Learning Hierarchical Representations for Face Verification with Convolutional Deep Belief Networks

Gary B. Huang
University of Massachusetts
Amherst, MA
gbhuang@cs.umass.edu

Honglak Lee
University of Michigan
Ann Arbor, MI
honglak@eecs.umich.edu

Erik Learned-Miller
University of Massachusetts
Amherst, MA
elm@cs.umass.edu

Abstract

Most modern face recognition systems rely on a feature representation given by a hand-crafted image descriptor, such as Local Binary Patterns (LBP), and achieve improved performance by combining several such representations. In this paper, we propose deep learning as a natural source for obtaining additional, complementary representations. To learn features in high-resolution images, we make use of convolutional deep belief networks. Moreover, to take advantage of global structure in an object class, we develop local convolutional restricted Boltzmann machines, a novel convolutional learning model that exploits the global structure by not assuming stationarity of features across the image, while maintaining scalability and robustness to small misalignments. We also present a novel application of deep learning to descriptors other than pixel intensity values, such as LBP. In addition, we compare performance of networks trained using unsupervised learning against networks with random filters, and empirically show that learning weights not only is necessary for obtaining good multilayer representations, but also provides robustness to the choice of the network architecture parameters. Finally, we show that a recognition system using only representations obtained from deep learning can achieve comparable accuracy with a system using a combination of hand-crafted image descriptors. Moreover, by combining these representations, we achieve state-of-the-art results on a real-world face verification database.

1. Introduction

There has been a significant amount of progress made in the area of face recognition, with recent research focusing on the face verification problem. In this set-up, pairs of images are given at training time, along with a label indicating whether the pair contains two images of the same person (matched pair), or two images of two different persons (mismatched pair). At test time, a new pair of images is presented, and the task is to assign the appropriate matched/mismatched label. Unlike other face recognition problem formulations, it is not assumed that the person identities in the training and test sets have any overlap, and often the two sets are disjoint.

This set-up removes one of the fundamental assumptions of the traditional experimental design, making it possible to perform recognition on never-before-seen faces. Another important assumption that has been relaxed recently is the amount of control the experimenter has over the acquisition of the images. In unconstrained face verification, the only assumption made is that the face images were detected by a standard face detector. In particular, images contain significant variations in nuisance factors such as complex background, lighting, pose, and occlusions. These factors lead to large intra-class differences, making the unconstrained face verification problem very difficult.

The current standard for benchmarking performance on unconstrained face verification is the Labeled Faces in the Wild (LFW) data set [11]. Since the release of the database, classification accuracy on LFW has improved dramatically, from initial methods getting less than 0.75 accuracy to current state-of-the-art methods getting 0.84 to 0.86 accuracy.

The majority of existing methods for face verification rely on feature representations given by hand-crafted image descriptors, such as SIFT [18] and Local Binary Patterns (LBP) [22]. Further performance increases are obtained by combining several of these descriptors [38]. Rather than spending time attempting to engineer new image descriptors by hand, we instead propose obtaining new representations automatically through unsupervised feature learning with deep network architectures [10, 1, 30, 27, 14].

These representations offer several advantages over those obtained through hand-crafted descriptors: They can capture higher-order statistics such as corners and contours, and can be tuned to the statistics of the specific object classes being considered (*e.g.*, faces). Further, an end system making use of deep learning features can be more readily adapted to new domains where the hand-crafted descrip-

tors may not be appropriate.

Our primary contributions are as follows:

1. We develop local convolutional restricted Boltzmann machines (RBMs), a novel extension of convolutional RBMs that can adapt to the global structure in an object class, which still scale to high-resolution images and are robust to minor misalignments.
2. We present a novel application of deep learning to a Local Binary Pattern representation rather than pixel intensity representation, demonstrating the potential to learn additional representations that capture higher-order statistics of hand-crafted image descriptors.
3. We evaluate the role of learning in deep convolutional architectures, and find that although random filters perform surprisingly well for single layer models (consistent with prior work, such as [33, 12]), learning filters is necessary to obtain useful multi-layer networks and also provides robustness to the choice of the network architecture parameters.
4. We demonstrate that, despite the amount of prior efforts spent on engineering good image descriptors, by using representations obtained from deep learning, we are able to achieve comparable accuracy with state-of-the-art methods using these hand-crafted descriptors. Moreover, the information captured by the deep learning representations is complementary to the hand-crafted descriptors, and thus by combining the two sets of representations, we are able to improve the state-of-the-art face verification results on LFW.

2. Background

We review relevant work on unconstrained face verification and on deep learning.

2.1. Unconstrained Face Verification

As mentioned in the introduction, the top performing face recognition systems generally use some number of hand-crafted image descriptors, such as LBP. Cao *et al*. [3] form a pixel-level feature representation by circular sampling similar to LBP, then quantize these feature vectors using random-projection trees. Classification is done using multiple representations and comparing L_2 distance.

Wolf *et al*. [38] use the approach of "One-Shot Similarity" (OSS) measure and extensions such as "Two-Shot Similarity" (TSS). The idea of OSS is to learn a discriminative model specific to a pair of test images by using a set of background samples. A model is learned that separates one image in the pair from the background images, and is then applied to classify the other image in the pair, and this is repeated for the other image. By combining OSS and TSS using both LDA and SVM, over variants of LBP and SIFT descriptors, this method has set the current state-of-the-art results on LFW.

Nguyen and Bai [20] apply cosine similarity metric learning (CSML) to face verification, combining pixel intensity, LBP, and Gabor representations. As this approach achieves high accuracy using a small number of representations compared with [38], we use a variant of this method in our work, which we describe in Section 3. Guillaumin *et al*. [8] also apply metric learning to face verification, learning Mahalanobis metrics, particularly for situations in which a large amount of training data is available (unrestricted setting of LFW).

Kumar *et al*. [13] take a different approach, using additional outside supervised training data to learn binary classifiers for attributes such as gender, goatee, and round face, and binary classifiers that recognize a particular facial region of a particular person, referred to as simile classifiers. Face images are represented as vectors over the outputs of these different classifiers, and classification is performed using an SVM with an RBF kernel.

Deep learning has also been previously applied to face verification, and we describe this method in the next section. Pinto and Cox [24] also make use of a multi-layer architecture, where, rather than learning filters, they perform high-throughput screening by employing high-end graphics hardware and performing brute-force search for good feature representations.

Yin *et al*. [40] leverage pose information from Multi-PIE, in the form of images of the same face taken from a number of known poses, and apply this information to handle intra-class variation in LFW. By attempting to correct for intra-personal variation, they achieve state-of-the-art performance, for methods that make use of labeled training data external to LFW.

2.2. Deep Learning

As one of the most representative models in deep learning, the deep belief network (DBN) [10] is a generative graphical model consisting of a layer of visible units and multiple layers of hidden units, where each layer encodes correlations in the units in the layer below. DBNs and related unsupervised learning algorithms such as auto-encoders [1] and sparse coding [23, 15] have been used to learn higher-level feature representations from unlabeled data, suitable for use in tasks such as classification. These methods have been successfully applied to visual recognition tasks [29, 41, 16, 39, 28, 12].

Nair and Hinton [19] applied deep learning to object recognition and face verification, using a modification to binomial units that they refer to as noisy rectified linear units. To make learning computationally tractable, they subsample the face images to 32x32. In addition, their method was not translation invariant and had to rely on manual alignment through hand-corrected eye coordinates as preprocessing. In contrast, we take a convolutional learning approach,

thus we are able to train the models directly on the full-sized images without relying on careful manual alignment.

As other related work, Ranzato et al. [31] proposed a deep generative model with applications to face recognition (*e.g.*, classification), and Susskind et al. [36] applied 3-way RBMs for modeling pairs of face images. Compared to these models, we consider more scalable algorithms that can be applied to larger-sized images (150x150 pixels vs. 48x48 pixels). We also focus on the challenging task of face verification.

Our work also studies three different strategies for training the deep architectures. The straightforward approach is to train the model using images drawn from the same distribution as that for the test images, which in our case would be learning from faces in the training set. In many machine learning problems, however, we are given only a limited amount of labeled data, and this can cause an overfitting problem. Thus, we also examine the strategy of self-taught learning [26] (which is also related to semi-supervised learning [21, 4]). Finally, we also consider using random filters, motivated by the success of prior work [33, 12].

The idea of self-taught learning is to use a large amount of unlabeled data that are not directly related to the labeled data, and "transfer" low-level structures that can be shared between unlabeled and labeled data. For instance, we can imagine, for a binary image classification problem of classifying cars versus motorcycles, using a large amount of unlabeled images (that can be cheaply obtained through the web) to learn low-level features (*e.g.*, edges). For the case of generic object categorization tasks, Raina *et al.* [26] and Lee *et al.* [16] have shown successful applications of self-taught learning, using sparse coding and deep belief networks to learn feature representations from natural images. However, self-taught learning has not been used for face verification tasks.

Unlike categorizing generic object images, face verification focuses on a much more restricted subset of images (*i.e.*, faces), requiring a fine granularity of discrimination solely between images within this restricted class. Therefore, there are two interesting questions: first, whether features learned from faces, which have been trained to be useful for generating face images, are useful for discriminating between different faces; and second, whether features obtained from self-taught learning capture useful structures and representations that can be "transferred" from natural images to the face verification problem.

Finally, recent work has shown that random filters can give good performance in a convolutional architecture [33, 12]. This has led to the suggestion that one test different architectures quickly using random filters, and then select the top performing architecture to use with learned weights. In this paper, we evaluate this strategy for face verification tasks using a multiple-layer deep architecture.

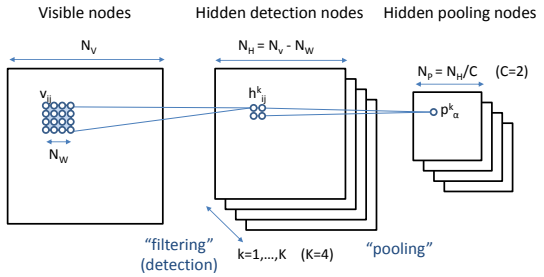

Figure 1. Schematic diagram of convolutional RBM with probabilistic max-pooling. For illustration, we used pooling ratio $C = 2$ and number of filters $K = 4$. See text for details.

3. Methods

In this section, we describe the deep learning architectures we apply to learn representations, as well as the face verification algorithm.

3.1. Learning Hierarchical Representations

We first review the convolutional restricted Boltzmann machine (CRBM) and convolutional deep belief network (CDBN) [16], then present its extension, the *local CRBM*.

3.1.1 Convolutional RBM and DBN

The convolutional restricted Boltzmann machine is an extension of the restricted Boltzmann machine (RBM). The RBM is a Markov random field with a hidden layer and a visible layer (corresponding to input data, such as image pixels), with bipartite connections between the layers (*i.e.*, there are no connections among visible nodes or among hidden nodes). In CRBM, rather than fully connecting the hidden layer and visible layer, the weights between the hidden units and the visible units are local (*i.e.*, 10x10 pixels instead of full image) and shared among all locations in the hidden units. The CRBM captures the intuition that if a certain image feature (or pattern) is useful in some locations of the image, then the same image feature can also be useful in other locations.

In this paper, we utilize a convolutional RBM with real-valued visible input nodes $\mathbf{v}$ and binary-valued hidden nodes $\mathbf{h}$. The visible input nodes can be viewed as intensity values in the $N_V \times N_V$ pixel image, and the hidden nodes are organized in 2-D configurations (*i.e.*, $\mathbf{v} \in \mathbb{R}^{N_V \times N_V}$ and $\mathbf{h} \in \{0,1\}^{N_H \times N_H}$). An illustration of CRBM can be found in Figure 1.

The CRBM has three sets of parameters: (1) K convolution filter weights between a hidden node and the visible nodes, where each filter covers $N_W \times N_W$ pixels (*i.e.*, $W^k \in \mathbb{R}^{N_W \times N_W}, k = 1, ..., K$); (2) hidden biases $b^k \in \mathbb{R}$ that are shared among hidden nodes; and (3) visible bias $c \in \mathbb{R}$ that is shared among visible nodes.

To make CRBMs more scalable, Lee *et al*. further developed "probabilistic max-pooling", a technique for incorporating local translation invariance. Max-pooling refers to operations where a local neighborhood (*e.g*., 2x2 grid) of feature detection outputs is shrunk to a pooling node by computing the maximum of the local neighbors. Max-pooling makes the feature representation become more invariant to local translations in the input data, and it has been shown to be useful in visual recognition problems [12, 2]. Probabilistic max-pooling enables the CRBM to incorporate max-pooling like behavior, while allowing probabilistic inference (such as bottom-up and top-down inference). It further enables increasingly more invariant representations as we stack CRBMs [7].

We define the energy function of the probabilistic max-pooling CRBM (with real-valued visible units) as follows:

$$\begin{aligned} P(\mathbf{v},\mathbf{h}) &= \frac{1}{Z}\exp(-E(\mathbf{v},\mathbf{h})) \\ E(\mathbf{v},\mathbf{h}) &= -\sum_{k=1}^{K}\sum_{i,j=1}^{N_H}\sum_{r,s=1}^{N_W} h_{ij}^k W_{rs}^k v_{i+r-1,j+s-1} \\ &\quad + \sum_{i,j=1}^{N_V}\frac{1}{2}v_{ij}^2 - \sum_{k=1}^{K} b_k \sum_{i,j=1}^{N_H} h_{ij}^k - c\sum_{i,j=1}^{N_V} v_{ij} \\ \text{s.t.} &\quad \sum_{(i,j)\in B_\alpha} h_{i,j}^k \le 1, \quad \forall k,\alpha \end{aligned} \tag{1}$$

Here, B_α refers to a $C \times C$ block of locally neighboring hidden units $h_{i,j}^k$ that are pooled to a pooling node p_α^k.

Under this energy function, the conditional probabilities can be computed as follows:

$$P(v_{ij}=1|\mathbf{h}) = \mathcal{N}((\sum_k W^k *_f h^k)_{ij} + c, 1) \tag{2}$$

$$P(h_{i,j}^k=1|\mathbf{v}) = \frac{\exp(I(h_{i,j}^k))}{1+\sum_{(i',j')\in B_\alpha}\exp(I(h_{i',j'}^k))} \tag{3}$$

where $I(h_{ij}^k) \triangleq b_k + (\tilde{W}^k *_v v)_{ij}$, $\mathcal{N}(\cdot)$ is a normal distribution, $\tilde{W}$ refers to flipping the original filter W in both upside-down and left-right directions, $*_v$ denotes valid convolution, and $*_f$ denotes full convolution.

At the same time, the pooling node p_α^k is a stochastic random variable that is defined as $p_\alpha^k \triangleq \sum_{(i,j)\in B_\alpha} h_{i,j}^k$, and the marginal posterior can be written as a softmax function:

$$P(p_\alpha^k=1|\mathbf{v}) = \frac{\sum_{(i',j')\in B_\alpha}\exp(I(h_{i',j'}^k))}{1+\sum_{(i',j')\in B_\alpha}\exp(I(h_{i',j'}^k))} \tag{4}$$

When sampling from the posterior (given the visible nodes), we can efficiently sample the hidden nodes in each block in parallel from multinomial distributions, then set the pooling node values accordingly.

The objective function is the log-likelihood of the training data. Although exact maximum likelihood training is intractable, the contrastive divergence approximation allows us to estimate an approximate gradient efficiently [9]. Contrastive divergence is not unbiased, but has low variance, and has been successfully applied in optimizing many undirected graphical models that have intractable partition functions [32, 37, 10].

As in Lee *et al*., we also apply sparsity regularization. Since the model is highly over-complete, it is necessary to regularize the model to prevent it from learning trivial or uninteresting feature representations (cf., see [23, 30] for other methods for enforcing sparsity.) Specifically, we add a sparsity penalty term to the log-likelihood objective to encourage each hidden unit group to have a mean activation close to a small constant. We implemented this with the following simple update rule (following each contrastive divergence update):

$$\Delta b_k \propto p - \frac{1}{N_H^2}\sum_{i,j} P(h_{ij}^k=1|\mathbf{v}), \tag{5}$$

where p is a target sparsity, and each image is treated as a mini-batch. The learning rate for the sparsity updates was chosen to make the hidden group's average activation (over entire training data) close to the target sparsity, while allowing variations of activations depending on specific input images. For more details of the overall training procedure, see [17, 34].

After training a max-pooling CRBM, we can use it to compute the posterior of the hidden (pooling) units given the input data. These hidden (pooling) unit "activations" can be used as input to further train the next layer CRBM.

By stacking the CRBMs, the algorithm can capture high-level features, such as hierarchical object-part decompositions. In our experiments, we trained CDBNs with up to two layers of CRBMs. After constructing a convolutional deep belief network, we perform (approximate) inference of the whole network in a feedforward (bottom-up) manner.

3.1.2 Local Convolutional RBM

The weight sharing scheme in a CRBM assumes that the distribution over features is stationary in an image with respect to location. However, for images belonging to a specific object class, such as faces, this assumption is no longer true. One strategy for removing this stationarity assumption is to connect each hidden unit to only a local receptive field in the visible image, as in the CRBM, but remove the parameter tying between weights for different hidden units [31]. However, even with only local connections, without any parameter tying, it is computationally intractable to scale this model to high resolution images (*e.g*.,

150x150 pixel images in the LFW dataset). Moreover, without parameter tying, the model becomes sensitive to local deformations and misalignments.

To maintain the advantages of the CRBM while exploiting global structure, we divide the image into a number of overlapping regions. The *local convolutional restricted Boltzmann machine* extends the CRBM by using a separate set of weights for each region. When trained on images with some global structure, a local CRBM can learn a more efficient representation than a CRBM since features for a particular location are learned only if they are useful for representing the corresponding region. Moreover, since filter weights are no longer shared globally, a local CRBM may be able to avoid spurious activations of hidden units outside the pre-specified local regions.

We formulate the local CRBM as follows. First, we divide the image into L overlapping regions, with the l-th region defined as $\{R^{(l)} : (r^{(l)}_{min}, r^{(l)}_{max}, c^{(l)}_{min}, c^{(l)}_{max})\}$, where r and c represent row or column index for the region in the image. For convenience of presentation, we assume that each region is square, with height and width equal to N_R. We denote by $\mathbf{v}^{(l)}$ the "submatrix" of the visible units that correspond to the l-th region.

Let each region have K filters $W^{(l),k}$ of size $N_W \times N_W$. The hidden units $\mathbf{h}^{(l),k}$ are binary random variables with 2D spatial structure ($N_H \times N_H$ grid), where $N_H \triangleq N_R - N_W + 1$.

We can now define the energy function of the local convolutional RBM as follows:[1]

$$E(\mathbf{v},\mathbf{h}) = -\sum_{l=1}^{L}\sum_{k=1}^{K}\left(\mathbf{v}^{(l)} * \tilde{W}^{(l),k}\right) \odot \mathbf{h}^{(l),k} + \sum_{i,j=1}^{N_V}\frac{1}{2}(\mathbf{v}_{ij}-c)^2 + \sum_{l=1}^{L}\sum_{k=1}^{K}\sum_{r,s=1}^{N_H} b^{(l)}_k \mathbf{h}^{(l),k}_{r,s} \quad (6)$$

where $\odot$ is the element-wise product operator, c is a visible bias, and $b^{(l)}_k$ is a hidden bias. With $\mathbf{v}$ fixed, the conditional probability of hidden units $\mathbf{h}^{(l)}$ can be defined as:

$$P(\mathbf{h}^{(l),k}_{r,s} = 1|\mathbf{v}^{(l)}) = \sigma((\mathbf{v}^{(l)} * \tilde{W}^{(l),k})_{r,s} + b^{(l)}_k). \quad (7)$$

where the $\sigma(x) = \frac{1}{1+\exp(-x)}$. We can also define the conditional probability of the visible units given the hidden units.

$$P(\mathbf{v}|\mathbf{h}) = \mathcal{N}\left(\sum_{l=1}^{L} I^{(l)}\left(\sum_{k=1}^{K} W^{(l),k} *_f \mathbf{h}^{(l),k}\right) + c, \mathbf{I}\right). \quad (8)$$

Here, $\mathbf{h} = \{\mathbf{h}^{(1)}, ..., \mathbf{h}^{(L)}\}$, and $I^{(l)}(Y)$ is a projection operator from $R^{N_R \times N_R}$ to $R^{N_V \times N_V}$ where Y is a $N_R \times N_R$ image, used to accumulate the contribution of each local region to the visible layer. Specifically, $I^{(l)}(Y)$ is defined as

$$\left[I^{(l)}(Y_{r',c'})\right]_{r,c} = \begin{cases} Y_{r',c'} & \text{if } (r,c) = \\ & (r' + r^{(l)}_{min} - 1, c' + c^{(l)}_{min} - 1) \\ 0 & \text{otherwise.} \end{cases}$$

With these conditional probabilities, we can train the local CRBM following the similar procedure as for the CRBM using contrastive divergence.

3.1.3 Learning Features from Existing Descriptors

Deep learning for images is usually performed by letting the visible units be whitened pixel intensity values. We learn additional novel representations by learning deep networks on Local Binary Patterns, demonstrating the potential for learning representations that capture higher-order statistics of hand-crafted image descriptors. Using uniform LBPs (at most two bitwise transitions), we have a 59 dimensional binary vector at each pixel location. We find a small increase in performance by first forming histograms of 3x3 neighbors (average pooling), and then learning a binary CRBM on this representation.

3.2. Recognition Algorithm

Inspired by the success of Cosine Similarity Metric Learning (CSML) [20], our face verification algorithm is also based on a metric-learning approach. For the hand-crafted model, we use the same features as was used with CSML (pixel intensity, LBP, Gabor). We additionally follow the same set-up by using PCA to reduce the dimensionality to 500, for all feature representations.

Rather than using CSML to learn a matrix A_{CSML}, we instead apply Information-Theoretic Metric Learning (ITML) [5] to produce a Mahalanobix matrix M. We then perform a Cholesky decomposition yielding a matrix A such that $A^T A = M$.

Letting x be the representation of an image after applying PCA, we obtain a feature vector y for an image by applying A and unit-normalizing, $y = \frac{Ax}{\|Ax\|}$. We then form a feature vector z for a pair of two images (with features y and y', respectively) using element-wise multiplication $z = y \odot y'$. Finally, we apply a linear SVM to the feature vectors z (for pairs of images) to perform face verification.

In practice, we find that using ITML improves performance over CSML by several percentage points. Note that if A is the identity matrix and the weights of the SVM are 1, then our system reduces to cosine similarity. Consistent with previous work [3], we found that compression using PCA followed by normalization gave the best performance.

[1] Note that we can also define probabilistic max-pooling for the local CRBM. However, for the simplicity of presentation, we present a case without probablistic max-pooling. Further, note that we use binary local CRBM when we stack the local CRBM as the second layer.

4. Experiments

For our experiments, we used the LFW-a[2] face images aligned using a commercial face alignment software, provided in [38]. We use three croppings of each image (150x150, 125x75, 100x100), resizing to the same input size for the visible layer, to capture information at different scales. For self-taught learning, we used images from the Kyoto natural images data set [6].[3]

To solve the SVM, we use the Shogun Toolbox [35].[4] We set the SVM C parameter using the development view of LFW. We optimized our CDBN code to use a GPU,[5] allowing us to test a single kernel system in several minutes and learn weights in a DBN in less than an hour.

4.1. Setting Architecture, Model Hyperparameters

One of the challenges of using a deep learning architecture is the number of architecture and model hyperparameters that one must set. For each layer of CDBN, we must decide the size of the filters, number of filters, max-pooling region size, and sparsity of the hidden units.

Saxe *et al*. [33] found some correlation between performance with random filters and learned filters for a given architecture, and suggested using search over architectures with random filters as a proxy for selecting a best architecture to use with learned weights.

We evaluated the correlation between random weight and learned weight performance for a one layer network with 16 different architectures, varying the above architecture hyperparameters. In this experiment, we used a single cropping only and did not use metric learning. Figure 2 shows a scatter plot of random weight performance versus learned weight performance. We find a somewhat high correlation of 0.40. However, a more interesting finding is that the range of accuracies for the learned filters is much more concentrated around higher values compared with the random filters. Thus, we hypothesize that, although networks with random filters can approach the same accuracy as networks with learned filters given the right architecture, an added benefit of learning is that the accuracy becomes more robust to the specific architecture hyperparameters.

Moreover, we find that multi-layer networks with random weights at each layer yield representations that lead to near-chance recognition performance. Empirically, this seems to indicate that, at least for the face verification task, the non-linearities in a multi-layer network with random filters do not give good representations, and learning is necessary. Given these findings, we set the hyperparameters by performing a coarse search over the possible values, and learning and evaluating the model on the development view of LFW.

[2] http://www.openu.ac.il/home/hassner/data/lfwa/

[3] http://www.cnbc.cmu.edu/cplab/data_kyoto.html

[4] http://www.shogun-toolbox.org/

[5] We used code from Graham Taylor: http://www.cs.nyu.edu/~gwtaylor/code/GPUmat/.

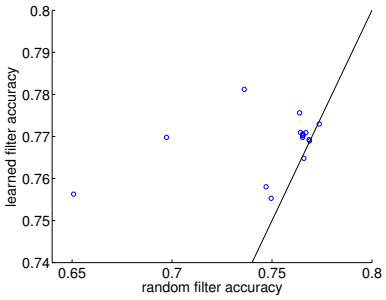

Figure 2. Random filter accuracy versus learned filter accuracy for a one-layer network, using a single image cropping and no metric learning (SVM only). The line indicates the diagonal $y = x$. From this figure, it can be seen that although there is some correlation between random filter accuracy and learned filter accuracy, learning filters has the benefit of being robust to the choice of architecture, increasing the accuracy significantly for architectures where random filters give low accuracy.

Source	Rep.	Layer	Model	Accuracy
Kyoto	pixels	1	CRBM	0.8527
Faces	pixels	1	CRBM	0.8530
Kyoto	pixels	2	CRBM	0.8522
Faces	pixels	2	CRBM	0.8457
Faces	pixels	2	local CRBM	0.8538
Kyoto	LBP	1	CRBM	0.8520
Faces	LBP	1	CRBM	0.8485
Kyoto	pixels	1+2		0.8572
Faces	pixels	1+2		0.8582
Kyoto	both	1+2		0.8660
Faces	both	1+2		0.8642
both	both	1+2		0.8688

Table 1. Verification accuracy with different deep learning architectures and training sources. The second column indicates the representation for the visible units, and "pixels" stands for whitened pixel intensity values. Top: Single representations. Bottom: Combining representations with linear SVM.

4.2. Results

The top section of Table 1 gives the accuracy for individual deep architectures. Since we expect the basic image features learned by a single layer CRBM to be largely edge-like features that are shared throughout the image, we apply our local CRBM model only at the second layer. The second layer CRBM and local CRBM have approximately the same size hidden layer representation, but the local CRBM is able to learn more filters since they are specific to each region, and achieves a higher accuracy. Figure 3 shows a visualization of the filters learned by the local CRBM. The bottom

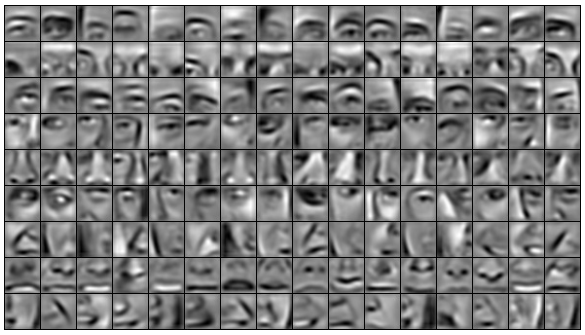

Figure 3. Visualization of sample filters from the second layer local CRBM. Each row represents filters corresponding to each local region, where the training images were divided into 9 half-overlapping regions (i.e., the size of each region is half the image size). We can see that the local CRBM capture characteristic facial parts corresponding to the local regions.

section of Table 1 gives the accuracy when combining the scores from multiple deep architectures using a linear SVM. As the different layers are capturing complementary information, we are able to achieve higher accuracy by fusing these scores.

Table 2 gives the final accuracy of our system using the deep learning representations, and the combined deep learning and hand-crafted image descriptor representations, in comparison with other systems trained using the image-restricted setting of LFW. Our system, using only deep learning representations, is competitive with state-of-the-art methods that rely on a combination of descriptions of hand-crafted image descriptors, and is state-of-the-art relative to the existing deep learning method of [19], despite the fact that [19] used manual annotations of eye coordinates to align the faces.

By combining the representations from deep learning and hand-crafted image descriptors, we obtain further improvements and achieve a new state-of-the-art accuracy. Wolf *et al*. [38] combine hand-crafted image descriptors such as LBP, Gabor, and SIFT, and additionally combine each of these representations for six different similarity metrics. Results for a single similarity metric (OSS only) are also given in Table 2. Our general methodology of learning additional representations through deep learning could also be applied to multiple similarity metrics rather than just a single metric, potentially further improving our results.

Similarly, the recent paper of Yin *et al*. [40], who achieve state-of-the-art accuracy using external training data containing pose information to handle intra-personal variation, relies on a fusion of four different hand-crafted image descriptors, and could also potentially be improved by adding additional deep learning representations.

Method	$\hat{\mu} \pm S_E$
V1-like with MKL [25]	0.7935 ± 0.0055
Linear rectified units [19]	0.8073 ± 0.0134
CSML [20]	0.8418 ± 0.0048
Learning-based descriptor [3]	0.8445 ± 0.0046
OSS, TSS, full [38]	0.8683 ± 0.0034
OSS only [38]	0.8207 ± 0.0041
Hand-crafted	0.8718 ± 0.0049
Deep Learning	0.8688 ± 0.0062
Combined	$\mathbf{0.8777 \pm 0.0062}$

Table 2. Comparison of our method with current state-of-the-art methods on LFW. The right column gives mean classification accuracy and standard error of the mean.

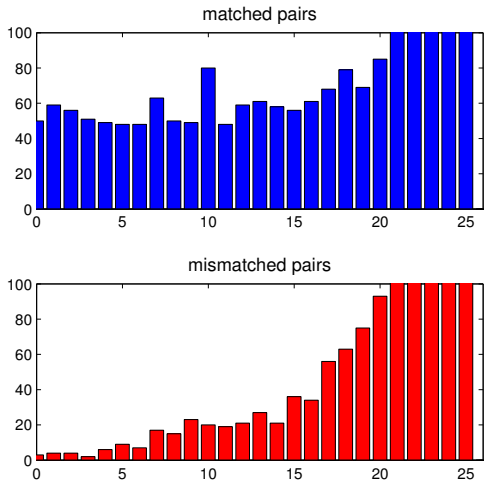

Figure 4. Histograms over the number of representations correctly classifying each pair, for matched and mismatched pairs (cut off at 100 pairs).

4.3. Analysis

We can gain additional insight into the face verification problem by looking at the number of representations whose score correctly classifies each pair. Figure 4 shows a histogram over these values, separately for mismatched pairs and matched pairs. Interestingly, the pairs that are correctly classified by few or no representations is heavily skewed toward matched pairs. Figure 5 gives some example pairs that were incorrectly classified by all representations. On our project webpage,[6] we show all 50 such pairs that were always incorrectly classified, out of which all but 3 were matched pairs. These image pairs highlight a fundamental difficulty with face verification, and verification within an object class in general, namely the large amount of intra-class variation due to several factors (*e.g*., pose).

[6] http://vis-www.cs.umass.edu/faceRecognition/

Figure 5. Sample matched pairs from LFW that were incorrectly classified by all representations.

5. Conclusion

We have demonstrated that we can improve upon methods that utilize a combination of representations from hand-crafted image descriptors by adding additional representations from deep learning. We obtain novel representations through a new local convolutional RBM model and by applying deep learning to LBP. By combining such deep learning representations with hand-crafted descriptors, we achieve new state-of-the-art accuracy on the LFW face verification database, and our methodology can be readily applied to other systems as well.

Acknowledgments

We give warm thanks to Kihyuk Sohn for helpful comments. This work was supported in part by National Science Foundation under CAREER award IIS-0546666 and the Google Faculty Research Award.

References

[1] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *NIPS*, 2007. 1, 2

[2] Y.-L. Boureau, F. R. Bach, Y. LeCun, and J. Ponce. Learning mid-level features for recognition. In *CVPR*, 2010. 4

[3] Z. Cao, Q. Yin, X. Tang, and J. Sun. Face recognition with learning-based descriptor. In *CVPR*, 2010. 2, 5, 7

[4] O. Chapelle, B. Schölkopf, and A. Zien. *Semi-supervised learning*. MIT Press, 2006. 3

[5] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *ICML*, 2007. 5

[6] E. Doi, T. Inui, T.-W. Lee, T. Wachtler, and T. J. Sejnowski. Spatiochromatic receptive field properties derived from information-theoretic analyses of cone mosaic responses to natural scenes. *Neural Computation*, 15:397–417, 2003. 6

[7] I. J. Goodfellow, Q. V. Le, A. M. Saxe, H. Lee, and A. Y. Ng. Measuring invariances in deep networks. In *NIPS*, volume 22, 2009. 4

[8] M. Guillaumin, J. Verbeek, and C. Schmid. Is that you? Metric learning approaches for face identification. In *ICCV*, 2009. 2

[9] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002. 4

[10] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006. 1, 2, 4

[11] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, 2007. 1

[12] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *ICCV*, 2009. 2, 3, 4

[13] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar. Attribute and simile classifiers for face verification. In *ICCV*, 2009. 2

[14] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *ICML*, 2007. 1

[15] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *NIPS*, 2007. 2

[16] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*, 2009. 2, 3

[17] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Unsupervised learning of hierarchical representations with convolutional deep belief networks. *Communications of the ACM*, 54(10):95–103, 2011. 4

[18] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 1

[19] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010. 2, 7

[20] H. V. Nguyen and L. Bai. Cosine similarity metric learning for face verification. In *ACCV*, 2010. 2, 5, 7

[21] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39:103–134, 2000. 3

[22] T. Ojala, M. Pietikinen, and D. Harwood. A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition*, 19(3):51–59, 1996. 1

[23] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996. 2, 4

[24] N. Pinto and D. Cox. Beyond simple features: A large-scale feature search approach to unconstrained face recognition. In *Automatic Face and Gesture Recognition*, 2011. 2

[25] N. Pinto, J. J. DiCarlo, and D. D. Cox. How far can you get with a modern face recognition test set using only simple features? In *CVPR*, 2009. 7

[26] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *ICML*, 2007. 3

[27] M. Ranzato, Y. Boureau, and Y. LeCun. Sparse feature learning for deep belief networks. *NIPS*, 2007. 1

[28] M. Ranzato and G. E. Hinton. Modeling Pixel Means and Covariances Using Factorized Third-Order Boltzmann Machines. In *CVPR*, 2010. 2

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

[30] M. Ranzato, C. Poultney, S. Chopra, and Y. LeCun. Efficient learning of sparse representations with an energy-based model. In *NIPS*, 2006. 1, 4

[31] M. Ranzato, J. Susskind, V. Mnih, and G. Hinton. On deep generative models with applications to recognition. In *CVPR*, 2011. 3, 4

[32] S. Roth and M. J. Black. Fields of experts. *IJCV*, 82(2):205–229, 2009. 4

[33] A. Saxe, P. W. Koh, Z. Chen, M. Bhand, B. Suresh, and A. Ng. On random weights and unsupervised feature learning. In *ICML*, 2011. 2, 3, 6

[34] K. Sohn, D. Y. Jung, H. Lee, and A. Hero III. Efficient learning of sparse, distributed, convolutional feature representations for object recognition. In *ICCV*, 2011. 4

[35] S. Sonnenburg, G. Raetsch, S. Henschel, C. Widmer, J. Behr, A. Zien, F. de Bona, A. Binder, C. Gehl, and V. Franc. The SHOGUN machine learning toolbox. *JMLR*, 11:1799–1802, June 2010. 6

[36] J. Susskind, G. Hinton, R. Memisevic, and M. Pollefeys. Modeling the joint density of two images under a variety of transformations. In *CVPR*, 2011. 3

[37] M. Welling, G. E. Hinton, and S. Osindero. Learning sparse topographic representations with products of student-t distributions. In *NIPS*, 2003. 4

[38] L. Wolf, T. Hassner, and Y. Taigman. Similarity scores based on background samples. In *ACCV*, 2009. 1, 2, 6, 7

[39] J. Yang, K. Yu, Y. Gong, and T. S. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009. 2

[40] Q. Yin, X. Tang, and J. Sun. An associate-predict model for face recognition. In *CVPR*, 2011. 2, 7

[41] M. Zeiler, D. Krishnan, G. Taylor, and R. Fergus. Deconvolutional networks. In *CVPR*, 2010. 2