Convolutional Transform learning

Jyoti Maggu[(1)], Emilie Chouzenoux[(2,3)], Giovanni Chierchia[(2)] and Angshul Majumdar[(1)] *

1− Indraprastha Institute of Information Technology of Delhi, Okhla Industrial Estate, Delhi, India
2− LIGM, UMR CNRS 8049, Univ. Paris Est Marne-la-Vallée, Champs-sur-Marne, France
3− CVN, INRIA Saclay, CentraleSupélec, Univ. Paris Saclay, Gif sur Yvette, France
jyotim@iiitd.ac.in, emilie.chouzenoux@univ-mlv.fr, giovanni.chierchia@esiee.fr, angshul@iiitd.ac.in

Abstract. *This work proposes a new representation learning technique called convolutional transform learning. In standard transform learning, a dense basis is learned that analyses the image to generate the representation from the image. Here, we learn a set of independent convolutional filters that operate on the images to produce representations (one corresponding to each filter). The major advantage of our proposed approach is that it is completely unsupervised; unlike CNNs where labeled images are required for training. Moreover, it relies on a well-sounded minimization technique with established convergence guarantees. We have compared the proposed method with dictionary learning and transform learning on standard image classification datasets. Results show that our method improves over the rest by a considerable margin.*

Keywords: Representation learning ; Transform learning ; Convolutive models ; Image classification ; Alternating optimization ; Proximal approaches.

1 Introduction

Learning representations from the data has always been an interesting problem for the machine learning community. A model is trained from the data to represent it in some other domain, and the learned coefficients in the other domain are used as features for solving tasks such as classification and reconstruction. There has been extensive research on learning good representations from data using well-known techniques like auto-encoders [1–3], convolutional neural networks (CNN) [4, 5], dictionary learning [6–12], and more recently transform learning [13–20].

The key idea behind CNN is to reduce drastically the number of connections to be learned by assuming that only a few learnt convolutional filters are enough

* This work was supported by the CNRS-CEFIPRA project under grant NextGenBP PRC2017.

to analyse the entire image. This automatically leads to improved generalization performance, and to a reduction of over-fitting effects. Nowadays, the success of CNNs have become so pervasive that in top tier conferences more than half of the papers are based on it. However there are some stark shortcomings. First, CNNs cannot be learned without supervision since they are based on backpropagation. Getting large volumes of labeled data is a challenge in many application fields outside digital imaging, e.g. medical imaging and remote sensing. Secondly, there is no guarantee that the learned filters are mutually different; CNN just initializes them randomly and depends on the non-convergence of backpropagation algorithm to maintain the mutual difference.

In dictionary learning, a dictionary is learned from the data such that it can synthesize the data from the learned coefficients [6, 7]. Inspired by the success of CNN models, there has been recently an increased interest for convolutional dictionary learning models, where the sought dictionary is expressed as convolutive operators associated to kernels with various sizes and shapes [10–12]. The field is still nascent and the performance of such techniques have yet to reach those of CNNs.

Transform learning can be viewed as the analysis equivalent of dictionary learning, where a basis (transform) is learned such that it analyzes the data to generate the coefficients [13–15]. Such formulation has been mainly used for the solution of inverse problems arising in image and signal processing; there are only a handful of studies that use it for machine learning tasks. In [8], transform learning (dubbed as analysis sparse coding) was used for unsupervised feature extraction. A later work [9] imposed discriminative penalties on it. In [17], a kernelized version of transform learning has been proposed. Deep versions of transform learning are also getting developed [18, 19].

A possible issue with the dictionary learning formulation is its synthesis nature; in neural network terms, this would correspond to a feed-backward neural network. On the other hand, transform learning based techniques are interpretable as a feed-forward neural network. Motivated by this observation, and by the promising results obtained by convolutional models (either based on CNNs or dictionary learning), we introduce in this work a novel transform learning strategy, called convolutional transform learning. To understand our proposal, one needs to rethink transform learning as a neural network. Instead of looking at a transform as a basis, one can think of it as connections from the input (data) to the representation (coefficient). With this interpretation, a conceptual extension to a convolutional formulation is then natural. The learning of filters/weights will be unsupervised which is an additional advantage in contrast to CNN. A spectral barrier penalty will be employed, in order to promote the diversity of the learned filters, expecting improved performance in terms of analysis metrics. Our learning procedure will be sufficiently versatile so that the proposed formulation can easily be extended to any machine learning problem.

In the following sections, we describe the proposed formulation, the associated optimization algorithm, we present experimental results on several image-based datasets, and finally we draw conclusions.

2 Background

In this section, we recall the concepts of dictionary and transform learning.

2.1 Dictionary Learning

Dictionary learning is a popular approach to learn a representation from the data in an unsupervised fashion. From the given input data S, a dictionary/basis D and coefficients/features X are learned in such a way that the data S can be reproduced from the learned dictionary and coefficients. Mathematically, this is represented as

$$S = DX. \tag{1}$$

For learning the sparse representations (D, X), the most popular technique is probably K-SVD [7], which aims to solve the following problem:

$$\underset{D,X}{\text{minimize}} \ \|S - DX\|_F^2 \quad \text{such that} \quad \|X\|_0 \leqslant \tau, \tag{2}$$

with $\tau > 0$ the desired level of sparsity. Other techniques, based on more sophisticated priors can also be used [9–12].

2.2 Transform Learning

Dictionary learning can be seen as the task of inferring a synthesis transform from the data. The dual task of inferring an *analysis* transform from the data is called transform learning. Mathematically, this concept is expressed as $ST \approx X$, where T is the analysis transform, S is the data, and X the corresponding coefficients. For instance, in [13], the following formulation was proposed to estimate the matrices T and X:

$$\underset{T,X}{\text{minimize}} \ \|ST - X\|_F^2 + \lambda(\|T\|_F^2 - \log\det T) + \beta\|X\|_1, \tag{3}$$

with $\lambda > 0$ and $\beta > 0$. Hereabove, the $-\log\det$ term imposes a full rank on the learned transform; this prevents the degenerate solution $T = 0, X = 0$. The additional penalty $\|T\|_F^2$ is to balance scale; without this the $-\log\det$ term can keep on increasing and producing degenerate results in the other extreme. Both of these additional constraints promote the good conditioning of the learned transform. Finally, the term $\|X\|_1$ imposes a sparsity constraint on the learned coefficients.

Transform learning model is expected to be more general than dictionary learning in its notion of compressibility. It also leads to a faster learning scheme as the sparse coding step is simply one step of thresholding as contrast to dictionary learning, where the sparse coding step typically involves the inversion of a linear system. Our proposal is to extend the above formulation to the case when matrix T encodes a convolutive structure, mimicking one layer of CNN.

3 Proposed approach

We now introduce our formulation of convolutional transform learning in Section 3.1, the associated optimization algorithm in Section 3.2, and the mathematical derivations in Sections 3.3–3.4.

3.1 Convolutional Transform Learning

Let us consider a dataset $\left\{s^{(k)}\right\}_{1\leqslant k\leqslant K}$ with K entries in $\mathbb{R}^N$. Our convolutional transform learning formulation relies on the key assumption that matrix T gathers a set of M kernels $t_1, \ldots, t_M$ with M entries, i.e.

$$T = [t_1 \mid \ldots \mid t_M] \in \mathbb{R}^{M\times M}. \tag{4}$$

The proposed model then reads:

$$(\forall k \in \{1, \ldots, K\}) \quad S^{(k)}T \approx X_k. \tag{5}$$

Hereabove, $\left(S^{(k)}\right)_{1\leqslant k\leqslant K} \in \mathbb{R}^{N\times M}$ are Toeplitz matrices associated to $(s^{(k)})_{1\leqslant k\leqslant K}$ such that:

$$\begin{aligned}(\forall k \in \{1, \ldots, K\}) \quad S^{(k)}T &= \left[S^{(k)}t_1 \mid \ldots \mid S^{(k)}t_M\right] \\ &= \left[t_1 * s^{(k)} \mid \ldots \mid t_M * s^{(k)}\right]\end{aligned} \tag{6}$$

where $*$ is a discrete convolution operator with suitable padding, and

$$(\forall k \in \{1, \ldots, K\}) \quad X_k = \left[x_1^{(k)} \mid \ldots \mid x_M^{(k)}\right], \tag{7}$$

contains the coefficients associated to each entry $k \in \{1, \ldots, K\}$ of the dataset. Let us denote:

$$X = [X_1^\top \mid \ldots \mid X_K^\top]^\top \in \mathbb{R}^{NK\times M}. \tag{8}$$

The goal is then to estimate (T, X) from $\left\{s^{(k)}\right\}_{1\leqslant k\leqslant K}$. To this aim, we propose to solve the following optimization problem generalizing (3) to our convolutional learning framework:

$$\underset{T\in\mathbb{R}^{M\times M}, X\in\mathbb{R}^{NK\times M}}{\text{minimize}} \quad F(T, X) \tag{9}$$

where the objective function F is defined, for every $T \in \mathbb{R}^{M\times M}$ and every $X \in \mathbb{R}^{NK\times M}$ as:

$$\begin{aligned}F(T, X) = &\frac{1}{2}\sum_{m=1}^{M}\sum_{k=1}^{K}\|t_m * s^{(k)} - x_m^{(k)}\|_2^2 \\ &+ \sum_{m=1}^{M}\sum_{k=1}^{K}\left(\beta\|x_m^{(k)}\|_1 + \iota_{[0,+\infty[}(x_m^{(k)})\right) \\ &+ \mu\|T\|_F^2 - \lambda\log\det T \qquad (10)\\ = &\frac{1}{2}\sum_{k=1}^{K}\|S^{(k)}T - X_k\|_F^2 + \mu\|T\|_F^2 \\ &- \lambda\log\det T + \beta\|X\|_1 + \iota_{[0,+\infty[^{NK\times M}}(X). \qquad (11)\end{aligned}$$

Hereabove, function $\iota_{[0,+\infty[}$ denotes the indicator function of the positive orthant, equals to 0 for nonnegative entries, $+\infty$ elsewhere. Moreover, $(\lambda, \mu, \beta) \in]0, +\infty[^3$ are regularization parameters.

3.2 Optimization algorithm

The resolution of Problem (9) requires an efficient algorithm for dealing with nonsmooth functions and hard constraints. In the optimization literature, proximal algorithms constitute one of the most efficient approaches to tackle such problems [22–24]. The key tool in those methods is the proximity operator [25, 26] of a proper, lower semi-continuous, convex function $\psi : \mathbb{R}^N \mapsto]-\infty, +\infty]$ defined as:[1]

$$(\forall \widetilde{x} \in \mathbb{R}^N) \quad \operatorname{prox}_{\psi}(\widetilde{x}) = \arg\min_{x \in \mathbb{R}^N} \psi(x) + \frac{1}{2}\|x - \widetilde{x}\|^2. \tag{12}$$

Problem (9) fits nicely into the framework provided by the alternating proximal algorithm from [24, 27]. For any initialization $T^{[0]} \in \mathbb{R}^{M \times M}$ and $X^{[0]} \in \mathbb{R}^{NK \times M}$, its iterations are as follows:

$$\begin{array}{l} \text{For} \quad n = 0, 1, \ldots \\ \left\lfloor \begin{array}{l} T^{[n+1]} = \operatorname{prox}_{\gamma_1 F(\cdot, X^{[n]})}\left(T^{[n]}\right) \\ X^{[n+1]} = \operatorname{prox}_{\gamma_2 F(T^{[n+1]}, \cdot)}\left(X^{[n]}\right) \end{array} \right. \end{array} \tag{13}$$

where γ_1 and γ_2 are some positive constants. The convergence of sequence $(T^{(n)}, X^{(n)})_{n \in \mathbb{N}}$ to a minimizer of F is guaranteed, as a consequence of the convergence properties of the proximal regularization of Gauss-Seidel method algorithm established in [24]. In the remaining of this section, we show that the updates on both variables T and X have closed form expressions, and thus can be computed with high precision in an efficient manner.

3.3 Update of T

Let $n \in \mathbb{N}$. Then, by definition,

$$T^{[n+1]} = \operatorname{prox}_{\gamma_1 F(\cdot, X^{[n]})}\left(T^{[n]}\right) \tag{14}$$

$$= \operatorname{argmin}_{T \in \mathbb{R}^{M \times M}} \frac{1}{2} \sum_{k=1}^{K} \|S^{(k)} T - X_k^{[n]}\|_F^2$$

$$+ \mu \|T\|_F^2 - \lambda \log\det T + \frac{1}{2\gamma_1}\|T - T^{[n]}\|_F^2. \tag{15}$$

Using [28], we deduce that:

$$T^{[n+1]} = \frac{1}{2} \Lambda^{-1/2} V \left(\Sigma + (\Sigma^2 + 2\lambda I_M)^{1/2}\right) U^\top, \tag{16}$$

[1] See also http://proximity-operator.net/

with

$$\Lambda = \sum_{k=1}^{K} (S^{(k)})^\top S^{(k)} + \gamma_1^{-1} I_M + 2\mu I_M, \tag{17}$$

the singular value decomposition:

$$U \Sigma V^\top = \left(\sum_{k=1}^{K} (X_k^{[n]})^\top S^{(k)} + \gamma_1^{-1} T^{[n]} \right) \Lambda^{-1/2}, \tag{18}$$

and I_M the identity matrix of $\mathbb{R}^M$.

Remark for rectangular T: Let us emphasize that our approach, and the above update can easily be generalized to the case when matrix T is rectangular, that is $T \in \mathbb{R}^{M_1 \times M_2}$ with non necessarily equality between M_1 and M_2. Then, the penalization term on T should be replaced by:

$$(\forall T \in \mathbb{R}^{M_1 \times M_2}) \quad R(T) = \begin{cases} \mu \|T\|_F^2 - \lambda \sum_{m=1}^{M} \log(\lambda_m) & \text{if} \quad T \in \mathcal{S}_M^{++}, \\ +\infty & \text{otherwise,} \end{cases} \tag{19}$$

with $M = \min(M_1, M_2)$, $(\lambda_m)_{1 \leqslant m \leqslant M}$ are the singular values of T and $\mathcal{S}_M^{++}$ indicates the set of matrices $T \in \mathbb{R}^{M_1 \times M_2}$ with strictly positive singular values (i.e. T has rank equals to M). The gradient of (19) on its definition domain reads:

$$(\forall T \in \mathcal{S}_M^{++}) \quad \nabla R(T) = 2\mu T - \lambda T^\dagger, \tag{20}$$

with $(\cdot)^\dagger$ the pseudo-inverse operation (equivalent to inverse, when $M_1 = M_2 = M$). Using Proposition 24.68 from [21], we can determine the new update for variable T in our algorithm: Let $n \in \mathbb{N}$. Then:

$$T^{[n+1]} = \text{prox}_{\gamma_1 F(\cdot, X^{[n]})} \left(T^{[n]} \right) \tag{21}$$

$$= \text{argmin}_{T \in \mathbb{R}^{M_1 \times M_2}} \frac{1}{2} \sum_{k=1}^{K} \|S^{(k)} T - X_k^{[n]}\|_F^2 + \mu \|T\|_F^2 + \lambda R(T) + \frac{1}{2\gamma_1} \|T - T^{[n]}\|_F^2 \tag{22}$$

$$= \frac{1}{2} \Lambda^{-1} U \text{Diag} \left(\left[\sigma_1 + (\sigma_1^2 + 2\lambda)^{1/2}, \ldots, \sigma_M + (\sigma_M^2 + 2\lambda)^{1/2}, 0, \ldots, 0 \right] \right) V^\top \tag{23}$$

with

$$\Lambda^\top \Lambda = \sum_{k=1}^{K} (S^{(k)})^\top S^{(k)} + \gamma_1^{-1} I_{M_1} + 2\mu I_{M_1}, \tag{24}$$

and the singular value decomposition:

$$U \Sigma V^\top = \left(\sum_{k=1}^{K} (X_k^{[n]})^\top S^{(k)} + \gamma_1^{-1} T^{[n]} \right) \Lambda^{-1}, \tag{25}$$

with $U \in \mathbb{R}^{M_1 \times M_1}$, $V \in \mathbb{R}^{M_2 \times M_2}$ orthogonal matrices and

$$\Sigma = \mathrm{Diag}\left([\sigma_1, \ldots, \sigma_M, 0, \ldots, 0]\right).$$

The impact of log-det term in (10) is straight-forward. Such penalty allows to ensure that the kernels are diverse enough to capture good correlations and hence generate good features. Changing the penalty parameter associated to the log-det term has an important impact on the learned kernels. When the kernel size equals the number of its elements (i.e., square case), then a full rank property is enforced on T, and in the limit case when μ tends to infinity, the operator T is such that $T^{-1} = \frac{2\mu}{\lambda} T$.

3.4 Update of X

Let $n \in \mathbb{N}$. Then, using the definition of the proximity operator,

$$X^{[n+1]} = \mathrm{prox}_{\gamma_2 F(T^{[n+1]},\cdot)}\left(X^{[n]}\right) \tag{26}$$

$$= \mathrm{argmin}_{X \in \mathbb{R}^{KN \times M}} \frac{1}{2} \sum_{k=1}^{K} \|S^{(k)} T^{[n+1]} - X_k\|_F^2 + \beta \|X\|_1 + \iota_{[0,+\infty[^{KN \times M}}(X) + \frac{1}{2\gamma_2} \|X - X^{[n]}\|_F^2. \tag{27}$$

By relying on the useful properties of the proximity operator listed in [26], we obtain that, for every $k \in \{1, \ldots, K\}$,

$$X_k^{[n+1]} = \max\left(\mathcal{S}_{\frac{\gamma_2 \beta}{\gamma_2 + 1}}\left(\frac{X_k^{[n]} + \gamma_2 S^{(k)} T^{[n+1]}}{\gamma_2 + 1}\right), 0\right) \tag{28}$$

where $\mathcal{S}_\theta$ denotes the soft thresholding operator with parameter $\theta \geqslant 0$, i.e.:

$$(\forall u \in \mathbb{R}) \quad \mathcal{S}_\theta(u) = \begin{cases} u + \theta & \text{if} \quad u < -\theta \\ 0 & \text{if} \quad u \in [-\theta, \theta] \\ u - \theta & \text{if} \quad u > \theta. \end{cases} \tag{29}$$

4 Numerical results

To assess the performance of the proposed approach, we considered the following datasets of small-to-medium size, on which we performed feature extraction.

YALE [29]. The Yale dataset contains 165 images of 15 individuals, downscaled to 32-by-32 pixels. There are 11 images per subject, one per different facial expression or configuration. For our experiments, we shuffled all the samples, and took 70% for training and 30% for testing. Moreover, we generated different train/test splits: YALE-2,...,YALE-8. In a YALE-p dataset, p images per subject are kept in train set, and $11-p$ images are kept in test set. So doing, train set contains $15p$ images and test set contains $15(11-p)$ images.

E-YALE-B [30]. The Extended Yale B database contains 2432 images with 38 subjects under 64 illumination conditions. Each image is cropped to 192-by-168 pixels, and downscaled to 48-by-42 pixels. For our experiments, we shuffled all the samples, took 70% for training and 30% for testing.

AR-Face [31]. This database contains more than 4000 images of 126 different subjects (70 male and 56 female). The images have various facial expressions, the lighting varies, and some of the images are partially occluded by sunglasses and scarves. For our experiments, we selected 2600 images of 100 individuals (50 males and 50 females), that is 26 different images for each subject. Train set contains 2000 images and 600 images are kept in test set. Each image has 540 features.

4.1 Classification Accuracy

We compared the proposed feature extraction approach (ConvTL – convolutional transform learning) with transform learning (TL) [8] and dictionary learning (DL) [7]. Since our method is unsupervised, it is only fair to compare with other unsupervised representation learning tools. As these are all unsupervised learning methods, we evaluated their performance by feeding the extracted features to a supervised classifier and then computing the classification accuracy. We also performed the classification directly on raw images (Raw). For the classification task, we used two popular techniques: nearest neighbor (NN) and support vector machine (SVM). Our algorithm was ran until convergence (typically 10 iterations are sufficient), with parameters $\gamma_1 = \gamma_2 = 1$. For every tested method, the hyper-parameters were cross-validated. The results are reported in Table 1.

We found that the proposed method (ConvTL) yields better results than regular transform learning (TL) for all the considered datasets and classifiers, while being better than dictionary learning (DL) on all the datasets when using nearest neighbor classifier, and on YALE, E-YALE-B, YALE-2, YALE-6, YALE-7, and YALE-8 when using SVM classifier.

To complete our analysis, we also compared to a convolutional neural network (CNN) trained on raw images through a standard supervised classification procedure. We used a custom CNN composed of the following layers: $Conv[64 \times 3 \times 3] \rightarrow ReLU \rightarrow Pool[2 \times 2] \rightarrow BNorm \rightarrow Conv[128 \times 3 \times 3] \rightarrow ReLU \rightarrow Pool[2 \times 2] \rightarrow BNorm \rightarrow Dropout \rightarrow FC[256] \rightarrow ReLU \rightarrow FC[classes] \rightarrow Softmax$.

According to the results reported in Table 1, the proposed ConvTL compares favorably with the CNN. This may be related to the fact that CNNs are known to require large training sets in order to achieve breakthrough performance, whereas the considered datasets are small.

Another important observation is that in most of our experiments on down-sampled data, we have observed that SVM outperforms KNN. Intuitively, when we have a limited set of points in many dimensions, SVM tends to be very good because it should be able to find the linear separation that should exist. Moreover, SVM is expected to be robust to outliers since it only uses the most

Table 1: Classification accuracy on benchmark datasets.

	Dataset	Raw	TL	DL	ConvTL
NN classifier	YALE	58.00	68.00	54.00	**70.00**
	E-YALE-B	71.03	72.28	71.72	**84.00**
	AR-Faces	55.00	53.50	54.50	**56.00**
	YALE-2	43.40	49.63	43.70	**51.85**
	YALE-3	49.40	48.33	47.50	**55.83**
	YALE-4	52.38	50.48	44.76	**54.28**
	YALE-5	51.11	53.33	44.44	**54.44**
	YALE-6	53.33	50.67	50.67	**57.33**
	YALE-7	60.20	61.67	53.33	**66.67**
	YALE-8	63.60	57.78	57.78	**71.11**
SVM classifier	YALE	68.00	78.00	80.00	**88.00**
	E-YALE-B	93.24	94.21	95.58	**97.38**
	AR-Faces	87.33	84.33	**97.67**	88.87
	YALE-2	58.52	51.11	58.52	**62.22**
	YALE-3	62.50	60.83	**66.67**	64.17
	YALE-4	60.95	53.33	**64.76**	64.52
	YALE-5	66.67	57.78	**68.89**	66.67
	YALE-6	73.33	61.33	81.33	**82.67**
	YALE-7	80.00	66.67	78.33	**83.33**
	YALE-8	80.00	71.11	80.00	**84.44**
CNN classifier	YALE	84.00	-	-	-
	E-YALE-B	98.60	-	-	-
	AR-Faces	95.50	-	-	-
	YALE-2	62.96	-	-	-
	YALE-3	64.17	-	-	-
	YALE-4	67.60	-	-	-
	YALE-5	74.44	-	-	-
	YALE-6	76.00	-	-	-
	YALE-7	81.67	-	-	-
	YALE-8	82.22	-	-	-

relevant points to find the linear separation (support vectors). In general, if we have large dataset in a low dimensional space then KNN is probably a suitable choice. If we have few points in the dataset, lying in a high dimensional space, then a linear SVM is probably better.

Our classification accuracy is comparable to the one obtained with CNN. It should however be emphasized that the upvote for the proposed methodology is its unsupervised way of learning convolved features in contrast to CNN, where convolved features are learned in a supervised manner.

The learned features by the proposed method are general enough to be used for other image processing tasks by making small changes in the formulation.

4.2 Computational Time

The proposed method is tested on small size images which are downsampled from the original full size images. While the DL and TL methods take one to ten seconds for learning representations, the proposed approach takes around one minute. The difference in terms of computational time is simply related to the fact that, in case of TL and DL, the transform requires a matrix-vector product while in the proposed approach, convolution and deconvolution operations are needed.

4.3 Analysis of the learned kernels

A given number M_2 of kernels with $M_1 = M_2^2$ coefficients is learned to ideally represent the dataset. Each kernel t_m is convolved with the image s to generate a different feature vector x_m. The intra-kernel diversity is taken care by the penalties in the proposed formulation.

Figure 1 shows the kernels learned on YALE dataset, for different sizes $M_2 \in \{3, 5, 7, 9\}$. One can observe that the proposed algorithm is capable of learning nontrivial and nonidentical kernels, thanks to the regularization on T present in (9). In particular, the results reported in Table 1 were obtained by fixing $M_2 = 5$, which corresponds to a good trade-off between model accuracy and complexity.

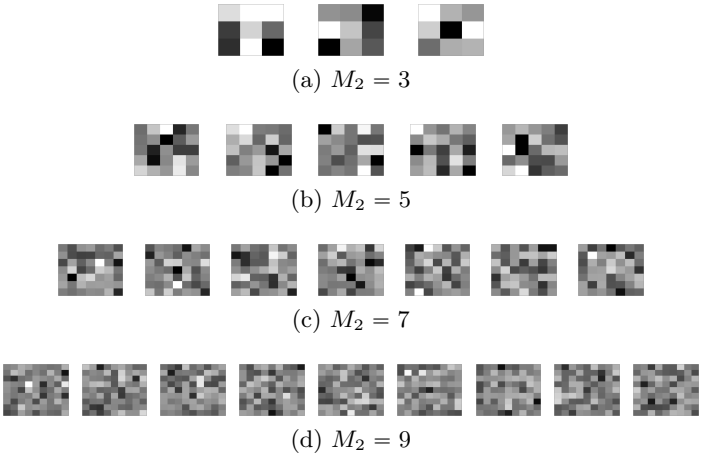
Since $M_1 > M_2$ here, the estimated T is rectangular and over-complete. The retrieved kernels are distinct from each other, as soon as $\mu > 0$. In contrast, if we had considered a large number of small size kernels (i.e., rectangular case with $M_1 < M_2$), T would have been under-complete and the number of distinct kernels would be equals to the smallest dimension of T, that is M_1; the others being some linear combination of each other. The results for this scenario are not presented here due to the lack of space.

Note that the initialization of T plays no role in the learning process, since the optimization problem in (9) is convex.

5 Conclusion

This paper introduces a novel representation learning technique, named convolutional transform learning. Comparison was performed with the off-the-shelf dictionary learning and transform learning formulations on image classification tasks. In the future, we plan to compare with several other representation learning techniques, namely autoencoder and its convolutional version, restricted Boltzmann machine and its convolutional version discriminative variants of dictionary and transform learning.

Our current formulation relies on an efficient alternating optimization technique with sounded theoretical guarantees. When applied to large scale problems, the approach can nonetheless be quite time consuming, so that, in the future we plan to parallelize portions of the algorithm with the aim to improve its computational efficiency. This will allow us to compare with deeper versions of the



(a) $M_2 = 3$

(b) $M_2 = 5$

(c) $M_2 = 7$

(d) $M_2 = 9$

Fig. 1: Kernels learned on YALE dataset.

aforesaid techniques on larger datasets. The next possible extension to the proposed method could be making it multilayered architecture involving various size and number of kernels in each layer. One can expect the multilayer formulation to scale well with large number of full size images.

References

1. P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." *J. Mach. Learn. Res.*, pp. 3371–3408, Dec. 2010.
2. D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. of ICLR*, Banff, Canada, Apr. 2014.
3. A. Makhzani, J. Shlens, N. Jaitly, and I. Goodfellow, "Adversarial autoencoders," in *Proc. of ICLR*, San Juan, Puerto Rico, May 2016.
4. LeCun, Y., Boser, B.,Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. and Jackel, L. E.: Backpropagation applied to handwritten zip code recognition, J. Neural Comput., 1, 4,541–551,(1989).
5. Lecun, Y., Bottou, L., Bengio, Y. and Haffner, P.:Gradient-based learning applied to document recognition, In: Proc. of IEEE, pp. 2278–2324, vol. 86 (1998).
6. Lee, D.D and Seung, H.S.: Learning the parts of objects by non-negative matrix factorization. J. Nature, 401, 6755, 788–791 (1999).
7. Aharon, M.,Elad, M. and Bruckstein, A.: K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation, J. IEEE Trans. Signal Process., 54, 11, 4311–4322 (2006).
8. Shekhar, M., Patel, S. and Chellappa, R.: Analysis sparse coding models for image-based classification, In: Proc. of ICIP,pp. 5207–5211, Paris, France, (2014) .
9. Guo, J., Guo, Y., Kong, X., Zhang,M. and He, R.: Discriminative analysis dictionary learning ,In: Proc. of AAAI, pp. 1617–1623, Phoenix, AZ, USA,(2016).

10. Huang, F. and Anandkumar, A.: Convolutional dictionary learning through tensor factorization, In: NIPS workshop: Feature Extraction, pp. 116–129, Montreal, Canada, (2015).
11. Garcia-Cardona, C. and Wohlberg, B.: Convolutional dictionary learning, Preprint arXiv:1709.02893 (2017).
12. Papyan,V., Romano, Y., Sulam, J. and Elad, M.: Convolutional dictionary learning via local processing, Preprint arXiv:1705.03239 (2017).
13. Ravishankar, S., Bresler, Y.: Learning sparsifying transforms. J. IEEE Trans. Signal Process. 61, 5, 1072–1086 (2013).
14. Ravishankar, S., Wen, B., and Bresler, Y.: Online sparsifying transform learning - Part I. J. IEEE J. Sel. Topics Signal Process. 9, 4, 625–636 (2015).
15. Ravishankar, S., Bresler, Y.: Online sparsifying transform learning - Part II. J. IEEE J. Sel. Topics Signal Process. 9, 4, 637–646 (2015).
16. O. Chabiron, F. Malgouyres, J. Y. Tourneret, N. Dobigeon, "Toward Fast Transform Learning", *Int. J. Comput. Vis.*, 114, 195 (2015).
17. Maggu, J. and Majumdar, A.: Kernel transform learning, J. Pattern Recognit. Lett., 98, 117–122 (2017).
18. Maggu, J. and Majumdar, A.: Greedy deep transform learning, In: Proc. of ICIP, Beijing, China, (2017).
19. Maggu, J. and Majumdar, A.: Unsupervised deep transform learning, In Proc. of ICASSP, Calgary, Canada, (2018).
20. D. Fagot, C. Fevotte, and H. Wendt, "Nonnegative Matrix Factorization with Transform Learning," *Preprint arXiv:1705.04193*, Dec. 2017.
21. H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, 2nd ed. New York: Springer, 2017.
22. Chouzenoux, E.,Pesquet, J.C. and Repetti, A.: A block coordinate variable metric forward-backward algorithm, J. Global Optim., 66, 3, 457–485 (2016).
23. Bolte, J.,Sabach, S. and Teboulle, M.: Proximal alternating linearized minimization for nonconvex and nonsmooth problems, J. Math. Program., 146, 1-2, 459–494 (2014).
24. Attouch, H., Bolte, J. and Svaiter, B. F.: Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized Gauss–Seidel methods, J. Math. Program., 137, 1, 91–129 (2013).
25. Moreau, J.J.: Proximité et dualité dans un espace hilbertien,J. Bull. Soc. Math. France, 93, 273–299 (1965).
26. Combettes, P. L. and Pesquet, J. C.: Proximal splitting methods in signal processing, Fixed-Point Algorithms for Inverse Problems in Science and Engineering, 185–212, Springer-Verlag, New York (2010).
27. Bolte, J., Combettes, P. L. and Pesquet, J. C.: Alternating proximal algorithm for blind image recovery, In: Proc. of ICIP, pp. 1673–1676, Hong Kong, China (2010).
28. Chouzenoux, E., Benfenati, A. and Pesquet, J. C.: A proximal approach for a class of matrix optimization problems,Tech. Rep.(2017). `http://arxiv.org/abs/1801.07452`
29. Bellhumer, P.N., Hespanha, J. and Kriegman, D.: Eigenfaces vs. fisherfaces: Recognition using class specific linear projection, J. IEEE Trans. Pattern Anal. Mach. Intell., 17, 7, 711–720 (1997).
30. Lee, K. C., Ho, J. and Kriegman, D.: Acquiring linear subspaces for face recognition under variable lighting, J. IEEE Trans. Pattern Anal. Mach. Intell., 27, 5, 684–698 (2005).
31. Martinez, M. and Benavente, R.: The AR face database, Tech. Rep., CVC 24 (1998).