This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the version available on IEEE Xplore.

Deep Scene Image Classification with the MFAFVNet

Yunsheng Li Mandar Dixit Nuno Vasconcelos
University of California, San Diego
La Jolla, CA 92093
yul554@ucsd.edu mdixit@ucsd.edu nvasconcelos@ucsd.edu

Abstract

The problem of transferring a deep convolutional network trained for object recognition to the task of scene image classification is considered. An embedded implementation of the recently proposed mixture of factor analyzers Fisher vector (MFA-FV) is proposed. This enables the design of a network architecture, the MFAFVNet, that can be trained in an end to end manner. The new architecture involves the design of a MFA-FV layer that implements a statistically correct version of the MFA-FV, through a combination of network computations and regularization. When compared to previous neural implementations of Fisher vectors, the MFAFVNet relies on a more powerful statistical model and a more accurate implementation. When compared to previous non-embedded models, the MFAFVNet relies on a state of the art model, which is now embedded into a CNN. This enables end to end training. Experiments show that the MFAFVNet has state of the art performance on scene classification.

1. Introduction

Scene image classification is an important problems for applications of computer vision such as robotics, image search, geo-localization, etc. It is also a challenging problem, e.g. object recognition methods do not necessarily work for scenes. This is because scenes include both a holistic component, the *gist* of the scene, and an object-based component. Furthermore, the object vocabulary is usually open-ended and it does not suffice to recognize objects, as most scenes are collections of objects in characteristic spatial layouts. There is also a need to model relationships between objects. Historically, this motivated different approaches to scene classification, including holistic gist descriptors [20] and descriptors based on local features.

Localized approaches included descriptors of contextual relationships between objects [14] or semantics properties such as the objects in the scene [15] or scene attributes [25]. Many of these approaches were based on the bag-of-features (BoF) representation, using local features such as SIFT or HOG [19, 3], combined through a pooling operator. Pooling has a critical role in scene classification, for two reasons. First, there is a need to integrate object information across the scene. Second, this integration must be invariant to the locations of individual objects, which can change drastically within the same scene class. Eventually, sophisticated pooling strategie, such as the vector of locally aggregated descriptors (VLAD) [12] or the Fisher vector (FV) [23] emerged as the dominant pooling mechanisms for scene classification.

In recent years, CNNs have become the feature extractors of choice for scene classification. In fact, the implementation of a holistic scene classifier with a CNN is almost trivial. It suffices to train the CNN on whole scene images. The main challenges are the assembly of a large dataset of such images and the standard difficulties of training a deep network. These problems have been addressed in [29], through the assembly of the Places dataset and its use to train a deep scene classifier. The use of deep *object-based* representations for scene classification has, however, proven more challenging. This is, in great part, because object-based scene classification is a difficult transfer learning problem. While the ease with which CNNs can be transferred across datasets, by simple finetuning, is one of their greatest assets for vision, this procedure has its limitations. The transfer from a localized representation (needed to recognize objects) to a holistic task (classification of whole scenes) cannot, in general, be solved by simple finetuning.

The previous success of pooling on this type of transfer created a resurgence of interest in the topic within the deep learning realm. Several authors proposed Fisher vectors or similar pooling mechanisms for features extracted by object recognition CNNs. Early methods adopted a BoF-like approach, based on the extraction of features from intermediate CNN layers, which were then fed to dictionary learning methods such as clustering [11] or sparse coding [17] and finally used to implement descriptors such as the VLAD [11] or Fisher vector [17]. Later, [5] proposed the semantic Fisher Vector, which extracts features from the softmax layer at the top of the CNN, converting features from probability space to the natural parameter space.

All these methods suffer from two drawbacks. First, the Fisher vector structure is not easy to integrate in a CNN. This is because the Fisher vector is defined with respect to the probability distribution of the CNN features, usually estimated with a mixture learned by maximum likelihood. The Fisher vector is then a complex expression of the mixture parameters, which changes when these change. In result, the Fisher vector cannot be learned by simply backpropagating the output of the scene classifier. All methods above avoid this difficulty by using the CNN to extract features and learning the Fisher vector indepeddently. This, however, prevents end to end training and, consequently, the finetuning of the object network to the scene classification task. In result, the transfer between the two tasks relies solely on the Fisher vector, which is sub-optimal.

The second problem is that the Fisher vector is usually learned with respect to a Gaussian mixture model (GMM). Since CNN features are high dimensional, it is impractical to rely on Gaussians of full covariance. Instead, the mixture components are chosen to have diagonal covariance. This creates problems when the feature manifold is nonlinear. In this case, a large number of diagonal GMM components are required and the Fisher vector is very high dimensional. This is indeed the norm for computer vision applications, where Fisher vectors usually have several thousand dimensions. While the CNN is trained to produce linearly separable responses to the different classes, there is no guarantee CNN feature distributions are prone to modeling with the diagonal GMM. On the contrary, given the highly non-linear feature transformation implemented by a deep CNN, this is unlikely. Hence, a Fisher vector based on the diagonal-GMM is likely to be very high dimensional and potentially sub-optimal for scene classification.

Recently, some works have attempted to solve these problems. One possibility is to bypass the Fisher vector altogether. For example, [7] proposed a compact bilinear pooling (CBP) mechanism that enables end-to-end training by simple backpropagation. While this was shown applicable to scene classification, the performance of CBP is inferior to those of previous approaches, such as the semantic Fisher vector of [5] or the sparse coding methods of [18], for equivalent object CNNs. Another possibility is to embed the Fisher vector in the CNN architecture, by deriving a neural network implementation of its equations. [1] proposed the NetVLAD, an embedded implementation of the VLAD descriptor, and [26] proposed the Deep FisherNet, an embedded implementation of the GMM Fisher vector. However, to avoid the difficulties of the complete Fisher vector, these methods make approximations, such as disregarding covariance structure (VLAD) or using crude approximations of posterior mixture probabilities (Deep FisherNet). These ap-

proximations can be quite sub-optimal.

An alternative strategy, proposed by [6], is to use a better model of CNN feature statistics than the diagonal GMM. Instead, this work proposed a Fisher vector based on the mixture of factor analyzers (MFA). This has the advantage of accounting for the covariance information of each mixture component, which is modeled through factor analysis. Under the MFA model, good results can be achieved with mixtures of few components and Fisher vectors of reduced dimension. However, while the MFA-Fisher vector (MFA-FV) currently holds state of art results for scene classification, it is not an integrated model, i.e. it is learned independently of the network. This raises all the reservations discussed above.

In this work, we derive an embedded implementation of the MFA-FV, and use it to design a network architecture, the MFAFVNet, which can be trained in an end to end manner. This involves the derivation of a MFA-FV layer that implements a statistically correct version of the MFA-FV, through a combination of network computations and regularization. The computations replicate those of the MFA-FV, regularization guarantees that all parameters have a statistically valid interpretation. When compared to previous embedded implementations, the MFAFVNet relies on a more powerful statistical model, which accounts for covariance information, and a more accurate implementation. This results in significant performance gains for scene classification. When compared to previous non-embedded models, the MFAFVNet relies on a state of the art model, which is embedded. This enables end to end training and better scene classification performance. Extensive experiments on the MIT Indoor and SUN datasets show that the MFAFVNet achieves state of the art performance for scene classification.

2. The MFA Fisher Vector

In this section, we review the main ideas behind the MFA-FV.

2.1. Mixture of Factor Analyzers

The factor analysis (FA) model is a probabilistic extension of principal component analysis (PCA) [23]. Given a vector $x \in \mathbb{R}^D$ of D observations, it explains its covariance structure by assuming that the variability of the observations can be explained by a small number $d < D$ of hidden or latent *factors,* usually represented as a factor vector $z \in \mathbb{R}^d$. Observations x are assumed to be sampled according to the model

$$x - \mu = \Lambda z + \epsilon, \tag{1}$$

where μ is the mean value of x, $\Lambda \in \mathbb{R}^{D \times d}$ is a *factor loading matrix*, and ϵ is a noise term. Factors z and noise ϵ are independent of each other and Gaussian, and the noise variables are assumed uncorrelated, i.e. distributed as $\mathcal{N}(\epsilon; 0, I)$. The factors can be dependent, i.e. they are distributed as $\mathcal{N}(z; 0, \psi)$, but the matrix ψ is sometimes assumed diagonal. It follows from the linearity of (1) that x is Gaussian with covariance

$$\Sigma = cov(x - \mu) = cov(\Lambda z + \epsilon) = \Lambda\Lambda^T + \psi. \tag{2}$$

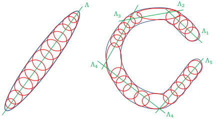

Figure 1. Probability distributions defined on linear susbpaces (left) or non-linear manfolds (right) can require many mixture components to approximate, when covariances are diagonal (shown in red). However, they can frequently be approximated by a few MFA components (in green).

Hence, the factor loading matrix Λ has a role similar to the principal component matrix of PCA.

The mixture of factor analyzers (MFA) is a mixture model whose components follow the factor analysis model. A MFA of C components is defined by the distributions

$$p(c) = \pi_c \tag{3}$$

$$p(z|c) = \mathcal{N}(z; 0, I) \tag{4}$$

$$p(x|z, c) = \mathcal{N}(x; \Lambda_c z + \mu_c, \Psi_c) \tag{5}$$

where $p(c)$ is the probability of component c and this component is a FA of mean μ_c, factor loading matrix Λ_c and noise covariance Ψ_c.

The MFA can be learned with a EM algorithm, which is discussed in [8]. This iterates between an expectation step that computes expected values for the hidden class c and factors z variables, and a maximization step that updates these model parameters so as to maximize the likelihoods of a set of observations $\{x_i\}_{i=1}^N$.

2.2. Fisher vectors

Given a dataset $\mathcal{D} = \{x_i\}$ and a probability model $p(x; \theta)$ the score $\mathcal{G}(\theta) = \frac{\partial}{\partial \theta} \log p(\mathcal{D}; \theta)$ measures the sensitivity of the likelihood $p(\mathcal{D}; \theta)$ to parameter θ. The normalization of this gradient vector by the square root of the Fisher information matrix $\mathcal{I}(\theta) = -\sum_i \frac{\partial^2}{\partial \theta^2} \log p(x_i; \theta)$, i.e. the vector $\mathcal{I}^{-1/2}\mathcal{G}(\theta)$ is usually denoted as the *Fisher vector* [23]. However, because the Fisher information can be difficult to compute, it is frequently ignored and the Fisher vector reduces to the score $\mathcal{G}(\theta)$. As is common in computer vision, we adopt this practice in this work.

The Fisher vector is commonly used with the standard Gaussian mixture model, defined by

$$p(c) = \pi_c \tag{6}$$

$$p(x|c) = \mathcal{N}(x; \mu_c, \Sigma_c). \tag{7}$$

However, because vision data tends to be high-dimensional, it is usually difficult to use full covariance Gaussians in this model, and the covariances Σ_c are assumed diagonal. This removes a lot of the expressiveness of the mixture model. In general, many components are needed to achieve a good approximation of the distribution $p(x)$. As illustrated in Figure 1, this is particularly true when the data lives on correlated low-dimensional subspaces or non-linear manifolds that can be approximated by a set of low-dimensional subspaces. The MFA is a substantially better model for this type of data since, in this case, only a few mixture components and a small number d of hidden factors are required to estimate the covariance structure of (2). This is illustrated in Figure 1 as well. This observation, motivated the introduction of the MFA-Fisher vector (MFA-FV) in [6], which was shown to have the form

$$\begin{aligned}
\mathcal{G}_{\mu_c}(\mathcal{I}) &= \sum_i p(c|x_i;\theta)\psi^{-1}(I - \Lambda_c\Gamma_c)(x_i - \mu_c) && (8)\\
\mathcal{G}_{\Lambda_c}(\mathcal{I}) &= \sum_i p(c|x_i;\theta)\psi^{-1}(\Lambda_c\Gamma_c - I)\\
&\quad [(x_i - \mu_c)(x_i - \mu_c)^T\Gamma_c^T - \Lambda_c] && (9)\\
\Gamma_c &= \Lambda_c^T S_c^{-1}. && (10)
\end{aligned}$$

This work has also shown that the MFA-FV is a substantially better representation than the classical FV when the observations x are feature vectors produced by a deep convolutional neural network (CNN). As far as we know, this is the state of the art representation for scene classification. However, [6] did not integrate the MFA-FV in the network computation. This prevents end-to-end training and the tuning of the network to the scene classification task. Since end to end training is an important reason for the recent success of the deep CNN architecture, it appears natural to pursue this integration.

3. Network implementation of the MFA-FV

In this section, we derive a neural network implementation of the MFA-FV.

3.1. The MFA-FV layer

To derive a version of (8)-(10) implementable as a neural network, we start by defining

$$\begin{aligned}
\Delta_{ic} &= x_i - \mu_c && (11)\\
S_c &= \Lambda_c\Lambda_c^T + \psi_c. && (12)
\end{aligned}$$

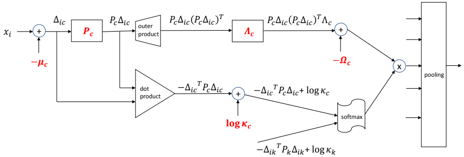
x_i
Δ_{ic}
$-\mu_c$
P_c
$P_c\Delta_{ic}$
outer product
$P_c\Delta_{ic}(P_c\Delta_{ic})^T$
Λ_c
$P_c\Delta_{ic}(P_c\Delta_{ic})^T\Lambda_c$
$-\Omega_c$
x
pooling
dot product
$-\Delta_{ic}{}^T P_c\Delta_{ic}$
$-\Delta_{ic}{}^T P_c\Delta_{ic}+\log\kappa_c$
$\log\kappa_c$
softmax
$-\Delta_{ik}{}^T P_k\Delta_{ik}+\log\kappa_k$

Figure 2. The MFA-FV layer. The expressions in red are the parameters of the cth MFA component. The remaining expressions show what is computed at each stage of the network.

Combining this with (10),

$$\begin{aligned}\psi_c^{-1}(I-\Lambda_c\Gamma_c) &= (S_c-\Lambda_c\Lambda_c^T)^{-1}(I-\Lambda_c\Lambda_c^T S_c^{-1})\\ &= S_c^{-1}(I-\Lambda_c\Lambda_c^T S_c^{-1})^{-1}(I-\Lambda_c\Lambda_c^T S_c^{-1})\\ &= S_c^{-1},\end{aligned}$$

and it follows that (8)-(9) can be written as

$$\mathcal{G}_{\mu_c}(\mathcal{I}) = \sum_i p(c|x_i;\theta)S_c^{-1}\Delta_{ic} \tag{13}$$

$$\begin{aligned}\mathcal{G}_{\Lambda_c}(\mathcal{I}) &= -\sum_i p(c|x_i;\theta)S_c^{-1}[\Delta_{ic}\Delta_{ic}^T S_c^{-1}\Lambda_c-\Lambda_c]\\ &= -\sum_i p(c|x_i;\theta)S_c^{-1}\Delta_{ic}[S_c^{-1}\Delta_{ic}]^T\Lambda_c\\ &\quad +\sum_i p(c|x_i;\theta)S_c^{-1}\Lambda_c\end{aligned} \tag{14}$$

Furthermore, since (2) implies that the cth mixture component $p(x|c)$ is distributed as $\mathcal{N}(x,\mu_c,S_c)$, it follows that

$$\begin{aligned}p(c|x_i;\theta) &= \frac{\pi_c\mathcal{N}(x_i;\mu_c,S_c)}{\sum_k \pi_k\mathcal{N}(x_i;\mu_k,S_k)} \\ &= \frac{\frac{\pi_c}{|S_c|^{\frac{1}{2}}}\exp\{-\frac{1}{2}\Delta_{ic}^T S_c^{-1}\Delta_{ic}\}}{\sum_k \frac{\pi_k}{|S_k|^{\frac{1}{2}}}\exp\{-\frac{1}{2}\Delta_{ic}^T S_k^{-1}\Delta_{ic}\}}\end{aligned} \tag{15}$$

Denoting

$$P_c = S_c^{-1}, \tag{16}$$

$$\Omega_c = S_c^{-1}\Lambda_c, \tag{17}$$

$$\kappa_c = \frac{\pi_c}{|S_c|^{\frac{1}{2}}}, \tag{18}$$

finally leads to

$$\mathcal{G}_{\mu_c}(\mathcal{I}) = \sum_i p(c|x_i;\theta)P_c\Delta_{ic} \tag{19}$$

$$\mathcal{G}_{\Lambda_c}(\mathcal{I}) = -\sum_i p(c|x_i;\theta)\{P_c\Delta_{ic}(P_c\Delta_{ic})^T\Lambda_c-\Omega_c\} \tag{20}$$

$$p(c|x_i;\theta) = \frac{\kappa_c\exp\{-\frac{1}{2}\Delta_{ic}^T P_c\Delta_{ic}\}}{\sum_k \kappa_k\exp\{-\frac{1}{2}\Delta_{ik}^T P_k\Delta_{ik}\}} \tag{21}$$

An implementation of (20) as a neural network layer is shown in Figure 2. The bottom branch computes the posterior probability of (21). The top branch computes the remainder of the argument of the summation in (20). The computations of (19) are similar. The bottom branch is identical, and the top branch omits the operations beyond $P_c\Delta_{ic}$. However, preliminary experiments showed no gains for the addition of this component. Hence, we use only the second order information, i.e. (20). Note that the operations inside circle are applied entry-wise, the boxes implement matrix multiplications implementable with standard layers of weights, the outer product layer is similar to that of [16], and the dot-product layer can be implemented with an elementwise multiplication and a sum.

3.2. Relation to other Fisher vectors

The MFA-FV is related to various previous representations of the same type. For example, if Λ_c is the identity matrix and S_c a diagonal matrix of elements σ_{ck}^2, then (19) reduces to

$$\mathcal{G}_{\mu_{ck}}(\mathcal{I}) = \sum_i p(c|x_{ik};\theta)\frac{(x_{ik}-\mu_{ck})}{\sigma_{ck}^2} \tag{22}$$

and (20) to

$$\mathcal{G}_{\sigma_{ck}}(\mathcal{I}) = \sum_i p(c|x_i;\theta)\frac{1}{\sigma_{ck}^2}\left\{\frac{(x_{ik}-\mu_{ck})^2}{\sigma_{ck}^2}-1\right\}, \tag{23}$$

which are similar to the Fisher Score of the standard Gaussian mixture model [23, 21]. Further omitting the second other information leads to

$$\mathcal{G}_{\mu_{ck}}(\mathcal{I}) = \sum_i p(c|x_i;\theta)(x_{ik}-\mu_{ck}) \tag{24}$$

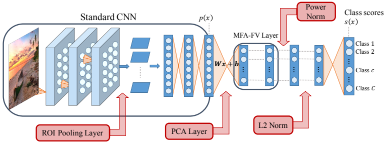

Figure 3. Architecture of MFAFVNet.

and replacing the posterior probabilities $p(c|x_i;\theta)$ by binary variables a_{ic} such that $a_{ic}=1$ if μ_c is the closest mean to x_i and $a_{ic}=0$ reduces to the VLAD

$$V_{ck} = \sum_i a_{ic}(x_{ik}-\mu_{ck}). \tag{25}$$

Note that the variables a_{ic} can be obtained by replacing the softmax by a max in Figure 2.

3.3. The MFAFVnet

The overall architecture of the MFAFVNet is shown in Figure 3. A model pretrained on ImageNet is first used to extract a vector $p(x)$ of image features. Our implementation uses either Alexnet or VGG, both of which include a sequence of several convolutional layers and three fully connected layers. As is common for scene classification, this network is applied to image patches [23], producing multiple feature maps per image to classify. When these patches are of a single scale, the model is converted to a fully convolutional network. When patches of multiple scales are used, the final pooling layer is replaced with a region-of-interest (ROI) pooling layer, which accepts feature maps of multiple sizes and produces a fixed size output to the fully connected layers. This is similar to the standard practice in the object detection literature [10, 9]. As is usual in the Fisher vector literature [23] the feature vector $p(x)$ is subject to dimensionality reduction. This is implemented by a fully connected layer of appropriate dimensions and creates the input to the MFA-FV layer. This is implemented as shown in Figure 2. Note that the MFA-FV layer pools multiple local features, corresponding to objects of different sizes and in different locations of the input image. It produces a single feature vector that represents the whole input image. As is standard in the Fisher vector literature [21], two normalization layers (power normalization and L2 normalization) are finally included in the network before the final linear classifier.

3.4. Loss Function

The parameters $\mu_c, P_c, \Lambda_c, \Omega_c$, and $\log k_c$ of the ne
of Figure 2 have an interpretation as statistical quar
which follows from the derivation of Section 3.1. To
tain this interpretation, they have to satisfy certain rel
ships, namely (12), (16), (17), and (18). Some of the
lationships do not necessarily need to be enforced. F
ample, since (18) is the only to involve π_c, there is
to one relationship between the values of $\log\kappa_c$ and
dependently of the value of $|S_c|^{1/2}$. Hence, it is equi
to learn π_c under the constraint (18) or simply learn l
Since the latter leads to a simpler optimization, it sho
favored. A similar observation can be made for (12),
is the only equation to constrain ψ_c.

On the other hand, some of the relationships m
enforced to maintain the MFA-FV interpretation.
are (16), (17), and the fact that P_c is a symmetric r
These relationships can be enforced by adding regu
tion terms to the loss function used to train the ne
Given a training set $\mathcal{D}=\{(x_i,y_i)\}$ this leads to a loss
tion

$$\begin{aligned}L(\mathcal{D}) &= L_c(\mathcal{D})+\lambda_1\sum_c ||\Omega_c-P_c\Lambda_c||_F^2\\ &+ \lambda_2\sum_c ||P_c-P_c^T||_F^2\end{aligned}$$

where $||A||_F$ is the Frobenius norm of A, the last two
are the regularizers that enforce the constraints and L
a standard classification loss. In our implementation
the hinge loss

$$L_c(\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C}[\max(0,1-\delta(y_i=k)s_k(x_i$$

where $s(x)$ is the input to the softmax at the top
network, $\delta(\cdot)$ the indicator function (1 when the arg
holds and zero otherwise), and λ_1, λ_2 two parameter

twork
ntities,
main-
lation-
ese re-
For ex-
a one
π_c, in-
ivalent
$\log \kappa_c$.
uld be
which

ust be
These
natrix.
lariza-
twork.
s func-

(26)

terms
$_c(.)$ is
this is

$)]^p$

(27)
of the
ument
s used

Table 1. Effect of initialization on MFAFVNet classification accuracy.

Initialization	MIT Indoor	SUN
AlexNet		
Random	69.82	50.23
Pre-Trained MFA	71.44	54.14
VGG-16		
Random	77.3	56.2
Pre-Trained MFA	80.3	62.51

to control the strength of the regularization. In our implementation we use $p = 2$. The choice of the hinge loss is mostly for consistency with the Fisher vector literature, which is mostly based on SVMs. Any other classification loss could in principle be used.

4. Experiments

In this section, we report on an extensive experimental evaluation of the MFAFVNet.

4.1. Experimental Setup

Datasets: All experiments were based on the 67 class MIT Indoor scenes dataset [22] and the 397 class MIT SUN dataset [28]. MIT Indoor includes 80 images of each category for training and 20 images for testing. SUN includes multiple train/test splits, with 50 images per class in the testing set. We present results for the average accuracy over splits.

Baselines: The MFAFVNet was compared to eight previous methods for scene classification. The VLAD of [11], the Sparse and H-Sparse coding of [17, 18], the Semantic Fisher Vector (SFV) of [5], the full (FBN) and compact (compact BN) bilinear pooling networks of [7], the MFA-FS of [6], the Deep FisherNet of [26] and the MetaClass method of [27]. While most of these methods [11, 5, 6, 26, 27] present results for both MIT Indoor and SUN, some [7, 18] only report in MIT Indoor. With the exception of the Deep FisherNet of [26], all these results are obtained by simply using features extracted from CNN layers, without any finetuning of the network. We simple restate their result. [26] did not address scene classification, only presenting results for object detection on PASCAL VOC 2012. We implemented the network as described in [26] and present its results on MIT Indoor and SUN.

Implementation Details: The MFAFVNet was implemented with three different object recognition networks trained on ImageNet [4]: Alexnet [13], VGG-16, and VGG-19 [24]. The object class probability vectors produced by these networks, per $l \times l$ patch, was converted to its natural parameter form, as described in [5], to generate the vector $p(x)$ of Figure 3. The PCA layer reduced this $1,000$ dimensional vector to the one with 500 dimensions, which was used to compute the MFA-FV. Input images were resized, by making the smaller side 512-pixel long and maintaining the original aspect ratio. Three patch sizes, $l \in \{96, 128, 160\}$ were used, producing between 590 and 1000 patches per image.

Table 2. Effect of regularization strength on MFAFVNet classification accuracy.

AlexNet					
λ	0.01	0.1	1	10	100
Accuracy	70.69	71.11	71.44	71.42	71.43
VGG-16					
λ	0.01	0.1	1	10	100
Accuracy	79.79	80.19	80.3	80.12	80.14

The MFA contained 100 mixture components and a 10 dimension latent variable subspace. This produced a vector of $500 \times 100 \times 10$ dimensions at the output of the MFA-FV layer. The parameters of the fully connected layer (FC9) at the network output were initialized randomly. The initialization of the resulting parameters is discussed below. Layer FC9 was learned with a learning rate of 0.001 and all other layers with a learning rate of 0.00001. Momentum and weight decay were set to 0.9 and 0.0005 respectively. For both datasets, the complete network was trained on 10 epochs. As is costumary in the literature, some results are presented for the combination of the MFA-FV and the Places network [29], a network learned on the large Places scene dataset. In this case, the output of the L2 normalization layer of the MFAFVNet was concatenated with the output of the penultimate layer of the Places network.

4.2. Parameter Initialization

Our experiments have shown that a good initialization of the the PCA and MFA-FV layers can lead to substantial gains in classification accuracy. The PCA layer was initialized by a PCA transformation learned from all patches at the output $p(x)$ of Alexnet or VGG. The low dimensional vectors at the output of the PCA layer were then used to learn the MFA parameters with the EM algorithm of [8]. Table 1 compares this initialization to one where all parameters are randomly initialized with a zero mean Gaussian distribution of standard deviation 0.01. The results in the table refer to a single patch size of 96 and $\lambda_1 = \lambda_2 = 1$, but we observed a similar behavior for other configurations. The performance of the randomly initialized network is 2% weaker on MIT Indoor and about 4% weaker on SUN. It is clear that the optimization has strong local minima, and it is important to rely on an initialization with a strong statistical interpretation.

Table 3. Effect of patch scale on MFAFVNet classification

	AlexNet		
	MIT Indoor	SUN	MI
96×96	71.44	54.14	
128×128	71.4	54.03	
160×160	69.89	52.51	
3 scales	75.01	57.15	

4.3. Influence of Regularization

We next investigated the importance of regularization, by considering different values of λ_1 and λ_2 in (26). For simplicity, we considered only the case where $\lambda_1 = \lambda_2 = \lambda$, which was also adopted in the remaining experiments. For small values of λ the network is free to learn a model that does not reflect the constraints of the MFA-FV, i.e. of weak statistical significance. For larger λ the parameters reflect the MFA constraints and the network has a stronger statistical significance. Table 2 presents results on MIT Indoor for different values of λ and a single scale patch with $l = 96$. There is an improvement of up to 1% when λ increases from 0.01 to 1 and performance stays approximately constant for larger λ. This shows that it is important to enforce the statistical significance of the parameters. In all subsequent experiments we have used the value of $\lambda = 1$.

4.4. Impact of Multiple Scales

Various recent works [5, 6, 17] have shown that is important combine multiple patch scales, since objects of different sizes can be informative for scene classification. Table 3 summarizes the effect of patch sizes ($l \in \{96, 128, 160\}$) on the classification accuracy of the MFAFVNet, for the three object recognition models. While scale 96×96 outperforms the other two, results improve substantially when the three scales are combined. This confirms the previous observations of [5, 6, 17] for the benefits of multi-scale feature combination.

4.5. Comparison to Object-based Scene Classifiers

Table 4 compares the MFAFVNet to various previous scene classifiers based on an object recognition network trained on ImageNet [2, 6, 7, 17, 6, 26, 1]. The FV of [2] uses both Alexnet and VGG-16 as CNN model and 10 patch scales. [17] extracts feature vectors at the output of the first fully connected layer, for a single patch size of $l = 128$, and uses them to derive a sparse coding based FV. [6] uses an MFA-based Fisher vector to pool the local features extracted from AlexNet, VGG-16, or VGG-19, but has no fine tuning. [26] simplifies the Fisher vector to allow end to end training of the network.

The MFAFVNet achieves state-of-the-art results on both datasets for both networks, even outperforming [2] which combines patches of ten different scales. The improved per-

accuracy. “3 scale” denotes the combination of three scales.

VGG-16		VGG-19	
IT Indoor	SUN	MIT Indoor	SUN
80.3	62.51	80.5	62.62
78.44	61.47	79.29	61.48
78.01	61.22	78.44	61.31
81.12	64.51	82.66	64.59

formance over [11, 17, 2] is justified by the fact that these works rely on a FV derived from a Gaussian mixture of diagonal covariance and no fine tuning of the network. The improvements over the sparse coding techniques [17, 18] suggest that the MFA is a better model for the statistics of features learned by deep CNNs. Overall, the closest competitor to the MFAFVNet is [6], which also uses an MFA-FV but does not support end-to-end network finetuning. The MFAFVNet improves the results of this method by $1-2\%$, even though [6] concatenates Fisher vectors of different patch scales, to form a vector that is three times as long as that of the MFAFVNet.

Somewhat surprising is the improvement of 6% over the only other method that tried to train the network end-to-end [26]. We have found that this is due to their simplification of the FV, which includes removing the weights of the mixture components and the normalization terms in the denominators of the posterior probabilities of (21). A similar simplification of the MFAFVNet incurred losses of significant magnitude. Note that the simplification is computationally significant since, as can be seen in Figure 3, these are the only terms that require inputs from the other mixture components k in the softmax of the bottom branch. On the other hand, these are the only connections that allow the mixture components to interact with each other. From a statistical standpoint, in the absence of the normalization, the posterior probabilities are not even probabilities and the model looses coherence. Note that much of the computation of EM is aimed to get the posterior probabilities right, since they determine the mixture assignments of the samples x_i. In fact, the E-step is mostly about getting good estimates of these probabilities.

4.6. Comparison to Scene Classifiers

[29] trained a network with the same architecture as Alexnet or VGG for scene image classification directly from the Places scene dataset. This contains 2.4M scene images. Comparisons to this network test the effectiveness of representations such as the Fisher vector for transfer learning. Since the dimension of the feature vector extracted by the Places network is 4096, the dimension of the MFA-FS was reduced to this value in these experiments[1]. A comparison of the two approaches is presented in Table 5. Interestingly, the object-based network outperforms the Places network by a significant amount (up to 6%). This is likely due to the fact that scenes involve complicated combinations of objects, which may appear at different scales and poses. A network that is trained holistically, i.e. over the whole image, likely has difficulty in inferring these object cues. On the other hand, the combination of the object-based network and the pooling operation of the Fisher vector is basically just learning the statistics of object appearances and some aspects of their configurations in the scene, e.g. relative properties such objects that appear at different sizes in a class of scenes. In any case, these results show that objects are informative for scene classification. As far as we know, this is also the only task on which transfer learning outperforms the training of a deep network directly from a large dataset of the target domain.

Table 4. Performance of sccene classifiers based on object recognition models.

Method	MIT Indoor	SUN
AlexNet-based		
Sparse Coding [17]	68.2	-
VLAD [11]	68.88	51.98
FV [5]	72.86	54.4
MFA-FS [6]	73.58	55.95
FV+FC [2]	74.4	-
MFAFVNet	75.01	57.15
VGG-based		
Compact BN [7]	76.17	-
Deep FisherNet [26]	76.48	57.91
Full BN[7]	77.55	-
Sparse Coding [18]	77.6	-
H-Sparse [18]	79.5	-
MFA-FS [6]	81.43	63.31
FV+FC [2]	81.0	-
MFAFVNet	**82.66**	**64.59**

Table 5. Comparison to a scene classifier learned on Places.

Method	MIT Indoor	SUN
AlexNet		
Places	68.24	54.3
MFAFVNet	74.86	56.96
VGG-16		
Places	79.47	61.32
MFAFVNet	**80.72**	**64.08**

4.7. Combined networks

It is also possible to combine the object- and scene-based models. This is, in fact, done in most previous works and shown to improve performance. Following the standard practice in these experiments, we concatenate the feature vectors extracted by the two networks and classify the resulting vector with a linear SVM of hyperparameter $C_{svm}=2$. These experiments did not use VGG-19, which has not been used in the Places network.

Table 6. Performance of combinations of object-based and scene-based scene classifiers.

Method	MIT Indoor	SUN
AlexNet		
MetaClass+Places [27]	78.9	58.11
FV+Places [5]	79.0	61.72
MFA-FS+Places [6]	79.86	63.16
MFAFVNet+Places	80.47	64.1
VGG-16		
Deep FIsherNet+Places [26]	78.81	59.7
MFA-FS+Places [6]	87.23	71.06
MFAFVNet+Places	**87.97**	**72.01**

Table 6 shows the results obtained by combining the two networks. The combination of the Fisher vector with the holistic scene representations achieves a big improvement (up to 8%) over the performace of either of the representations independently, on both MIT Indoor and SUN. Since the networks capture complementary information - the scene gist for Places and the object composition of the scene for the MFAFVNet - this suggests that these two classes of information are important, even complementary, for scene classification. Table 6 also shows that, even after combination with Places, the MFAFVNet achieves the best results among all Fisher vector representations. These results are, to the best of our knowledge, the state-of-art for scene image classification.

5. Conclusion

In this work, we considered the transfer of a deep CNN trained for object recognition to the task of scene image classification. An embedded implementation of the MFA-FV was proposed. This enabled the design of a network architecture, the MFAFVNet, that can be trained in an end to end manner. The new architecture is based on a MFA-FV layer that implements a statistically correct version of the MFA-FV, through a combination of network computations and regularization. When compared to previous neural implementations of Fisher vectors, the MFAFVNet relies on a more powerful statistical model and a more accurate implementation. When compared to previous non-embedded models, the MFAFVNet relies on a state of the art model, which is now embedded into a CNN. Experiments have shown the importance of maintaining a valid statistical interpretation for the network, through proper initialization and regularization and the benefits of end to end training. The MFAFVNet achieves state of the art performance on scene classification, both as an object-based scene model and when combined with the holistic Places representation.

---

[1] Note that this explains the slight difference to the results in Table 3.