the sparse representation and hence also from the reconstruction. Equivalently, one can view D as defining a function that maps a sparse vector $z \in \Re^L$ associated with a pixel to a predicted patch $P \in \Re^{m \times m \times c}$ which is superimposed on the surrounding image grid and added to overlapping predictions.

We want to replace D with a function $F(z)$ such that applying this procedure with $F(\cdot)$ produces overlapping patch predictions that, when averaged, reconstruct signal $\widehat{G}$ which closely approximates some desired ground-truth labeling G. G lives on the same 2-dimensional grid as I, but may differ in number of channels. For contour detection, G is a single-channel binary image indicating presence or absence of an edge at each pixel. For semantic labeling, G may

have as many channels as categories with each channel serving as an indicator function for category presence at every location.

We regard choice of $F(\cdot)$ as a transfer learning problem given examples of sparse representations and corresponding ground-truth, $\{(Z_0, G_0), (Z_1, G_1), \ldots\}$. To further simplify the problem, we consider only patch-wise correspondence. Viewing Z and G as living on the image grid, we sample a collection of patches $\{g_0, g_1, \ldots\}$ from $\{G_0, G_1, \ldots\}$ along with the length L sparse coefficient vectors located at the center of each sampled patch, $\{z_0, z_1, \ldots\}$. We rectify each of these sparse vectors and append a constant term:

$$\hat{z}_i = \begin{bmatrix} \max(z_i^T, 0), & \max(-z_i^T, 0), & 1 \end{bmatrix}^T \tag{3}$$

Our patch-level transfer learning problem is now to find $F(\cdot)$ such that:

$$F(\hat{z}_i) \approx g_i \quad \forall i \tag{4}$$

where $\hat{z}_i \in \Re^{2L+1}$ is a vector of sparse coefficients and $g_i \in \Re^{m \times m \times h}$ is a target ground-truth patch. Here, h denotes the number of channels in the ground-truth (and its predicted reconstruction).

While one could still choose any method for modeling $F(\cdot)$, we make an extremely simple and efficient choice, with the expectation that the sparse representation will be rich enough that simple transfer functions will work well. Specifically, we split $F(\cdot)$ into a set $[f_0, f_1, ..., f_{(m^2h-1)}]$ of independently trained predictors $f(\cdot)$, one for each of the m^2h elements of the output patch. Our transfer learning problem is now:

$$f_j(\hat{z}_i) \approx g_i[j] \quad \forall i, j \tag{5}$$

As all experiments in this paper deal with ground-truth in the form of binary indicator vectors, we set each $f_j(\cdot)$ to be a logistic classifier and train its coefficients using L2-regularized logistic regression.

Predicting $m \times m$ patches means that each element of the output reconstruction is an average of outputs from m^2 different $f_j(\cdot)$ classifiers. Moreover, one would expect (and we observe in practice) the accuracy of the classifiers to be spatially varying. Predicted labels of pixels more distant from the patch center are less reliable than those nearby. To correct for this, we weight predicted patches with a Gaussian kernel when spatially averaging them during reconstruction.

Additionally, we would like the computation time for prediction to grow more slowly than $O(m^2)$ as patch size increases. Because predictions originating from similar spatial locations are likely to be correlated and a Gaussian kernel gives distant neighbors small weight, we construct an adaptive kernel $\mathcal{W}$, which approximates the Gaussian, taking fewer samples with increasing distance, but upweighting them to compensate for decreased sample density. Specifically:

$$\mathcal{W}(x, y; \sigma) = \begin{cases} \mathcal{G}(x, y; \sigma)/\rho(x, y) & \text{if } (x, y) \in \mathcal{S} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

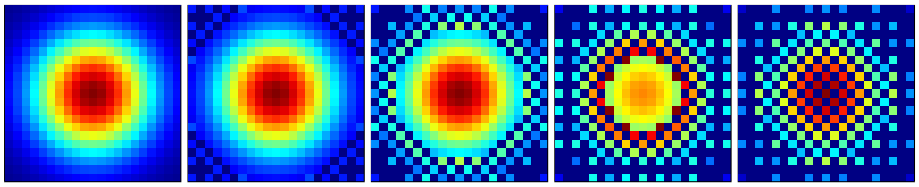

Fig. 3. Subsampled patch averaging kernels. Instead of uniformly averaging overlapping patch predictions during reconstruction, we weight them with a Gaussian kernel to model their spatially varying reliability. As a classifier evaluation can be skipped if its prediction will receive zero weight within the averaging procedure, we adaptively sparsify the kernel to achieve a runtime speedup. Using the aggressively subsampled *(rightmost)* kernel is 3x faster than using the non-subsampled *(leftmost)* version, and offers equivalent accuracy. We also save the expense of training unused classifiers.

where $\mathcal{G}$ is a 2D Gaussian, $\mathcal{S}$ is a set of sample points, and $\rho(x, y)$ measures the local density of sample points. Figure 3 provides an illustration of $\mathcal{W}$ for fixed σ and sampling patterns which repeatedly halve density at various radii.

We report all experimental results using the adaptively sampled approximate Gaussian kernel during reconstruction. We found it to perform equivalently to the full Gaussian kernel and better than uniform patch weighting. The adaptive weight kernel not only reduces runtime, but also reduces training time as we neither run nor train the $f_j(\cdot)$ classifiers that the kernel assigns zero weight.

5 Multipath Network

Sections 3 and 4 describe our system for reconstructive sparse code transfer in the context of a single generatively learned patch dictionary D and the resulting sparse representation. In practice, we must offer the system a richer view of the input than can be obtained from coding against a single dictionary. To accomplish this, we borrow the multipath sparse coding framework of Bo *et al.* [6] which combines two strategies for building richer representations.

First, the image is rescaled and all scales are coded against multiple dictionaries for patches of varying size. Second, the output sparse representation is pooled, subsampled, and then treated as a new input signal for another layer of sparse coding. Figure 2 describes the network architecture we have chosen in order to implement this strategy. We use rectification followed by hybrid average-max pooling (the average of nonzero coefficients) between layers 1 and 2. For 5×5 patches, we pool over 3×3 windows and subsample by a factor of 2, while for 11×11 patches, we pool over 5×5 windows and subsample by a factor of 4.

We concatenate all representations generated by the 512-atom dictionaries, rectify, and upsample them so that they live on the original image grid. This results in a 36000-dimensional sparse vector representation for each image pixel. Despite the high dimensionality, there are only a few hundred nonzero entries per pixel, so total computational work is quite reasonable.

The dictionary transfer stage described in Section 4 now operates on these high-dimensional concatenated sparse vectors ($L = 36000$) instead of the output of a single dictionary. Training is more expensive, but classification and reconstruction is still cheap. The cost of evaluating the logistic classifiers scales with the number of nonzero coefficients in the sparse representations rather than the dimensionality. As a speedup for training, we drop a different random 50% of the representation for each of the logistic classifiers.

6 Experiments

We apply multipath reconstructive sparse code transfer to two pixel labeling tasks: contour detection on the Berkeley segmentation dataset (BSDS) [12], and semantic labeling of human faces (into skin, hair, and background) on the part subset [14] of the Labeled Faces in the Wild (LFW) dataset [13]. We use the network structure in Figure 2 in both sets of experiments, with the only difference being that we apply a zero-mean transform to patch channels prior to encoding in the BSDS experiments. This choice was simply made to increase dictionary efficiency in the case of contour detection, where absolute color is likely less important. For experiments on the LFW dataset, we directly encode raw patches.

6.1 Contour Detection

Figure 4 shows contour detection results on example images from the test set of the 500 image version [8] of the BSDS [12]. Figure 5 shows the precision-recall curve for our contour detector as benchmarked against human-drawn ground-truth. Performance is comparable to the heavily-engineered state-of-the-art global Pb (gPb) detector [8].

Note that both gPb and SCG [15] apply a spectral clustering procedure on top of their detector output in order to generate a cleaner globally consistent result. In both cases, this extra step provides a performance boost. Table 1 displays a more nuanced comparison of our contour detection performance with that of SCG before globalization. Our detector performs comparably to (local) SCG. We expect that inclusion of a sophisticated spectral integration step [25] will further boost our contour detection performance, but leave the proof to future work.

It is also worth emphasizing that our system is the only method in Table 1 that relies on neither hand-crafted filters (global Pb, SCG) nor hand-crafted features (global Pb, Structured Edges). Our system is learned entirely from data and even relies on a *generatively trained* representation as a critical component.

Additional analysis of our results yields the interesting observation that the second layer of our multipath network appears crucial to texture understanding. Figure 6 shows a comparison of contour detection results when our system is restricted to use only layer 1 versus results when the system uses the sparse representation from both layers 1 and 2. Inclusion of the second layer (deep sparse coding) essentially allows the classification stage to learn an off switch for texture edges.

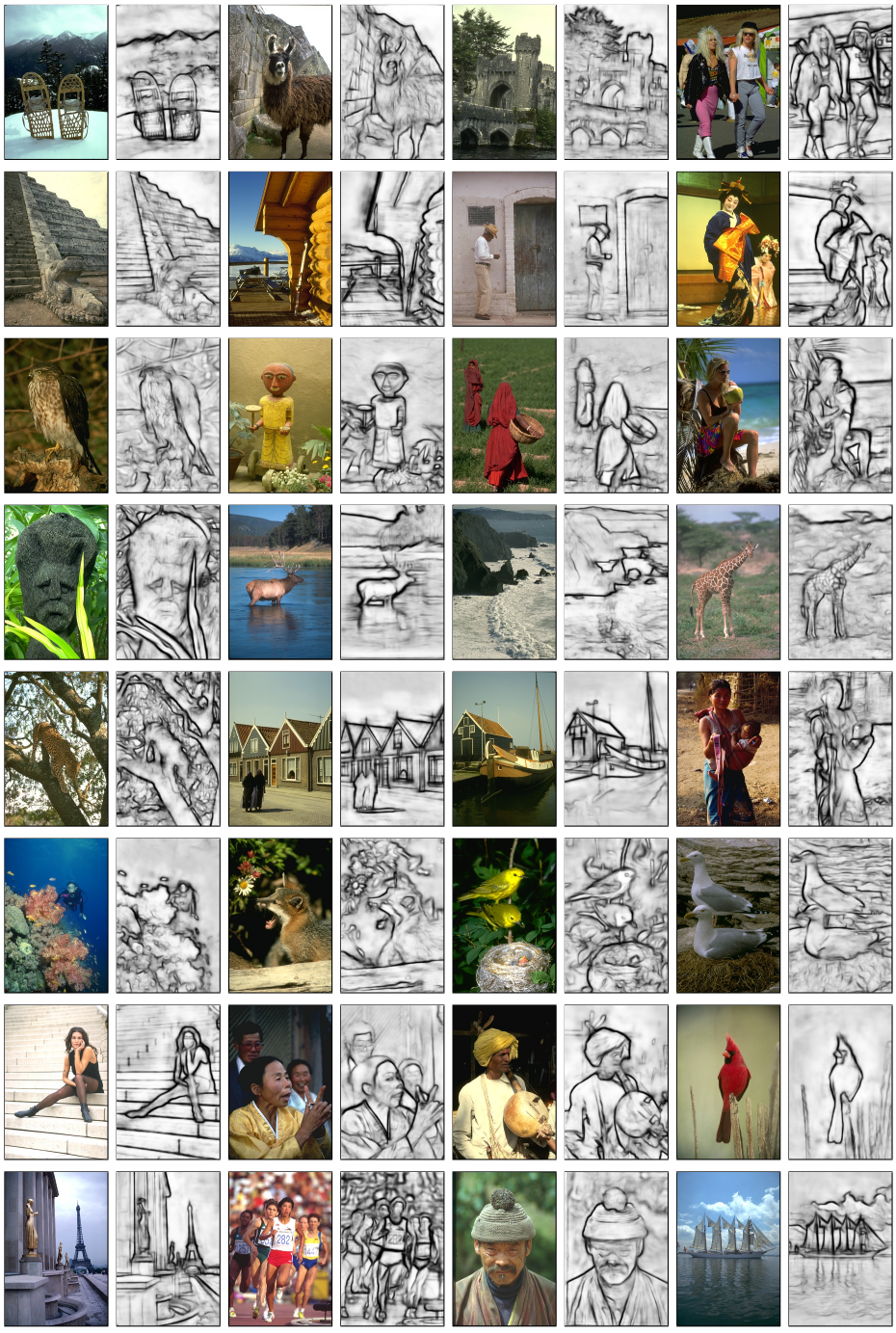

Fig. 4. Contour detection results on BSDS500. We show images and corresponding contours produced via reconstructive sparse code transfer. Contours are displayed prior to applying non-maximal suppression and thinning for benchmarking purposes.

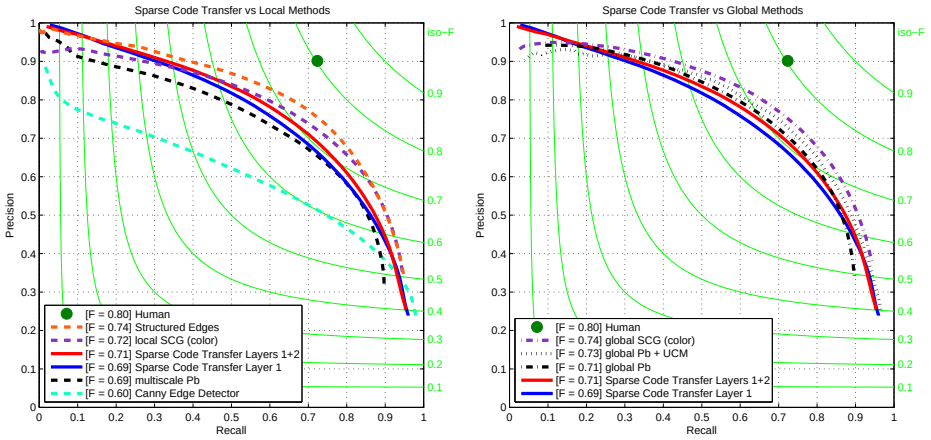

Fig. 5. Contour detection performance on BSDS500. Our contour detector (solid red curve) achieves a maximum F-measure ($\frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$) of 0.71, similar to other leading approaches. Table 1 elaborates with more performance metrics. *Left:* We show full precision-recall curves for algorithms that, like ours, predict boundary strength directly from local image patches. Of these algorithms, sparse code transfer is the only one free of reliance on hand-designed features or filters. Note that addition of the second layer improves our system's performance, as seen in the jump from the blue to red curve. *Right:* Post-processing steps that perform global reasoning on top of locally detected contours can further boost performance. Application of spectral clustering to multiscale Pb and local SCG yields superior results shown as global Pb and global SCG, respectively. Further transforming global Pb via an Ultrametric Contour Map (UCM) [26] yields an additional boost. Without any such post-processing, our local detector offers performance equivalent to that of global Pb.

	Performance Metric			Hand-Designed		Spectral
	ODS F	OIS F	AP	Features?	Filters?	Globalization?
Human	0.80	0.80	–	–	–	–
Structured Edges [16]	**0.74**	**0.76**	**0.78**	yes	no	no
local SCG (color) [15]	0.72	0.74	0.75	no	yes	no
Sparse Code Transfer Layers 1+2	0.71	0.72	0.74	**no**	**no**	no
Sparse Code Transfer Layer 1	0.69	0.71	0.72	**no**	**no**	no
local SCG (gray) [15]	0.69	0.71	0.71	no	yes	no
multiscale Pb [8]	0.69	0.71	0.68	yes	yes	no
Canny Edge Detector [27]	0.60	0.63	0.58	yes	yes	no
global SCG (color) [15]	**0.74**	**0.76**	0.77	yes	yes	yes
global Pb + UCM [8]	0.73	**0.76**	0.73	yes	yes	yes + UCM
global Pb [8]	0.71	0.74	0.65	yes	yes	yes

Table 1. Contour benchmarks on BSDS500. Performance of our sparse code transfer technique is competitive with the current best performing contour detection systems [8, 15, 16]. Shown are the detector F-measures when choosing an optimal threshold for the entire dataset (ODS) or per image (OIS), as well as the average precision (AP). The upper block of the table reports scores prior to application of spectral globalization, while the lower block reports improved results of some systems afterwards. Note that our system is the only approach in which both the feature representation and classifier are entirely learned.

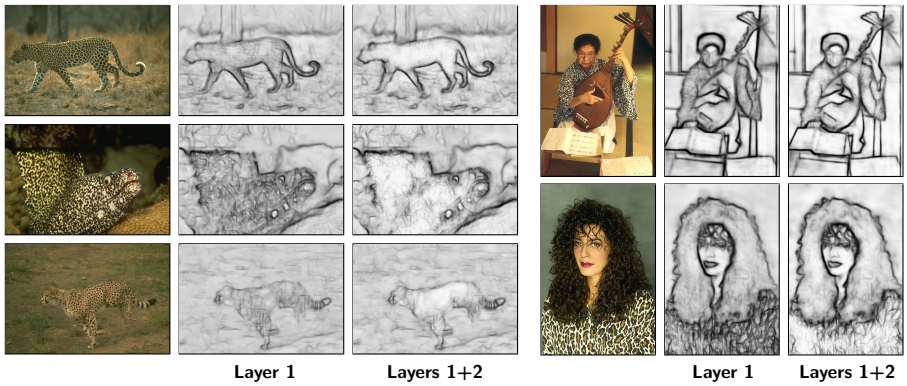

Fig. 6. Texture understanding and network depth. From left to right, we display an image, the contours detected using only the sparse representation from the layer 1 dictionaries in Figure 2, and the contours detected using the representation from both layers 1 and 2. Inclusion of the second layer is crucial to enabling the system to suppress undesirable fine-scale texture edges.

6.2 Semantic Labeling of Faces

Figure 7 shows example results for semantic segmentation of skin, hair, and background classes on the LFW parts dataset using reconstructive sparse code transfer. All results are for our two-layer multipath network. As the default split of the LFW parts dataset allowed images of the same individual to appear in both training and test sets, we randomly re-split the dataset with the constraint that images of a particular individual were either all in the training set or all in the test set, with no overlap. All examples in Figure 7 are from our test set after this more stringent split.

Note that while faces are centered in the LFW part dataset images, we directly apply our algorithm and make no attempt to take advantage of this additional information. Hence, for several examples in Figure 7 our learned skin and hair detectors fire on both primary and secondary subjects appearing in the photograph.

7 Conclusion

We demonstrate that sparse coding, combined with a reconstructive transfer learning framework, produces results competitive with the state-of-the-art for contour detection. Varying the target of the transfer learning stage allows one to port a common sparse representation to multiple end tasks. We highlight semantic labeling of faces as an additional example. Our approach is entirely data-driven and relies on no hand-crafted features. Sparse representations similar to the one we consider also arise naturally in the context of deep networks for image recognition. We conjecture that multipath sparse networks [6] can produce

Fig. 7. Part labeling results on LFW. *Left:* Image. *Middle:* Ground-truth. Semantic classes are skin (green), hair (red), and background (blue). *Right:* Semantic labeling predicted via reconstructive sparse code transfer.

shared representations useful for many vision tasks and view this as a promising direction for future research.

Acknowledgments. ARO/JPL-NASA Stennis NAS7.03001 supported Michael Maire's work.

References

1. Le, Q.V., Ranzato, M., Monga, R., Devin, M., Chen, K., Corrado, G.S., Dean, J., Ng, A.Y.: Building high-level features using large scale unsupervised learning. ICML (2012)
2. LeCun, Y., Kavukcuoglu, K., Farabet, C.: Convolutional networks and applications in vision. ISCAS (2010)
3. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. NIPS (2012)
4. Zeiler, M.D., Taylor, G.W., Fergus, R.: Adaptive deconvolutional networks for mid and high level feature learning. ICCV (2011)
5. Yu, K., Lin, Y., Lafferty, J.: Learning image representations from the pixel level via hierarchical sparse coding. CVPR (2011)
6. Bo, L., Ren, X., Fox, D.: Multipath sparse coding using hierarchical matching pursuit. CVPR (2013)
7. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. ECCV (2014)
8. Arbeláez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. PAMI (2011)
9. Elad, M., Aharon, M.: Image denoising via sparse and redundant representations over learned dictionaries. IEEE Transactions on Image Processing (2006)
10. Pati, Y.C., Rezaiifar, R., Krishnaprasad, P.S.: Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. Asilomar Conference on Signals, Systems and Computers (1993)
11. Rubinstein, R., Zibulevsky, M., Elad, M.: Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit. (2008)
12. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. ICCV (2001)
13. Huang, G.B., Ramesh, M., Berg, T., Learned-Miller, E.: Labeled faces in the wild: A database for studying face recognition in unconstrained environments. (2007)
14. Kae, A., Sohn, K., Lee, H., Learned-Miller, E.: Augmenting CRFs with Boltzmann machine shape priors for image labeling. CVPR (2013)
15. Ren, X., Bo, L.: Discriminatively trained sparse code gradients for contour detection. NIPS (2012)
16. Dollár, P., Zitnick, C.L.: Structured forests for fast edge detection. ICCV (2013)
17. Martin, D., Fowlkes, C., Malik, J.: Learning to detect natural image boundaries using local brightness, color and texture cues. PAMI (2004)
18. Shi, J., Malik, J.: Normalized cuts and image segmentation. PAMI (2000)
19. Malik, J., Belongie, S., Leung, T., Shi, J.: Contour and texture analysis for image segmentation. IJCV (2001)
20. Ren, X., Fowlkes, C., Malik, J.: Figure/ground assignment in natural images. ECCV (2006)

21. Lim, J., Zitnick, C.L., Dollár, P.: Sketch tokens: A learned mid-level representation for contour and object detection. CVPR (2013)
22. Mairal, J., Leordeanu, M., Bach, F., Hebert, M., Ponce, J.: Discriminative sparse image models for class-specific edge detection and image interpretation. ECCV (2008)
23. Yang, J., Wang, Z., Lin, Z., Shu, X., Huang, T.: Bilevel sparse coding for coupled feature spaces. CVPR (2012)
24. Aharon, M., Elad, M., Bruckstein, A.: K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. IEEE Transactions on Signal Processing (2006)
25. Maire, M., Yu, S.X., Perona, P.: Progressive multigrid eigensolvers for multiscale spectral segmentation. ICCV (2013)
26. Arbeláez, P.: Boundary extraction in natural images using ultrametric contour maps. POCV (2006)
27. Canny, J.: A computational approach to edge detection. PAMI (1986)