Learning of Separable Filters by Stacked Fisher Convolutional Autoencoders

Arash Shahriari
arash.shahriari@anu.edu.au;csiro.au

Australian National University (ANU)
Commonwealth Scientific & Industrial Research Organisation (CSIRO)
Canberra, Australia

Abstract

Learning of convolutional filters in deep neural networks proves high efficiency to provide sparse representations for the purpose of image recognition. The computational cost of these networks can be alleviated by focusing on separable filters to reduce the number of learning parameters. Autoencoders are a family of powerful deep networks to build scalable generative models for automatic feature learning. Inspired by their stacked hierarchy, we introduce Fisher convolutional autoencoders to learn separable filters in a distributed architecture. These novel overcomplete autoencoders employ discriminant analysis to impose the highest possible distinction among texture classes whilst holds the minimum separation within each individual class. A distributed network of stacked Fisher autoencoders learns banks of separable filters in parallel and makes an ensemble of deep convolutional features with higher separability for a better classification. This network automatically adjusts depth of each stack with respect to the capability of its correspondent separable filter on extracting higher order convolutional features for the dataset under study. We conduct our experiments on several publicly available datasets varying in number of classes and quality of samples by using a standard implementation. Our results confirm the supremacy of our method on improving the precision of texture understanding in comparison with the recently published benchmarks.

1 Introduction

Autoencoders introduce a powerful tool for hierarchical feature learning. An autoencoder is a neural network trained to prioritize useful aspects of the input data. They generally were used for dimensionality reduction or feature learning but recently, their theoretical links with latent variable models connect them to the generative modeling [15].

In contrast to regularized autoencoders, the variational [16] and generative stochastic networks [1] learn high capacity, overcomplete encodings of the input without regularization. Stacked convolutional autoencoders are also trained using online gradient descent without additional regularization terms and properly scale to the high-dimensional inputs [22].

Deep generative models, such as Restricted Boltzmann Machines (RBM) [13], Deep Belief Networks (DBNs) [14] and Deep Boltzmann Machines (DBMs) [24] were generally trained by MCMC-based algorithms [24]. Recently, they learn by direct backpropagation and avoid the difficulties of MCMC training.

Pages 54.1-54.13
DOI: https://dx.doi.org/10.5244/C.30.54

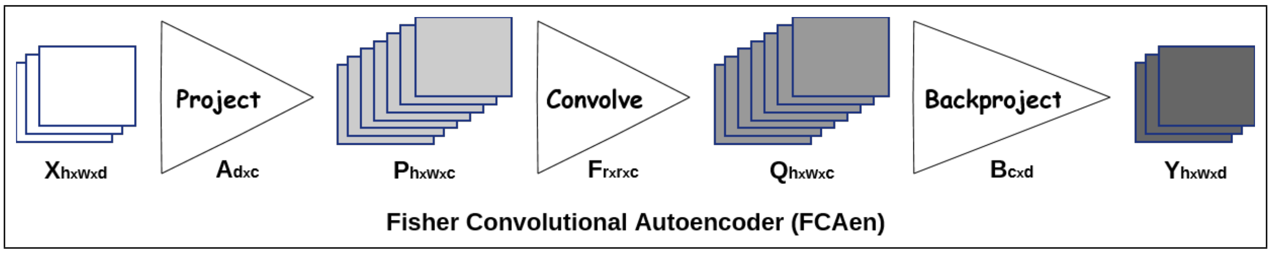
Project
Convolve
Backproject
$X_{h \times w \times d}$
$A_{d \times c}$
$P_{h \times w \times c}$
$F_{r \times r \times c}$
$Q_{h \times w \times c}$
$B_{c \times d}$
$Y_{h \times w \times d}$
Fisher Convolutional Autoencoder (FCAen)

Figure 1: A Fisher convolutional autoencoder projects its input $\mathbf{X}$ to higher dimensions, convolves them and then, backprojects to the original dimensions. It tries to expose higher distinction among classes in the output $\mathbf{Y}$ compared to the input $\mathbf{X}$.

For example, the variational autoencoders [16] or importance weighted autoencoders [3] employ a recognition network for the prediction of posterior distribution of latent variables. Generative adversarial networks [12] use an adversarial training procedure to directly shape the output distribution of the network via backpropagation [20].

In this paper, we introduce Fisher convolutional autoencoders which are arranged in a distributed network of stacks to train banks of separable filters in parallel for the purpose of texture recognition.

The first contribution of our framework is proposing of a novel discriminant analysis which holds higher class separability in a projected space spanned by the number of texture classes in dataset at hand. Our second contribution is parallel learning of separable filters by distributed discriminant analysis [30]. The third contribution of our method is automatic depth adjustment for each individual stack with respect to the distinction power of its specific separable filter.

The rest of paper is organized as follows. We present our learning framework in Section 2 followed by our experiments and discussion in Section 3 and finally, conclude in Section 4. We also provide two appendices in supplementary material to formulate the mathematics of proposed learning framework in detail.

2 Learning Framework

Inspired by the functionality of stacked overcomplete autoencoders [32], we propose our framework for the learning of separable filters by Fisher convolutional autoencoders. We stack these overcomplete autoencoders in several layers to learn each of individual separable filters and then, arrange all the stacks in a network to provide the final convolutional feature set for texture classification.

2.1 Stacked Fisher Convolutional Autoencoders

An overcomplete autoencoder is a regularized autoencoder trying to reconstruct noisy inputs based on stacking layers which are trained locally to denoise the corrupted versions of their inputs [32]. In the Fisher convolutional autoencoder, we try to employ the same reasoning but in contrast to the overcomplete autoencoder which aims at minimizing the dissimilarity between its input and output, our objective is to reconstruct an output with higher distinction by maximizing the separability between its various classes and minimizing the scatterings in each individual ones.

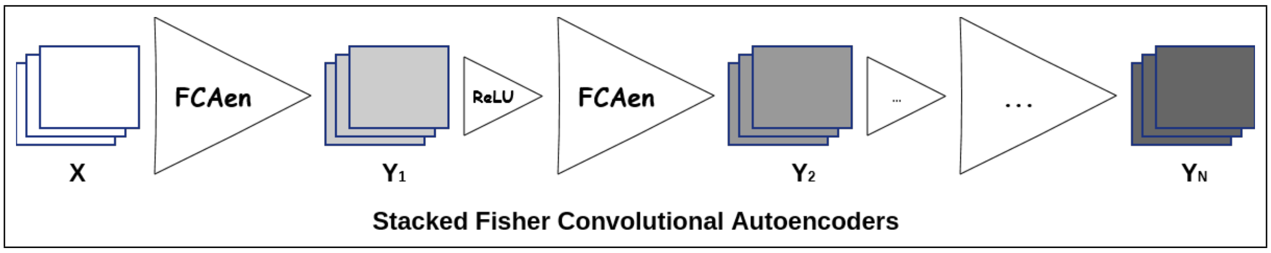

Figure 2: Stacked Fisher convolutional autoencoders are back to back autoencoders which the rectified output of each autoencoder feeds the input of the next one as $\mathbf{X}_{t+1} = g(\mathbf{Y}_t)$ until no additional class separation can be imposed to $\mathbf{Y}_t$ compared to the input $\mathbf{X}_t$.

As illustrated in Figure 1, the Fisher Convolutional Autoencoder (FACen) includes three consecutive modules called projection, convolution and backprojection. An input $\mathbf{X} \in \mathbb{R}^{h \times w \times d}$ (d color components of height h and width w) is presented to a projection module $\mathbf{A} \in \mathbb{R}^{d \times c}$ which maps it to $\mathbf{P} = \mathbf{X} \times \mathbf{A}$ such that $\mathbf{P} \in \mathbb{R}^{h \times w \times c}$ and c is the number of texture classes in dataset under study. Then, the projection set $\mathbf{P}$ convolves by filter set $\mathbf{F} \in \mathbb{R}^{r \times r \times c}$ (c texture filters of size $r \times r$) to provide $\mathbf{Q} = \mathbf{P} * \mathbf{F}$ that $\mathbf{Q} \in \mathbb{R}^{h \times w \times c}$ is set of convolutional features. Finally, the feature set $\mathbf{Q}$ moves towards a backprojection unit $\mathbf{B} \in \mathbb{R}^{c \times d}$ to give the output of autoencoder $\mathbf{Y} = \mathbf{Q} \times \mathbf{B}$ which $\mathbf{Y} \in \mathbb{R}^{h \times w \times d}$ is of the same dimensions as the input $\mathbf{X}$.

Since deep autoencoders generally benefit from depth like feedforward networks [15], we stack them as presented in Figure 2. The Fisher convolutional autoencoders in a stack are connected to each other by applying an activation function to transfer features between deep layers. We employ softsign function $g(x) = \frac{x}{1+|x|}$ to rectify the output $\mathbf{Y}$ because this formulation shows robustness to the initialization and gentle nonlinearity in comparison with other common activation functions [11].

Considering a set of filters $\mathcal{F} = \{\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_{|\mathcal{F}|}\}$ that we intend to train within each individual stack of Fisher convolutional autoencoders, a sample network of $|\mathcal{F}|$ distributed stacks of maximum N layers, is depicted in Figure 3. Each stack trains an individual filter of set $\mathcal{F}$ independently and its depth increases until no meaningful improvement in the class separation is gained.

After training of all separable filters of set $\mathcal{F}$, We ensemble the convolutional features of each stack to form $\mathbf{Y}_{\mathcal{F}} = cat(\mathbf{Y}_{1N_1}, \ldots, \mathbf{Y}_{kN_k}, \ldots, \mathbf{Y}_{|\mathcal{F}|N_{|\mathcal{F}|}})$ and employ dictionary learning (improved Fisher vectors) to feed our classifier of choice (linear support vector machine) and recognize the category of a test sample in the dataset under study.

2.2 Projection/Backprojection

In this section, we further elaborate projection/backprojection modules shown in Figure 1. Suppose that the input $\mathbf{X}$ is in depth d that contains c texture classes such that $c > d$. By projection, we mean supervised mapping to a high-dimensional space spanned by the number of classes. Considering Fisher criterion [2], we aim at minimizing the ratio of inter/intra class scatterings $\mathbf{S_w}_A$ and $\mathbf{S_b}_A$ by figuring out a projection matrix $\mathbf{A} \in \mathbb{R}^{d \times c}$ such that

$$arg\,min\, \mathcal{H}(\mathbf{A}) = \frac{tr\big(\mathbf{A}\,\mathbf{S_w}_A\,\mathbf{A}^T\big)}{tr\big(\mathbf{A}\,\mathbf{S_b}_A\,\mathbf{A}^T\big)} + \|\mathbf{I} - \mathbf{A}\,\mathbf{A}^T\|_2 \tag{1}$$

Here, $tr(.)$ is the trace operator, $\mathbf{I}$ indicates the identity matrix and $\|.\|_2$ corresponds to the L2-norm.

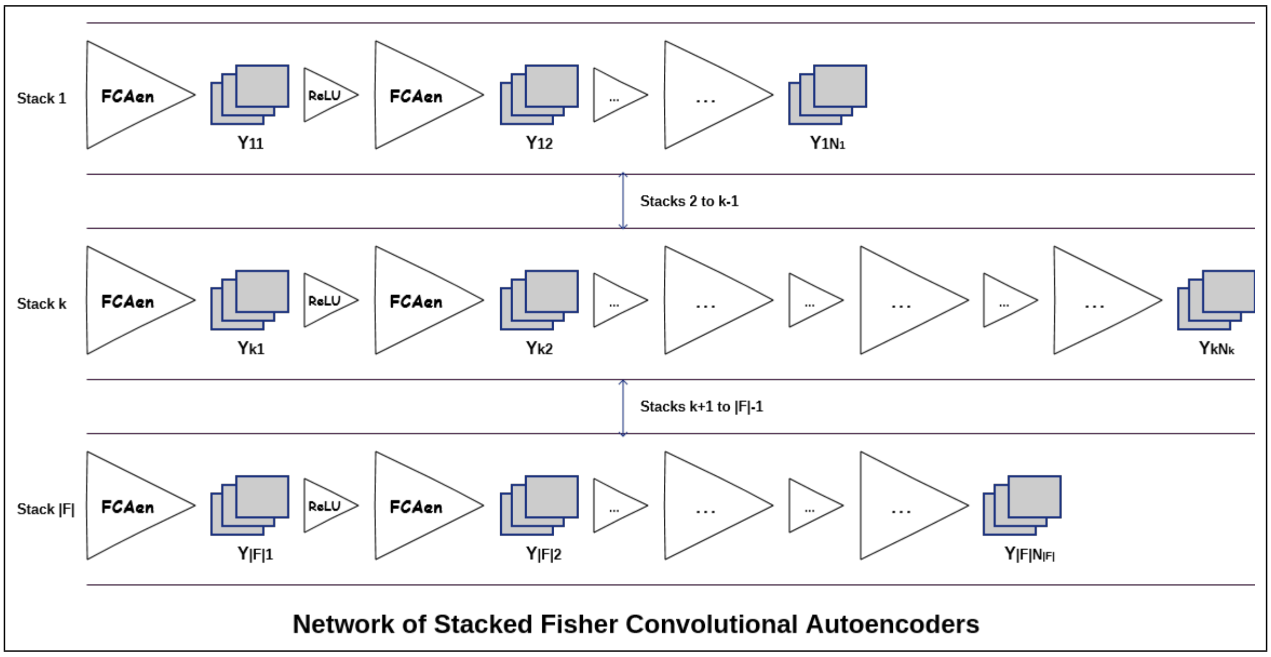

Figure 3: A network of stacked Fisher convolutional autoencoders consists of $|\mathcal{F}|$ stacks of maximum N autoencoders which are automatically arranged in different depths with respect to the distinction power of each individual filter of set $\mathcal{F}$.

The first term of Equation 1 tries to make the highest possible separability among classes. The second term is a regularization term to impose orthogonality into the projection matrix. To make Equation 1 dimensionally consistent, the scattering set $\mathcal{S}_A = \{\mathbf{S_{w}}_A, \mathbf{S_{b}}_A\}$ should belong to $\mathbb{R}^{c\times c}$ but it is currently aligned with the depth of X which holds $\mathcal{S}_A \in \mathbb{R}^{d\times d}$. This means that we are not able to employ classic discriminant analysis [10] to solve Equation 1 because it is no longer a dimension reduction problem. In contrary, this tries to increase the depth of input (d) by projection to the higher dimensions $(c > d)$.

Our solution for this inconsistency is redefinition of the scatterings based on the number of classes (c) rather than the depth of input (d). Starting from a conventional scattering set $\mathcal{S}_B \in \mathbb{R}^{d\times d}$, the within/between-class scatterings $\{\mathbf{S_{w}}_B, \mathbf{S_{b}}_B\}$ are defined as

$$\mathbf{S_{w}}_B = \sum_{j=1}^{c} \sum_{x_i \in \mathbf{C}_j} (x_i - \mu_j)(x_i - \mu_j)^T \tag{2}$$

$$\mathbf{S_{b}}_B = \sum_{j=1}^{c} (\mu_j - \bar{\mu})(\mu_j - \bar{\mu})^T \tag{3}$$

where x_i, c, μ_j and $\bar{\mu}$ are the input sample, number of classes, mean over class $\mathbf{C}_j$ and mean over all dataset, respectively.

Now, we proceed to define our proposed scattering set $\mathcal{S}_A \in \mathbb{R}^{c\times c}$ by assuming $\mathbf{S_{w}}_A$ as a square matrix of size $c \times c$ with all zeros except main diagonal entries such that

$$\mathbf{S_{w}}_A(j,j) = tr\left(\sum_{x_i \in \mathbf{C}_j} (x_i - \mu_j)(x_i - \mu_j)^T \right) \quad \forall j \in [1,c] \tag{4}$$

Then we introduce a non-singular matrix $\mathbf{\Gamma_{w}}$ as

$$vec(\mathbf{\Gamma_{w}}) = \mathbf{I} \otimes (-\mathbf{S_{w}}_A) - \mathbf{S}^{\mathbf{T}}_{\mathbf{w}B} \otimes \mathbf{I} \tag{5}$$

```
Algorithm 1 Supervised Projection
  Input: input $\mathbf{X}$ with $n$ pixels in depth $d$
  Output: optimal projection matrix $\mathbf{A} \in \mathbb{R}^{d\times c}$

  1. Compute $\mathbf{S_w}_A$ (Eq.4) and $\mathbf{S_b}_A$ (Eq.6) for $\mathbf{X} \in \mathbb{R}^{n\times d}$
  2. Set $\mathbf{A}^{(0)}$ as $c$ largest eigenvalues of $\mathbf{S_w}_A^{-1}\,\mathbf{S_b}_A$
  3. Optimize Equation 1 to compute $\mathbf{A}$
```

```
Algorithm 2 Supervised Backprojection
  Input: input $\mathbf{Q}$ with $n$ convolutional vectors in depth $c$
  Output: optimal backprojection matrix $\mathbf{B} \in \mathbb{R}^{c\times d}$

  1. Compute $\mathbf{S_w}_B$ (Eq.2) and $\mathbf{S_b}_B$ (Eq.3) for $\mathbf{Q} \in \mathbb{R}^{n\times c}$
  2. Set $\mathbf{B}^{(0)}$ as $d$ largest eigenvalues of $\mathbf{S_w}_B^{-1}\,\mathbf{S_b}_B$
  3. Optimize Equation 7 to compute $\mathbf{B}$
```

which $vec(.)$ is vectorization operator. Equation 5 is a closed form solution of Sylvester equation [18] for $\mathbf{\Gamma_w}$ by using Kronecker tensor trick or generalized eigen decomposition. Setting $\mathbf{\Gamma_b} = \mathbf{\Gamma_w}$ implies that

$$\mathbf{S_b}_A = \mathbf{\Gamma_b}\,\mathbf{S_b}_B\,\mathbf{\Gamma_b}^{-1} \tag{6}$$

In our supplementary material, we formulate above equations in detail and prove that set of eigenvectors corresponding to the largest c eigenvalues of $\mathbf{S_w}_A^{-1}\,\mathbf{S_b}_A$ is a solution for Equation 1. Although this solution can be considered as an sub-optimal projection matrix, we employ it as a starting point $\mathbf{A}^{(0)}$ to solve Equation 1. In general, Fisher criterion is the trace-of-quotient which can be solved by generalized eigenvalue method [2] but Equation 1 is the quotient-of-trace that requires different solution [8].

We also work out the closed form derivatives of Equation 1 in supplementary material. With above ingredients at hand, we employ nonlinear least squares minimization with trust region reflective and use the built-in implementation of Matlab optimization toolbox [7] to solve Equation 1. Algorithm 1 summarizes the computing of optimal projection $\mathbf{A}$.

The story for backprojection matrix $\mathbf{B} \in \mathbb{R}^{c\times d}$ is the same. Here, we solve a minimizing problem as follows

$$arg\,min\ \mathcal{Q}(\mathbf{B}) = \frac{tr\big(\mathbf{B}^T\,\mathbf{S_w}_B\,\mathbf{B}\big)}{tr\big(\mathbf{B}^T\,\mathbf{S_b}_B\,\mathbf{B}\big)} + ||\mathbf{I} - \mathbf{B}^T\mathbf{B}||_2 \tag{7}$$

This time, we initialize $\mathbf{B}^{(0)}$ by the largest d eigenvalues of $\mathbf{S_w}_B^{-1}\,\mathbf{S_b}_B$ of $\mathbf{Q}$ in Figure 1 and follow the Algorithm 2 to compute optimal backprojection $\mathbf{B}$ by the same nonlinear least squares minimization method used for $\mathbf{A}$.

It is worth mentioning that the convolutional set $\mathbf{Q}$ is in depth c and hence, we have $\mathcal{S}_B(Q) \in \mathbb{R}^{c\times c}$. This is in contrast to the input set $\mathbf{X}$ of depth d implied $\mathcal{S}_B(X) \in \mathbb{R}^{d\times d}$. It is the key difference of formulations for the projection (Equation 1) and backprojection (Equation 7) modules, although they are both solved by the same algorithm.

Algorithm 3 Optimization of Separable Filters

```
Input: input $\mathbf{X}$, projection $\mathbf{A}_k$ and backprojection $\mathbf{B}_k$
Output: optimal vertical $\mathbf{v}_k$ and horizontal $\mathbf{h}_k$ vectors
for $k = 1$ to $|F|$ do
    1. Set ${\mathbf{F}_k}^{(0)} = \mathbf{F}_k$ (Equation 8)
    2. Compute $\mathbf{S}_{\mathbf{w}k}$ (Equation 2), $\mathbf{S}_{\mathbf{b}k}$ (Equation 3) for $\mathbf{Y}_k$
    3. Optimize Equation 11 for $\mathbf{v}_k$ and $\mathbf{h}_k$
end for
```

2.3 Convolutional Separable Filters

We again consider the set of filters $\mathcal{F} = \{\mathbf{F}_1, \mathbf{F}_2, \dots, \mathbf{F}_{|\mathcal{F}|}\}$ such that the filter $\mathbf{F}_k$ is the kth filter of size $r \times r$ generated by multiplication of two vertical $\mathbf{v}_k \in \mathbb{R}^{r \times 1}$ and horizontal $\mathbf{h}_k \in \mathbb{R}^{1 \times r}$ vectors as follows

$$\mathbf{F}_k = \mathbf{v}_k \times \mathbf{h}_k \tag{8}$$

Computing of $\{\mathbf{v}_k, \mathbf{h}_k\}$ form $\mathbf{F}_k$ is straightforward [28]. Suppose that

$$[\mathbf{U}, \mathbf{S}, \mathbf{V}] = svd(\mathbf{F}_k) \tag{9}$$

then the vertical and horizontal vectors to separate $\mathbf{F}_k$ are defined as

$$\begin{aligned} \mathbf{v}_k &= \mathbf{U}_{(:,1)} \times \sqrt{\mathbf{S}_{(1,1)}} \\ \mathbf{h}_k &= \mathbf{V}^T_{(:,1)} \times \sqrt{\mathbf{S}_{(1,1)}} \end{aligned} \tag{10}$$

Here, our aim is finding $\{\mathbf{v}_k, \mathbf{h}_k\}$ which impose the maximum possible class separation in the output of our stacked Fisher convolutional autoencoders. Considering the kth stack which corresponds to $\mathbf{F}_k$, projecting the input $\mathbf{X}$ with $\mathbf{A}_k$ creates latent projected vector $\mathbf{P}_k$. We move forward by convolving with the filter $\mathbf{F}_k$ and generate latent convolutional features $\mathbf{Q}_k$. Then, $\mathbf{Q}_k$ multiplies by the backprojection matrix $\mathbf{B}_k$ to give $\mathbf{Y}_k$.

To proceed on the optimization, we fix $\{\mathbf{A}_k, \mathbf{B}_k\}$ and try to maximize the separation whilst minimize the scattering of texture classes by optimizing a least square minimization problem for the set $\mathcal{S}_k = \{\mathbf{S}_{\mathbf{w}k}, \mathbf{S}_{\mathbf{b}k}\}$ of $\mathbf{Y}_k$ (Equations 2-3). Our proposed formulation is

$$\begin{gathered} arg\,min\, \mathcal{R}_{\mathcal{F}}(\mathbf{v}_k, \mathbf{h}_k) = \\ \Big(1 - log\big[tr(\mathbf{S}_{\mathbf{w}k})\big]\Big)^2 + \Big(1 - log\big[tr(\mathbf{S}_{\mathbf{b}k})\big]\Big)^{-2} + \left(1 - log\left[\frac{tr(\mathbf{S}_{\mathbf{w}k})}{tr(\mathbf{S}_{\mathbf{b}k})}\right]\right)^2 \end{gathered} \tag{11}$$

which the first two terms are smoothing functions to impose such a symmetry to $\mathcal{R}_{\mathcal{F}}$ that avoids biases towards majority/minority texture classes, respectively. Here, the logarithm function improves the overall convergence rate. The solution of Equation 11 is a set of optimal vectors $\{\mathbf{v}_k, \mathbf{h}_k\}$ that finally provides the optimal separable convolutional filter $\mathbf{F}_k$. To solve Equation 11, we employ least squares minimization with Levenberg-Marquardt algorithm implemented in Matlab optimization toolbox [7]. We wrap up this optimization procedure in Algorithm 3.

3 Experiments

In our experiments, we employ three well-known local texture descriptors (LM, MR, Schmid) and train them in our proposed learning framework for five publicly available texture datasets (UIUC, KTH-TIPS2-a, KTH-TIPS2-b, FMD, DTD). We compare our results with three state-of-the-art performances in literature [6] which are Improved Fisher Vector on dense SIFT [4] besides DeCAF(FC6) [9] and VGG-VD [27] pre-trained on ILSVRC [23] dataset. For implementation, we develop our method inside Oxford Visual Geometry Group's code for texture understanding [5] and report mean accuracy of recognition averaged over standard number of splits according to the evaluation protocols.

3.1 Texture Filters

The separable filters are initialized by three banks of texture filter including 99 filters with size 49×49. The first filter bank is Leung-Malik (LM) [19] with 36 first/second derivatives of Gaussian filters at three scales $\{\sqrt{2}, 2, 2\sqrt{2}\}$ and six orientations $\{\frac{\pi}{6}, \frac{\pi}{3} \ldots, \pi\}$, eight LoG and four Gaussian filters at scales $\{\sqrt{2}, 2, 2\sqrt{2}, 4\}$. The second filter bank is Maximum Response (MR) [31] consisting of 36 filters at three scales $\{1, 2, 4\}$ and six orientations added to two isotropic Gaussian and LoG filters. The third filter bank is Schmid (S) [25] that contains 13 rotationally invariant filters with $\sigma \in \{2, 4, 6, 8, 10\}$ and $\tau \in \{1, 2, 3, 4\}$.

3.2 Texture Datasets

We pick five texture datasets with enough room to show the advantage of our method over their state-of-the-art performances in literature. UIUC texture database [17] contains 1000 images (40 samples, 25 classes) in grayscale. KTH-TIPS2-a and KTH-TIPS2-b [21] stand for Textures under varying Illumination, Pose and Scale which the latter consists of 4572 images (4 samples, 108 images per sample and 11 categories) and the former uses only 72 images for 4 out of 44 samples. We follow [29] for evaluation which images on one sample are used to train and the other three samples to test. Flicker Material Dataset (FMD) [26] includes 1000 images (100 per category, 10 categories) manually selected from the Flickr. Describable Texture Dataset (DTD) [5] contains 5640 annotated texture images with one or more adjectives in a vocabulary of 47 English words (120 representative per attribute). There are 10 preset splits into equally-sized training, validation and test sets.

3.3 Experimental Setup

For the experiments, we follow [5] by computing local image descriptors and encoding them into a visual dictionary. The current state-of-the-art descriptors are 128-dimensional Dense SIFT features (DSIFT) computed for bins of size 6×6 pixels at scales $\{1, \frac{\sqrt{2}}{2}, \frac{1}{2}, \frac{\sqrt{2}}{4}, \frac{1}{4}\}$. The descriptors are soft quantized by Gaussian Mixture Model (GMM) and normalized to generate Improved Fisher Vectors (IFV). After normalization, we train a standard linear SVM solver and use the validation set to find its regularization parameter. We replace our descriptors (Ours) with DSIFT and proceed the above pipeline for texture recognition.

3.4 Results & Discussion

We report the results in four tables comparing our performance on texture recognition to DSIFT in Table 1, DeCAF in Table 2, VGG in Table 3 and finally, summarizing the best outcomes in Table 4. Bold values are the best performances on each experiment.

Table 1 represents DSIFT features compared to our descriptors (**Proposed**). It is clear that, we outperform in all texture datasets except UIUC. It is due to the fact that this is a grayscale dataset and hence, our paradigm cannot take the advantage of orthogonality in standard color spaces for imposing a powerful separation among texture classes.

Dataset	DSIFT [4]	**Proposed**
UIUC	$\mathbf{97.2 \pm 0.8}$	90.1 ± 0.8
KTH-a	82.5 ± 5.3	$\mathbf{85.5 \pm 4.9}$
KTH-b	69.3 ± 0.9	$\mathbf{70.1 \pm 0.7}$
FMD	58.1 ± 1.7	$\mathbf{71.8 \pm 2.2}$
DTD	58.6 ± 2.0	$\mathbf{59.1 \pm 1.3}$

Table 1: Mean accuracy of texture recognition for DSIFT and our descriptors.

In Tables 2-3, we compare performances of deep features (DeCAF, VGG) and their joints with DSIFT and our descriptors. It is worth mentioning that both of these deep models were pre-trained on ILSVRC [23] dataset with large number of samples and object classes which eventually benefits them by better generalization on larger texture datasets.

Dataset	DeCAF [5]	DSIFT+DeCAF [4]	**Proposed+DeCAF**
UIUC	94.2 ± 1.1	$\mathbf{99.0 \pm 0.5}$	96.1 ± 0.7
KTH-a	78.4 ± 2.0	84.7 ± 1.5	$\mathbf{86.0 \pm 2.3}$
KTH-b	70.7 ± 1.6	76.2 ± 3.1	$\mathbf{82.3 \pm 0.9}$
FMD	60.7 ± 2.0	65.5 ± 1.3	$\mathbf{84.3 \pm 2.8}$
DTD	54.8 ± 0.9	66.7 ± 0.9	$\mathbf{80.6 \pm 2.3}$

Table 2: Mean accuracy of texture recognition for DeCAF (pre-trained on ILSVRC).

Dataset	VGG [6]	DSIFT+VGG [4]	**Proposed+VGG**
UIUC	97.0 ± 0.7	$\mathbf{99.3 \pm 0.4}$	96.3 ± 0.1
KTH-a	77.8 ± 1.8	$\mathbf{83.6 \pm 1.7}$	82.2 ± 7.9
KTH-b	75.4 ± 1.5	$\mathbf{81.1 \pm 2.4}$	80.5 ± 0.7
FMD	77.4 ± 1.8	82.4 ± 1.5	$\mathbf{85.7 \pm 3.0}$
DTD	62.9 ± 0.8	74.7 ± 1.0	$\mathbf{85.9 \pm 1.4}$

Table 3: Mean accuracy of texture recognition for VGG (pre-trained on ILSVRC).

Again, we outperform on all datasets except UIUC in spite of the great improvements compared to the Table 1. On KTH-TIPS2-a and KTH-TIPS2-b, our joint performance with DeCAF is better than VGG. The reason is quality of texture images in KTH datasets captured on controlled lighting conditions and fix distances [21] that generates more homogeneous features for each individual class of textures.

In contrast, our combination with VGG performs better than DeCAF on FMD and DTD datasets because here, the texture images were gathered from web with a huge variety in lighting and capturing conditions. It seems that VGG generalizes better for uncontrolled conditions due to its higher depth in comparison with the DeCAF deep architecture.

Table 4 summarizes the current state-of-the-arts (SOA) and our best results from above tables. It confirms that our stacked Fisher convolutional autoencoders is quite successful on imposing distinction among highly-correlated texture patterns especially the ones which were captured in uncontrolled conditions.

Dataset	SOA [4]	**Ours**
UIUC	$\mathbf{99.3 \pm 0.4}$	96.3 ± 0.1
KTH-a	84.7 ± 1.5	$\mathbf{86.0 \pm 2.3}$
KTH-b	81.1 ± 2.4	$\mathbf{82.3 \pm 0.9}$
FMD	82.4 ± 1.5	$\mathbf{85.7 \pm 3.0}$
DTD	74.7 ± 1.0	$\mathbf{85.9 \pm 1.4}$

Table 4: Mean accuracy of texture recognition for others (SOA) and our framework (Ours).

Figure 4 illustrates some examples of initial filters and their corresponding separable ones learned by our proposed framework. The first column includes classic texture descriptors from mentioned filter banks. and the rest are trained separable filters for each datasets.

It can be seen that for some filters, the changes across various datasets is smaller than the others. This means that they are responsible to extract common features in texture patterns. The filters with considerable deformations usually connect to deeper stacks of autoencoders that capture higher order representations. For initial symmetric filters, the trained outputs are not necessarily symmetric everywhere, although they preserve the symmetry quite well in some datasets.

4 Conclusion

The supremacy of deep convolutional neural networks for automatic feature learning leads us to introduce a novel family of overcomplete autoencoders for the training of separable texture filters. Our stacked Fisher convolutional autoencoders employ a modified form of linear discriminant analysis to impose higher distinction among classes in the dataset under study. This lets us to arrange parallel stacks of them to train each individual separable filter independently. They also expand in different depths based on the power of corresponding filters to extract better sparse representations. Our experiments conducted for several texture datasets over a standard platform prove the advantage of our learning framework over other deep local descriptors in literature.

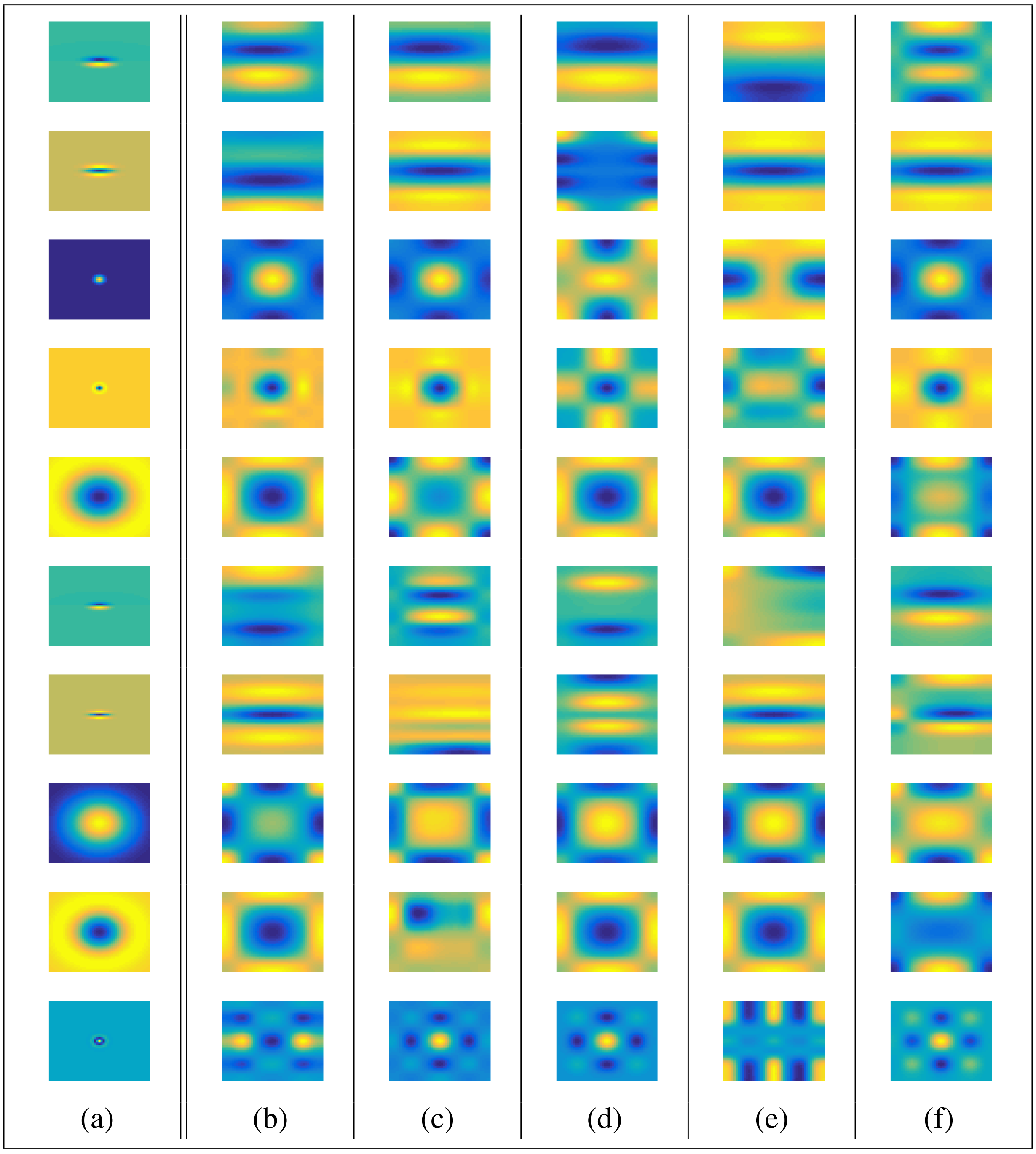

Figure 4: Examples of (a) initial filters and their corresponding separable filters learned for (b) UIUC; (c) KTH-TIPS2-a; (d) KTH-TIPS2-b; (e) FMD; (f) DTD datasets.

References

[1] Yoshua Bengio, Eric Thibodeau-Laufer, Guillaume Alain, and Jason Yosinski. Deep generative stochastic networks trainable by backprop. *arXiv preprint arXiv:1306.1091*, 2013.

[2] Christopher M Bishop. Pattern recognition. *Machine Learning*, 2006.

[3] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

[4] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

[5] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3606–3613, 2014.

[6] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, and Andrea Vedaldi. Deep filter banks for texture recognition, description, and segmentation. *International Journal of Computer Vision*, pages 1–30, 2015.

[7] Thomas F Coleman and Yuying Li. A reflective newton method for minimizing a quadratic function subject to bounds on some of the variables. *SIAM Journal on Optimization*, 6(4):1040–1058, 1996.

[8] John P Cunningham and Zoubin Ghahramani. Linear dimensionality reduction: Survey, insights, and generalizations. *Journal of Machine Learning Research*, 2015.

[9] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*, 2013.

[10] Reinosuke FUKUNAGA. Statistical pattern recognition. 1990.

[11] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256, 2010.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[13] Geoffrey E Hinton. Learning to represent visual input. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 365(1537):177–184, 2010.

[14] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[15] Yoshua Bengio Ian Goodfellow and Aaron Courville. Deep learning. Book in preparation for MIT Press, 2016. URL `http://goodfeli.github.io/dlbook/`.

[16] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589, 2014.

[17] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. A sparse texture representation using local affine regions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(8):1265–1278, 2005.

[18] Sang-Gu Lee and Quoc-Phong Vu. Simultaneous solutions of sylvester equations and idempotent matrices separating the joint spectrum. *Linear Algebra and its Applications*, 435(9):2097–2109, 2011.

[19] Thomas Leung and Jitendra Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International journal of computer vision*, 43(1):29–44, 2001.

[20] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian Goodfellow. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[21] P Mallikarjuna, Alireza Tavakoli Targhi, Mario Fritz, Eric Hayman, Barbara Caputo, and Jan-Olof Eklundh. The kth-tips2 database, 2006.

[22] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *Artificial Neural Networks and Machine Learning–ICANN 2011*, pages 52–59. Springer, 2011.

[23] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[24] Ruslan Salakhutdinov and Geoffrey E Hinton. Deep boltzmann machines. In *International conference on artificial intelligence and statistics*, pages 448–455, 2009.

[25] Cordelia Schmid. Constructing models for content-based image retrieval. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 2, pages II–39. IEEE, 2001.

[26] Lavanya Sharan, Ruth Rosenholtz, and Edward Adelson. Material perception: What can you see in a brief glance? *Journal of Vision*, 9(8):784–784, 2009.

[27] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[28] Amos Sironi, Bugra Tekin, Roberto Rigamonti, Vincent Lepetit, and Pascal Fua. Learning separable filters. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 37(1):94–106, 2015.

[29] Radu Timofte and Luc J Van Gool. A training-free classification framework for textures, writers, and materials. In *BMVC*, volume 13, page 14, 2012.

[30] S Valcarcel Macua, Pavle Belanovic, and Santiago Zazo. Distributed linear discriminant analysis. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 3288–3291. IEEE, 2011.

[31] Manik Varma and Andrew Zisserman. Texture classification: Are filter banks necessary? In *Computer vision and pattern recognition, 2003. Proceedings. 2003 IEEE computer society conference on*, volume 2, pages II–691. IEEE, 2003.

[32] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11:3371–3408, 2010.