Sparse coding and normalization for deep Fisher score representation

Sixiang Xu, Damien Muselet, Alain Trémeau

Computer Vision and Image Understanding 220 (2022) 103436

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Sparse coding and normalization for deep Fisher score representation

Sixiang Xu *, Damien Muselet, Alain Trémeau

Univ Lyon, UJM-Saint-Etienne, CNRS,Institut Optique Graduate School, Laboratoire Hubert Curien UMR 5516, F-42023, Saint-Etienne, France

ARTICLE INFO

Communicated by Nikos Paragios

Keywords:
Fisher score
Sparse coding
Orderless pooling
Square root normalization
Classification

ABSTRACT

Fisher scores have been shown to be accurate global image features for classification. However, their performance is very dependent on the quality of the input features as well as the normalization steps applied to them. In this paper, we propose to embed the Fisher scores in an end-to-end trainable deep network by concentrating on two elements: adapting the encoding to the deep features and normalizing the extracted second-order statistics. Therefore, we make use of a deep sparse coding module that allows to sample the center of each Gaussian function from a learned subspace and thus to better fit the high dimensional data distribution. Second, we introduce a new normalization module that computes an approximate square root matrix normalization well adapted to the Fisher scores. These processing steps are embedded in a deep network so that all the modules work together for the sole purpose of improving classification performance. Experimental results show that this solution outperforms many alternatives in the context of material, indoor scene and fine-grained image classification.

1. Introduction

Deep neural networks have emerged as an essential solution for performing classification tasks. In these networks, convolutional layers extract accurate local features that are pooled to a global feature vector which is sent to fully connected layers for classification. The first networks neglected the pooling step and directly sent the set of local features in the dense layers (Simonyan and Zisserman, 2015), while the series of ResNet apply a global average pooling to decrease the dimension of the global feature vector and hence reduce the number of parameters of the network (He et al., 2016). Orderless pooling was widely used before convolutional neural networks (CNN) with bags of visual words (BOW) (Lazebnik et al., 2006), VLAD (Jégou et al., 2012) or Fisher vectors (Sánchez et al., 2013) and has shown to provide good results when applied to CNN features (Cimpoi et al., 2015). Among them, Fisher vectors (FV)[1] were the most promising because they generalize VLAD and BOW. The main idea of FV is to model the distribution of the training data with a Gaussian mixture and to characterize each data point with the derivatives over the model parameters. It appears that two main steps are crucial in such an approach (Sánchez et al., 2013): the data distribution has to be accurately fitted by the Gaussian mixture and the provided second-order statistics have to be carefully normalized. In this paper, we propose to embed the Fisher representation in an end-to-end trainable network by concentrating on these two steps.

First, a Gaussian Mixture Model (GMM) seems not to be well adapted to the deep local features since they are lying in a high dimensional space and the space requires too many Gaussians to be accurately modeled (Liu et al., 2014). The authors proposed a solution to overcome this problem which consists in sampling the center of each Gaussian from a subspace and therefore benefiting from an infinite number of Gaussians to fit the data distribution. And they showed that this problem can be solved by a classical sparse coding method. Unfortunately, their approach cannot take advantage of end-to-end training of the feature extraction, the pooling and the classification layers. To cope with this problem, we propose in this paper, to make use of a deep sparse coding module proposed in the work (Gregor and LeCun, 2010).

Second, a recent study has shown that the normalization of second-order statistics has a strong impact on classification performance (Lin and Maji, 2017). The authors proposed in particular to use a square-root matrix normalization combined with element-wise square-root and l_2 normalization for bilinear pooling. Unfortunately, unlike the bilinear pooling (Lin et al., 2015), our Fisher score representation does not provide a square matrix, thus rendering the solution (Lin and Maji, 2017) unusable. In this paper, we propose to adapt the square-root matrix normalization to non-square matrices and to embed this original module in a deep network.

By combining these two main contributions, we propose an original end-to-end trainable deep network that extracts accurate features from

* Corresponding author.
E-mail address: sixiang.xu@univ-st-etienne.fr (S. Xu).

[1] FV is the Fisher score scaled by the inverse square root of Fisher Information Matrix (FIM). Since the impact of FIM is small (Perronnin et al., 2010), there is not a large difference between FV and Fisher score.

https://doi.org/10.1016/j.cviu.2022.103436
Received 31 July 2021; Received in revised form 1 April 2022; Accepted 13 April 2022
Available online 26 April 2022
1077-3142/

images, pools them into a deep Fisher representation and normalizes the representation. By backpropagating the gradient of the classification loss, we can make all these modules collaborate with the sole objective of improving the performance of the classification task. Experimental tests on three different datasets and three different backbone architectures show that our solution outperforms many alternatives.

This paper is an extended version of our previous work (Xu et al., 2021), called hereafter E2E-SCF for End-to-End Sparse Coding Fisher score, where only the sparse coding has been addressed. Our contributions are fourfold:

- We propose to produce Fisher score representation via sparse coding in an end-to-end trainable deep network.
- We make use of a deep sparse coding module that allows to better fit high dimensional data distribution.
- We introduce a new normalization module and mean vector subtraction which improve the Fisher score representation.
- Our solution outperforms many alternatives on three datasets for material, indoor scene and fine-grained image classification.

2. Related works

2.1. Orderless pooling

Orderless pooling was widely used before the emergence of CNN-based solutions. The most popular approaches were based on bags of visual words (BOW) (Lazebnik et al., 2006), VLAD (Jégou et al., 2012) or Fisher vectors (Sánchez et al., 2013). Inspired by these early methods, some works have evaluated the Fisher vectors or VLAD from deep features for texture or image classification (Cimpoi et al., 2015). They show improvements over the SIFT-based counterparts but, in their workflow, the dictionary or Gaussian mixture model is learned independently from the deep features and the classifier, providing opportunities for significant improvements.

Thus, the next works have focused on embedding orderless pooling in deep networks to allow end-to-end training. Passalis and Tefas (2017) have inserted a Bag-of-Features pooling in deep neural networks thanks to radial basis function neurons. The output of the pooling module is a histogram of the visual words (0th order statistics) learned on the training set. And their variations also achieve remarkable performance in other tasks, such as color constancy (Laakom et al., 2020), visual information analysis (Krestenitis et al., 2020) and human action recognition (Yang et al., 2020).

Instead of counting the occurrences of the visual words in one image, VLAD-based approaches aggregate the residuals between the local features and their nearest visual words (1st order statistics). NetVLAD (Arandjelović et al., 2016) is the first network that implements VLAD and allows end-to-end training for image retrieval tasks while Deep Ten and its variants transform VLAD as a residual module for image classification (Zhang et al., 2017; Hu et al., 2019; Mao et al., 2021). It has been shown that first-order statistics are more accurate to characterize images in classification tasks and the Fisher vectors go further by using first- and second-order statistics. Deep FisherNet (Tang et al., 2019) is an embedded implementation of the GMM Fisher vector. NetFV (Lin et al., 2017) extends NetVLAD by appending the second-order statistics. End-to-end trainable Fisher vectors are also applied to action recognition (Wang et al., 2019; Wang and Koniusz, 2021). The main disadvantage of all these approaches is that they rely on a limited number of codewords or Gaussian centers, which prevents accurate modeling of the data distribution in the high-dimensional deep feature spaces (Liu et al., 2014).

One interesting solution to cope with this problem has been proposed in the work (Li et al., 2017). The authors compute Fisher scores from a mixture of factor analyzers (MFA), instead of the classical GMM. Their solution is embedded in a deep network which is trainable end-to-end. The idea of MFA is to approximate the data manifold by low dimensional linear spaces and, in this sense, is similar to the idea of sparse coding (Liu et al., 2014). Nevertheless, even if the MFA module is embedded in a deep network, the authors show that an accurate initialization of the weights of the network is required to obtain good performance. This initialization consists in running an Expectation–Maximization algorithm on the set of local features that have to be saved in memory. Furthermore, it appears that this second-order representation has high computation costs, requires a high number of parameters to learn and occupies a very large memory space (500k dimensions which is more than the image itself) (Jacob et al., 2019).

Another group of second-order pooling works is based on bilinear pooling. For example, B-CNN is also an end-to-end trainable network and aggregates feature vectors by average-pooling their outer products (Lin et al., 2017). Since this pooled representation always has a large size, the approaches SMSO (Yu and Salzmann, 2018), RUN (Yu et al., 2020) and SRM (Yu et al., 2021) propose to compress the bilinear pooled features and improve the classification performance. The results of these methods will be compared with ours in the following experiments.

Our method is inspired by SCFVC (Liu et al., 2014), detailed in the next section. More recently, these authors have also proposed an improved version of their work, called HSCFV (Liu et al., 2017). It uses two dictionaries to encode input features and consequently, doubles the dimension size of the Fisher score. Nevertheless, their approach is not embedded in a deep CNN for end-to-end training. Sparse coding is also used in CNN-DL (Liu et al., 2018) and SCN (Sun et al., 2019) for extracting image features, but these methods do not use second-order statistics to characterize the images.

Our method combines all the benefits of these previous solutions: it is embedded in an end-to-end trainable network, it samples an infinite number of Gaussian centers from a learned subspace and it does not require any heavy computation or storage to initialize the weights.

2.2. Normalization

As a post-processing step, after orderless pooling, normalization plays an important role in improving the performances. Perronnin et al. (2010) found that Fisher vector representation is degraded by burstiness issues where important but relatively rare visual features are overwhelmed by those that are more frequent. To alleviate this problem, some papers propose element-wise signed square rooting and L2-normalization (Perronnin et al., 2010; Arandjelovic and Zisserman, 2013). This normalization combination is also widely adopted in several successive orderless pooling works (Arandjelović et al., 2016; Lin et al., 2017; Liu et al., 2014).

Besides the burstiness issue, Lin and Maji (2017) argued that the output of bilinear pooling should be normalized by matrix-logarithm functions in order to preserve the distances between elements in the manifold. Such normalization has been applied with success in many works (Carreira et al., 2012; Ionescu et al., 2015; Huang and Van Gool, 2017) with linear classifiers for semantic segmentation and image classification. The logarithm scales the eigenvalues in the Singular Value Decomposition (SVD) of a Symmetric Positive Definite (SPD) matrix A as $\log(A) = U \log(\Sigma) U^T$. Unfortunately, the SVD decomposition is computed inefficiently on GPUs (Lin and Maji, 2017), slowing down the network inference speed. Nevertheless, Lin and Maji (2017) proposed a fast alternative approach with comparable performances and based on a variant of Newton iterations. This solution approximates square-root matrix and can be embedded in a network that can be trained end-to-end.

Unfortunately, this approach is exclusively designed for SPD matrices such as the outputs of the bilinear pooling but it cannot directly be applied to our Fisher representations which are rectangular and non-symmetric matrices. Thus, we propose, in this paper, a new normalization step for such second-order statistics matrices and it can also be embedded in a deep network.

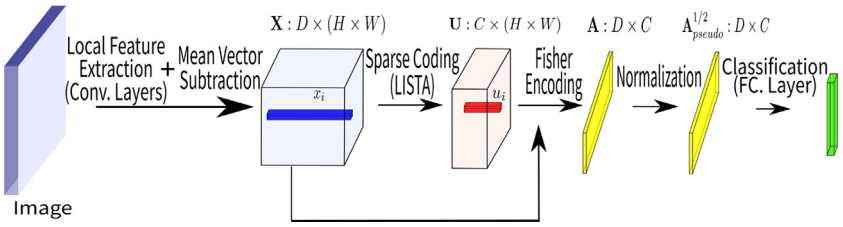

Fig. 1. Workflow of the proposed end-to-end trainable solution. First, deep features x_i are extracted with a classical CNN backbone and normalized (zero mean, see Section 3.1.4). Then they are encoded into their sparse codes u_i with a sparse coding module called LISTA presented in Section 3.1.2. Next, a Fisher score representation is produced (Eq. (5)) and normalized with our proposed approach detailed in Section 3.2.2.

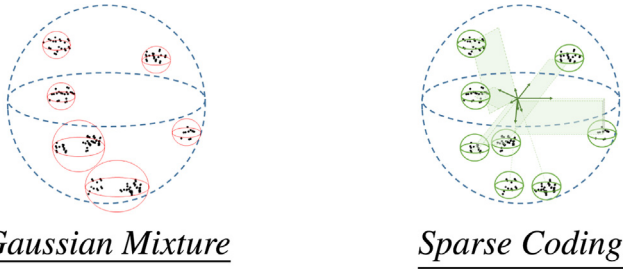

Fig. 2. Some data in a high dimensional space (illustrated by the sphere). Left: With GMM the data distribution is not well fitted because of the limited number of Gaussians. Right: With Sparse Coding, the Gaussian centers are encoded sparsely in an adapted basis (green arrows) allowing to create an unlimited number of Gaussians and so to fit better the data distribution. The sparsity is illustrated by the low number of basis required to encode each center position (lines, planes or parallelograms).

3. Our approach

Fig. 1 illustrates the complete workflow of our solution whose successive steps are detailed in this Section. Our network starts with a pre-trained backbone of convolutional layers, on top of which an iterative sparse coding module called LISTA is applied, detailed in Section 3.1.2. Then, a Fisher score is extracted from these features and normalized with our proposed solution. Lastly, a dense layer provides the predicted categories.

3.1. Sparse Fisher encoding

3.1.1. From subspace sampling to sparse coding

To increase the number of Gaussians that model the distribution of the data, we take advantage of the idea (Liu et al., 2014) that samples the Gaussian centers in a subspace spanned by a set of bases. Each mean vector is encoded in this dictionary B with a code u drawn from a zero-mean Laplacian distribution (to enforce sparsity). Then each local feature vector x extracted from the images and associated with the code u is drawn from a Gaussian distribution $\mathcal{N}(Bu, \Sigma)$ centered on Bu. Fig. 2 illustrates the interest of this approach.

Then, assuming a constant and diagonal covariance matrix as σ and using pointwise maximum to approximate the integral of the distribution, Liu et al. (2014) showed that the logarithm of the likelihood of x can be estimated as:

$$log(P(x|B)) = \min_u \frac{1}{\sigma^2}\|x - Bu\|_2^2 + \lambda\|u\|_1, \tag{1}$$

where λ is the scale of the Laplacian distribution of u.

Interestingly, this equation represents the classical problem of sparse coding. Liu et al. (2014) proposed to use an off-the-shelf sparse coding solver to learn the dictionary B and infer the code u. Obviously, making use of such an independent solver is a good solution to minimize the reconstruction error of x with a sparse code, but it neglects the main goal which is to improve the performance in the classification task.

Hence, we propose in the next section to embed a sparse coding module in a deep neural network that is trained end-to-end. The main advantage of such an approach is that it is learning a dictionary and sparse codes that are accurate to discriminate the different categories in the current dataset.

3.1.2. Embedding sparse coding with LISTA

Our aim is to find a solution for the following equation:

$$\min_u f(u) + \lambda\|u\|_1 \tag{2}$$

where $f(u) = \|x - Bu\|_2^2$, x is a data point, B is the dictionary and u is the sparse code of x.

One way to solve this equation is to resort to an Iterative Shrinkage Thresholding Algorithm (ISTA) (Daubechies et al., 2004) that iteratively approximates the solution with:

$$u_k = \mathcal{T}_{\lambda t_k}(u_{k-1} - t_k \nabla f(u_{k-1})), \tag{3}$$

where $\mathcal{T}_\alpha$ is a component-wise vector shrinkage function such that $[\mathcal{T}_\alpha(v)]_i = (|v_i| - \alpha)_+ sign(v_i)$, t_k is the step size at iteration k and ∇ is the gradient operator.

Evaluating the gradient of $f(u)$ defined above, we get:

$$\begin{aligned} u_k &= \mathcal{T}_{\lambda t_k}(u_{k-1} - 2t_k B^T(Bu_{k-1} - x)), \\ &= \mathcal{T}_{\lambda t_k}((I - 2t_k B^T B)u_{k-1} + 2t_k B^T x), \\ &= \mathcal{T}_{\lambda t_k}(Su_{k-1} + Wx), \end{aligned} \tag{4}$$

where $S = I - 2t_k B^T B$ and $W = 2t_k B^T$.

This equation can be illustrated as a recurrent block diagram as in Fig. 3, left. Fortunately, Gregor and LeCun (2010) proposed a fast approximation of ISTA called Learned ISTA (LISTA). This is an unfolded version of ISTA with a fixed number of iterations and it can be plugged into a neural network to provide a sparse code (see Fig. 3, right). Embedding this LISTA module in our CNN is an effective way to learn a dictionary and sparse codes that help to discriminate between the categories of the current dataset.

3.1.3. Dictionary based Fisher encoding

When a classical GMM is used to model the data distribution, the Fisher score is based on the partial derivatives of the posterior probabilities with respect to the weights, the mean and the standard-deviation parameters of the model (Sánchez et al., 2013). In our case, we use a particular Fisher encoding, which is evaluated as the partial derivative of the log probability of the local features with respect to the dictionary itself:

$$\frac{\partial log(P(x|B))}{\partial B} = \frac{\partial \frac{1}{\sigma^2}\|x - Bu^*\|_2^2 + \lambda\|u^*\|_1}{\partial B} = (x - Bu^*)u^{*T}, \tag{5}$$

where the sparse code $u^* = \text{argmax}_u\, P(x|u, B)P(u)$ (see Liu et al. (2014) for details).

Sparse Fisher encoding module is very easy to insert into our deep network and provides the pooled features. These features are then sent, after normalization, to the last fully connected layer for classification. All these modules are constituting our CNN which can be trained end-to-end (see Fig. 1).

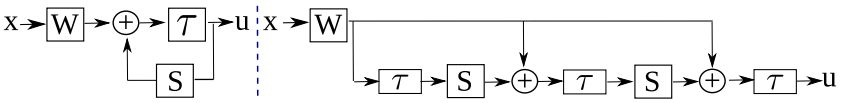

Fig. 3. Block diagrams of ISTA (left) and LISTA (right). ISTA evaluates the sparse code u of x with the iterative process detailed in Eq. (4). The shrinkage function is denoted by $\mathcal{T}$ in this equation and in the block diagrams. LISTA is an unfolded version of ISTA (2 iterations here) that can be embedded in an end-to-end trainable network. In our framework, the matrices S and W are learned and initialized thanks to our *warm-up* step, detailed in Section 4.3.

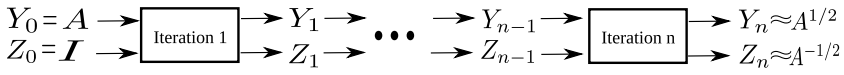

Fig. 4. Square-root matrix estimation with the Newton method. The inputs are a symmetric positive definite (SPD) matrix A and the identity I. After n iterations the outputs Y_n and Z_n converge into the square root matrix $A^{1/2}$ and the inverse square root matrix $A^{-1/2}$ of the input matrix A. Each iteration corresponds to Eq. (6).

3.1.4. Mean vector subtraction

It is worth mentioning that, for each image, the input local feature vectors x_i are centered to have a zero mean before applying previous sparse Fisher encoding $x_i' = x_i - \frac{1}{HW}\left(\sum_{i=1}^{HW} x_i\right)$, where HW is the number of feature vectors x_i.

This pre-processing is similar to an instance normalization (Ulyanov et al., 2016) without additive parameters to learn. It improves the generalization property of the model as observed in the experimental results (see more details in Section 4.4).

3.2. Fisher score normalization

As mentioned earlier, the second-order statistics tend to excessively emphasize very few coordinates, ignoring potential important features (Lin and Maji, 2017). To cope with this problem, many normalization solutions have been proposed. In this paper, we take advantage of the approach (Lin and Maji, 2017) to normalize our Fisher scores. Below, we first detail this approach and then, explain its extension to non-square matrices.

3.2.1. Bilinear square matrix normalization

Assuming that a network backbone provides a feature map $X \in \mathbb{R}^{D\times H\times W}$ (see Fig. 1), where D, H, W are the depth, height and width. This set of local feature vectors can be pooled into a global feature vector by using bilinear pooling (Lin et al., 2015), regardless of their spatial coordinates. Therefore, the feature map X is reshaped to a 2D matrix of size $D\times HW$ where each column x_i is a local feature vector of D dimensions. Then, the output of the bilinear pooling is evaluated as $A = \frac{1}{HW}\left(\sum_{i=1}^{HW} x_i x_i^T\right)$, where A is a symmetric positive definite (SPD) matrix of size $D \times D$. While element-wise square-root normalization helps improving the performance of the complete framework, Lin and Maji have shown that the results can be further boosted by applying a spectral normalization, i.e. scaling the eigenvalues of the associated covariance matrix (Lin and Maji, 2017). One way to do that is to transform the matrix A to its square-root $A^{1/2} = U\Sigma^{1/2}U^T$, where $A = U\Sigma U^T$ is the singular value decomposition (SVD) of A.

However, the computation of the SVD is poorly supported on GPUs and Lin and Maji suggest applying a variant of the Newton method to solve $F(Z) = Z^2 - A = 0$. Their approach is an iterative process where each iteration is as follow:

$$Y_{k+1} = \frac{1}{2}Y_k(3I - Z_kY_k) \quad \text{and} \quad Z_{k+1} = \frac{1}{2}(3I - Z_kY_k)Z_k. \tag{6}$$

By initializing $Y_0 = A$ and $Z_0 = I$, Y_k and Z_k converge to $A^{1/2}$ and $A^{-1/2}$ (see Fig. 4) in very few iterations (even one). And the process requires only matrix multiplications (no inverse).

This matrix normalization clearly improves the accuracy and efficiency of the bilinear pooling CNN (Lin and Maji, 2017), but it cannot be directly applied to our Fisher representation, as shown in the next section.

3.2.2. Matrix normalization for Fisher score representation

As previously explained, our Fisher representation is also a second-order matrix that could benefit from spectral normalization. From Eq. (5), we know that it is expressed as:

$$A = \frac{1}{HW}\left(\sum_{i=1}^{HW}(x_i' - Bu_i)u_i^T\right), \tag{7}$$

where $B \in \mathbb{R}^{D\times C}$ is a dictionary (with C codewords) and $u_i \in \mathbb{R}^{C\times 1}$ is the sparse code of x_i'.

This matrix $A \in \mathbb{R}^{D\times C}$ is neither square nor symmetric and thus, cannot be used as input for the Newton normalization that is restricted to SPD matrices. Indeed, since A is not SPD, its SVD is given as $A = U\Sigma V^T$, where $U \neq V$ and where $\Sigma \in \mathbb{R}^{D\times C}$ is not square.

In order to apply spectral normalization, we propose to estimate a so-called pseudo square root matrix $A^{1/2}_{pseudo}$ defined as $A^{1/2}_{pseudo} = U\Sigma^{1/2}_{pseudo}V^T$, where $\Sigma^{1/2}_{pseudo}$ is calculated by square rooting the diagonal elements of Σ. Note that there is no matrix $\Sigma^{1/2}$ such that $\Sigma = \Sigma^{1/2}\Sigma^{1/2}$.

Inspired by Lin and Maji (2017), to avoid SVD computation, we resort to the Newton method to evaluate such a $A^{1/2}_{pseudo}$ matrix. But, since this solution only accepts SPD square matrix as input, we transform A into a square SPD matrix D:

$$D = A^TA = V\Sigma^TU^TU\Sigma V^T = V\Sigma^T\Sigma V^T. \tag{8}$$

Note that this transform does not depend on U.

Since Σ is not symmetric, we introduce a helper matrix $H = [I_C|0]^T \in \mathbb{R}^{D\times C}$, with I_C the $C\times C$ identity matrix, such that Σ can be expressed as $\Sigma = H\tilde{\Sigma}$, where $\tilde{\Sigma}$ is a $C\times C$ square diagonal matrix.

Hence, Eq. (8) can be derived into:

$$D = V\Sigma^T\Sigma V^T = V\tilde{\Sigma}^TH^TH\tilde{\Sigma}V^T = V\tilde{\Sigma}^2V^T. \tag{9}$$

This equation is the SVD of the matrix D.

Feeding the previous Newton workflow with D and an identity matrix, we obtain $D^{1/2} = V\tilde{\Sigma}V^T$ and $D^{-1/2} = V\tilde{\Sigma}^{-1}V^T$ and feeding again this workflow with $D^{1/2}$ and an identity matrix, we obtain $D^{1/4} = V\tilde{\Sigma}^{1/2}V^T$ and $D^{-1/4} = V\tilde{\Sigma}^{-1/2}V^T$.

Finally, we have access to $A^{1/2}_{pseudo}$ thanks to:

$$\begin{aligned} AD^{-1/4} &= U\Sigma V^TV\tilde{\Sigma}^{-1/2}V^T = U\Sigma\tilde{\Sigma}^{-1/2}V^T = UH\tilde{\Sigma}\tilde{\Sigma}^{-1/2}V^T, \\ &= UH\tilde{\Sigma}^{1/2}V^T = U\Sigma^{1/2}_{pseudo}V^T = A^{1/2}_{pseudo}. \end{aligned} \tag{10}$$

Hence, without any SVD computation, this solution allows us to spectrally normalize a non-SPD matrix A as $A^{1/2}_{pseudo}$ very efficiently. Furthermore, this workflow can be easily embedded in an end-to-end trainable deep network.

Thus, we propose to apply this new spectral normalization to our Fisher score representation for classification tasks. All the frameworks can be trained end-to-end. In the next section, we propose to run extensive tests on different datasets to assess the quality of this method.

4. Experiments

In order to show that our solution generally helps the classification performance, we run experiments on three datasets, which vary between tasks and scales. The three datasets and their experimental settings are detailed in Sections 4.1 and 4.2. Next, the training strategy of our network is shown in Section 4.3. In Section 4.4, the results and comparisons will be discussed.

4.1. Datasets

Orderless pooling methods were originally designed for texture and material recognition tasks (Lin et al., 2017; Yu and Salzmann, 2018; Zhang et al., 2017). So we have first selected a reference material dataset. Then, these approaches have also been shown to provide good results on scene classification as well as fine-grained image classification (Liu et al., 2017; Lin et al., 2017; Li et al., 2017; Yu et al., 2021). Thus, we have also selected two dedicated datasets for these tasks. The choice of these three datasets (detailed hereafter) is also a good way to validate the versatility of our solution for different image classification tasks.

The dataset **MINC-2500** (Bell et al., 2015), containing 23 commonly-seen material categories and 2,500 images per category, is a challenging large-scale dataset with large intra-class variability. The dataset **MIT Indoor 67** (Quattoni and Torralba, 2009) is a medium but widely accepted benchmark for indoor scene classification task with 67 indoor categories and 100 images in each category. The dataset **CUB-200-2011** (Wah et al., 2011) provides 11,788 images of 200 bird species and is considered as a fine-grained classification dataset because the inter-class differences between bird species are subtle and sometime barely noticeable. In our experiments, we do not use the available object bounding boxes and part annotations. And we make use of official training-test splits released with these datasets.

4.2. Experimental settings

Deep Pooling Module (DPM) - Our DPM is composed of a 1×1 convolution layer, a LISTA module with two iterations (see Fig. 3), the Fisher encoding layer (see Section 3.1.3) and normalization process which includes matrix normalization(see Section 3.2.2), element-wise square root and l_2 normalization. Then, the DPM is followed by a fully connected layer with softmax activation for classification.

Depending on dataset scales and for a fair comparison with other works, we use different backbones and training strategies.

MIT-67 and CUB-200 settings - We adopt the settings of the state-of-the-art (Yu and Salzmann, 2018; Lin et al., 2017). The input image size is 448×448 and the backbone networks are the pre-trained VGG-D (a.k.a VGG-16), AlexNet and ResNet-50 (He et al., 2016). Our DPM is plugged after the ReLU activation of the last convolutional layer. The 1×1 convolutional layer in the DPM does not change the input feature size for AlexNet and VGG-D, and reduces the size to 512 for ResNet-50. The sparse code in LISTA has 100 elements.

MINC-2500 settings - The network backbone is the pre-trained ResNet-50 and VGG-D. With the 1×1 convolutional layer in the DPM, the input feature size is reduced to 128 and the size of sparse code in LISTA is 32. While training, we follow the data augmentation settings from Xue et al. (2018). First, the input image is resized to 256×256. Then we crop each image at (i) a random location with (ii) a random size (between 8% to 100% of the image area) and (iii) a random aspect ratio (between 3/4 and 4/3). The crop is resized to 224×224 and used as the network input. 50% chance horizontal and vertical flip is also applied. At test time, we use a central crop of 224×224 as input.

4.3. Training details

In the training phase, three consecutive steps are conducted. First, we run a PCA on a small subset of feature vectors (around 10,000) extracted from the backbone outputs and initialize the 1×1 convolutional layer of our DPM with these PCA parameters. Second, inspired by Branson et al. (2014), we apply a warm-up process that consists in training our DPM and FC layer (while the backbone is frozen) with an objective function which is the sum of the cross-entropy loss and the sparse coding loss (see Eq. (1)). Finally, the whole network is fine-tuned end-to-end under the supervision of the sole cross-entropy loss.

The optimization algorithm is a gradient descent with a mini-batch size of 64, a weight decay of $5e^{-4}$ and a momentum of 0.9. The learning rate is 0.004 during the warm-up. During the end-to-end finetuning, it starts from 0.004 and is divided by 10 when the training loss meets a plateau.

Table 1
Ablation study of our workflow on the MIT-67 dataset. Essential elements in our approach are progressively added and the accuracy(%) given by their different combination is measured, showing their individual contribution to the classification.

LISTA	Warm-up	Mean Sub.	Matrix Norm.	Accuracy
				76.72
✓				77.16
✓	✓			80.22
✓	✓	✓		80.60
	✓	✓	✓	80.67
✓	✓	✓	✓	**81.24**

Table 2
Inference time (ms) per mini-batch (64 samples) required by each element of our framework on a GPU Titan RTX. *Backb.*: the convolutional layers used to extract the deep features. *Norm.*: our matrix normalization step. *Fisher*: the Fisher encoding. *Classif.*: a linear classifier.

	Backb.	Mean Sub.	LISTA	Fisher	Norm.	Classif.
Time	1493	0.7	5.3	3.7	30.9	0.8

4.4. Results and discussions

In this section, we provide many results in order to assess the quality of each contribution, to measure the impact of the hyperparameters and to compare our whole framework with the state-of-the-art.

Ablation study - In order to measure the impact of each of our different contributions, we propose to conduct an ablation study. The tests are run on the MIT-67 dataset with the VGG-16 network and the results are provided in Table 1.

For this study, we propose to start from the baseline network without contributions and to consecutively add the proposed modules in order to assess their individual impact on the results. When the LISTA module is not in the network, it is replaced by a 1×1 convolutional layer providing the codes u_i.

As introduced in Section 4.3, our warm-up process is one of the three steps in the training phase. The goal is to train our DPM and the FC layer before fine-tuning the whole network. We can see in Table 1, that this training step boosts the performance from 77.16% to 80.22%, showing that an accurate initialization is important for our DPM and classifier.

Likewise, we notice that the proposed matrix normalization (called *Matrix Norm.* in Table 1. see details in Section 3.2.2) is one key element of our framework since it improves the accuracy from 80.60% to 81.24%.

Furthermore, centering the deep features (*Mean Sub.* in Table 1. see details in Section 3.1.4) also provides a slight improvement from 80.22% to 80.60%.

Finally, the impact of the LISTA module is measured with two different tests. Starting from the baseline and adding LISTA improves the results from 76.72% to 77.16%. And adding LISTA to the whole process helps to increase from 80.67% to 81.24%.

This ablation study is also a nice way to measure the improvement of our contributions over our previous work called E2E-SCF (Xu et al.,

Table 3
Comparison of the classification accuracy (%) with closed-related alternatives on three datasets and three backbone architectures.

	Approaches	MIT-67			CUB-200			MINC-2500	
		AlexNet	VGG16	ResNet50	AlexNet	VGG16	ResNet50	VGG16	ResNet50
Off-the-shelf	Baseline (Lin et al., 2017; Sharif Razavian et al., 2014)	58.4			53.3	60.4			
	GMMFVC (Liu et al., 2014, 2017)	64.3	72.6[a]		61.7	70.1[a]			
	SCFVC (Liu et al., 2014, 2017)	68.2	77.6[a]	83.28	66.4	77.3[a]	78.86		
	HSCFVC (Liu et al., 2017)		79.5[a]			80.8[a]			
Finetuned	Baseline (Lin et al., 2017; Yu and Salzmann, 2018)		64.51	76.45		70.4	74.51	73.01	79.12
	Deep Ten (Zhang et al., 2017)			71.3					80.6
	NetVLAD (Lin et al., 2017)					81.9			
	NetFV (Lin et al., 2017)		78.2			79.9			
	FisherNet (Tang et al., 2019)		76.4						
	MFAFVNet (Li et al., 2017)	69.89[b]	78.01[b]						
	CNN-DL (Liu et al., 2018)	66.60	78.33						
	B-CNN (Lin et al., 2017; Yu and Salzmann, 2018)		77.6			84.0		74.50	79.05
	SMSO (Yu and Salzmann, 2018)		79.45	79.68		85.01	85.77	**78.0**	81.33
	SRM (Yu et al., 2021)		80.3			85.5			
	RUN (Yu et al., 2020)		80.8			85.7			
	E2E-SCF (Xu et al., 2021)	70.15	80.22	84.85	76.8	84.28	84.47	76.56	81.5
	Ours	**70.60**	**81.24**	**85.52**	**77.49**	**85.8**	**87.38**	76.60	**81.8**

[a]Trained with VGG19 (not VGG16) with 2 scales, while the other approaches from the column are trained with a single scale.
[b]Since MFAFVNet works on patches and not on images, we have selected in Li et al. (2017) the results provided with the nearest patch scale from our settings (160 × 160).

Table 4
Impact of the number of iterations in LISTA on the accuracy (%).

Iter. number	0	1	2	3	4	5
Accuracy	80.67	81.04	81.24	81.34	81.04	80.30

Table 5
Impact of the dictionary size on the accuracy (%).

Dataset	Dictionary size					
	50	100	200	300	400	512
MIT-67	80.37	81.24	80.90	80.00	80.22	80.22
MINC-2500	76.00	76.37	76.45	76.89	77.08	76.90

2021). Indeed, in Table 1, the row with LISTA and Warm-up corresponds to our E2E-SCF. We notice that the additive contributions help to improve the accuracy from 80.22% to 81.24% on this dataset. Additional comparisons with this previous paper are proposed in Table 3 with 3 architectures and 3 datasets.

After analyzing the contribution of each element, we propose to discuss their individual computational costs. Thus, we have measured their inference times on the MIT-67 dataset with the VGG-16 backbone. According to Table 2, the feature extraction with the convolutional backbone (VGG-16, here) is clearly the bottleneck of the framework. The inference times of our proposed blocks are negligible compared to the one of the backbone. Among our proposed modules, the matrix normalization, i.e. the Newton algorithm, has the highest computational cost.

Hyperparameters - In order to go a step further in the analysis of our framework, we propose to study the impacts of two hyper-parameters on the results; namely the number of iterations in the LISTA module and the size of the dictionary in the sparse coding. Like the previous experiment, the tests are conducted on the MIT-67 dataset with the VGG-16 network.

LISTA is an unfolded version of ISTA and the number of iterations is a hyper-parameter. We investigate the performance of our framework across different numbers of iterations from 0 to 5, where 0 means that the LISTA module is replaced by a 1×1 convolutional layer. In Table 4, we notice that 2 or 3 iterations provide the best performance. After 3 iterations, the results start decreasing. Our intuition is that too many iterations of LISTA produce sparser codes at the expense of classification accuracy. For all the other tests in this paper, 2 iterations are used.

We also conducted an analysis on the number of codewords (dictionary size) required in the LISTA module. We measure the classification accuracy for a range of codeword numbers from 50 to 512 in Table 5. We notice that, due to overfitting, when the number of codewords is higher than 100, lower accuracy is observed. It appears that for larger datasets, like MINC-2500, the optimal dictionary size is larger. The default value for all the next tests is 100, unless it is specified.

Comparison with state-of-the-art - The top-1 classification accuracy of our approach and many alternatives are provided in Table 3. The results of the related works are directly extracted from the reference papers cited in the Table. Note that our CNN is trained on single-scale images while many state-of-the-art approaches are trained on multi-scales, so we have carefully selected the results that allow fair comparisons, but still some results in Table 3 are from multi-scale training (see comments in Table 3).

The methods called *Off-the-shelf* use independent modules that are not fine-tuned together while the *Finetuned* group contains approaches that use end-to-end trainable networks. We notice that the results provided by fine-tuned networks overall outperform those of the Off-the-shelf solutions. This shows that it is better to make the modules work together to optimize the same loss instead of independently optimizing them. We can also notice that the second order statistics, such as Fisher vectors (MFAFVNet, HSCFVC, ...) or bilinear pooling (SMSO, SRM, RUN, ...) provide the best results for the tested datasets. Our approach is built upon Deep Fisher Score Representation via Sparse Coding (SCFVC Liu et al., 2014) and our E2E-SCF (Xu et al., 2021) which produce more discriminative second-order pooled features than the classical Fisher vector. We can see in Table 3 that the proposed smart combination of these two advantages makes our method outperform the alternatives for almost all the tested datasets and backbones. And the improvement provided by our approach increases with feature dimension and the depth of the network (AlexNet ⟶ VGG16 ⟶ ResNet50) which shows that it is well designed for complex and high dimensional features.

5. Conclusion

In this paper, we have proposed a workflow to extract second-order statistics from images in the context of image classification. The approach is based on Fisher encoding which requires a data distribution

fitting with Gaussians. We have first proposed to sparsely encode the Gaussian centers with a learned basis in order to improve the data fitting. Second, since the second-order features require a spectral normalization before being used for classification, we have introduced an original matrix normalization based on a Newton algorithm. The main advantage of these two modules is that they can be embedded in a deep network that can be trained end-to-end. We have also proposed a training strategy that can easily initialize the network parameters before finetuning. Many experimental tests clearly show that our method outperforms the recent alternatives on three different datasets. The proposed non-SPD matrix normalization can be exploited to improve other second order statistics features such as those provided by compact bilinear pooling (Gao et al., 2015). This is the aim of our future works.

CRediT authorship contribution statement

Sixiang Xu: Conceptualization, Methodology, Software, Writing – original draft, Writing – review & editing. **Damien Muselet:** Methodology, Writing – original draft, Writing – review & editing. **Alain Trémeau:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Arandjelović, R., Gronat, P., Torii, A., Pajdla, T., Sivic, J., 2016. NetVLAD: CNN architecture for weakly supervised place recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Arandjelovic, R., Zisserman, A., 2013. All about VLAD. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1578–1585.

Bell, S., Upchurch, P., Snavely, N., Bala, K., 2015. Material recognition in the wild with the materials in context database. In: Computer Vision and Pattern Recognition (CVPR).

Branson, S., Van Horn, G., Belongie, S., Perona, P., 2014. Bird species categorization using pose normalized deep convolutional nets. arXiv preprint arXiv:1406.2952.

Carreira, J., Caseiro, R., Batista, J., Sminchisescu, C., 2012. Semantic segmentation with second-order pooling. In: European Conference on Computer Vision. Springer, pp. 430–443.

Cimpoi, M., Maji, S., Vedaldi, A., 2015. Deep filter banks for texture recognition and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3828–3836.

Daubechies, I., Defrise, M., Mol, C., 2004. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. Comm. Pure Appl. Math. 57, 1413–1457.

Gao, B.-B., Wei, X.-S., Wu, J., Lin, W., 2015. Deep spatial pyramid: The devil is once again in the details. arXiv preprint arXiv:1504.05277.

Gregor, K., LeCun, Y., 2010. Learning fast approximations of sparse coding. In: Proc. International Conference on Machine Learning (ICML'10).

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Hu, Y., Long, Z., AlRegib, G., 2019. Multi-level texture encoding and representation (multer) based on deep neural networks. In: 2019 IEEE International Conference on Image Processing (ICIP). IEEE, pp. 4410–4414.

Huang, Z., Van Gool, L., 2017. A riemannian network for spd matrix learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 31.

Ionescu, C., Vantzos, O., Sminchisescu, C., 2015. Matrix backpropagation for deep networks with structured layers. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2965–2973.

Jacob, P., Picard, D., Histace, A., Klein, E., 2019. Efficient codebook and factorization for second order representation learning. In: Proc. International Conference on Learning Representations (ICLR).

Jégou, H., Perronnin, F., Douze, M., Sánchez, J., Pérez, P., Schmid, C., 2012. Aggregating local image descriptors into compact codes. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) 34 (9).

Krestenitis, M., Passalis, N., Iosifidis, A., Gabbouj, M., Tefas, A., 2020. Recurrent bag-of-features for visual information analysis. Pattern Recognit. 106, 107380.

Laakom, F., Passalis, N., Raitoharju, J., Nikkanen, J., Tefas, A., Iosifidis, A., Gabbouj, M., 2020. Bag of color features for color constancy. IEEE Trans. Image Process. 29, 7722–7734.

Lazebnik, S., Schmid, C., Ponce, J., 2006. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), Vol. 2. pp. 2169–2178.

Li, Y., Dixit, M., Vasconcelos, N., 2017. Deep scene image classification with the MFAFVNet. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 5746–5754.

Lin, T.-Y., Maji, S., 2017. Improved bilinear pooling with CNNs. In: Kim, T.-K., Zafeiriou, S., Brostow, G., Mikolajczyk, K. (Eds.), Proceedings of the British Machine Vision Conference (BMVC). BMVA Press, pp. 117.1–117.12. http://dx.doi.org/10.5244/C.31.117.

Lin, T.-Y., RoyChowdhury, A., Maji, S., 2015. Bilinear cnn models for fine-grained visual recognition. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1449–1457.

Lin, T.-Y., RoyChowdhury, A., Maji, S., 2017. Bilinear convolutional neural networks for fine-grained visual recognition. IEEE Trans. Pattern Anal. Mach. Intell. 40 (6), 1309–1322.

Liu, Y., Chen, Q., Chen, W., Wassell, I., 2018. Dictionary learning inspired deep network for scene recognition. In: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32.

Liu, L., Shen, C., Wang, L., Hengel, A.v.d., Wang, C., 2014. Encoding high dimensional local features by sparse coding based fisher vectors. In: Advances in Neural Information Processing Systems(NIPS).

Liu, L., Wang, P., Shen, C., Wang, L., Van Den Hengel, A., Wang, C., Shen, H.T., 2017. Compositional model based fisher vector coding for image classification. IEEE Trans. Pattern Anal. Mach. Intell. 39 (12), 2335–2348.

Mao, S., Rajan, D., Chia, L.T., 2021. Deep residual pooling network for texture recognition. Pattern Recognit. 112, 107817.

Passalis, N., Tefas, A., 2017. Learning bag-of-features pooling for deep convolutional neural networks. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 5766–5774.

Perronnin, F., Sánchez, J., Mensink, T., 2010. Improving the fisher kernel for large-scale image classification. In: European Conference on Computer Vision. Springer, pp. 143–156.

Quattoni, A., Torralba, A., 2009. Recognizing indoor scenes. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, pp. 413–420.

Sánchez, J., Mensink, T., Verbeek, J., 2013. Image classification with the fisher vector: Theory and practice. Int. J. Comput. Vis. 105.

Sharif Razavian, A., Azizpour, H., Sullivan, J., Carlsson, S., 2014. CNN features off-the-shelf: An astounding baseline for recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops.

Simonyan, K., Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition. In: Proc. International Conference on Learning Representations (ICLR'15).

Sun, X., Nasrabadi, N.M., Tran, T.D., 2019. Supervised deep sparse coding networks for image classification. IEEE Trans. Image Process. 29, 405–418.

Tang, P., Wang, X., Shi, B., Bai, X., Liu, W., Tu, Z., 2019. Deep fishernet for image classification. IEEE Trans. Neural Netw. Learn. Syst. 30 (7), 2244–2250.

Ulyanov, D., Vedaldi, A., Lempitsky, V., 2016. Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022.

Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S., 2011. The Caltech-UCSD Birds-200–2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology.

Wang, L., Koniusz, P., 2021. Self-supervising action recognition by statistical moment and subspace descriptors. In: Proceedings of the 29th ACM International Conference on Multimedia. pp. 4324–4333.

Wang, L., Koniusz, P., Huynh, D.Q., 2019. Hallucinating idt descriptors and i3d optical flow features for action recognition with cnns. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 8698–8708.

Xu, S., Muselet, D., Trémeau, A., 2021. Deep fisher score representation via sparse coding. In: Proceedings of the 19th International Conference on Computer Analysis of Images and Patterns (CAIP). In: Springer Verlag's Series Lecture Notes in Computer Science.

Xue, J., Zhang, H., Dana, K., 2018. Deep texture manifold for ground terrain recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 558–567.

Yang, J., Liu, W., Yuan, J., Mei, T., 2020. Hierarchical soft quantization for skeleton-based human action recognition. IEEE Trans. Multimed. 23, 883–898.

Yu, T., Cai, Y., Li, P., 2020. Toward faster and simpler matrix normalization via rank-1 update. In: European Conference on Computer Vision. Springer, pp. 203–219.

Yu, T., Li, X., Li, P., 2021. Fast and compact bilinear pooling by shifted random maclaurin. In: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35. pp. 3243–3251.

Yu, K., Salzmann, M., 2018. Statistically-motivated second-order pooling. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 600–616.

Zhang, H., Xue, J., Dana, K., 2017. Deep ten: Texture encoding network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 708–717.