Fisher Vectors Meet Neural Networks: A Hybrid Classification Architecture

Florent Perronnin and Diane Larlus
Computer Vision Group, Xerox Research Centre Europe

Abstract

Fisher Vectors (FV) and Convolutional Neural Networks (CNN) are two image classification pipelines with different strengths. While CNNs have shown superior accuracy on a number of classification tasks, FV classifiers are typically less costly to train and evaluate. We propose a hybrid architecture that combines their strengths: the first unsupervised layers rely on the FV while the subsequent fully-connected supervised layers are trained with back-propagation. We show experimentally that this hybrid architecture significantly outperforms standard FV systems without incurring the high cost that comes with CNNs. We also derive competitive mid-level features from our architecture that are readily applicable to other class sets and even to new tasks.

1. Introduction

The goal of image classification is to tag an image with one or multiple class names based on its content. Our primary focus in this work is on learning classifiers on a large scale, *i.e*. when many classes and images are involved. This topic has received much attention [10, 53, 43, 31, 28, 47, 11] thanks in large part to the release of datasets such as ImageNet [12] and to the organization of public competitions such as the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [42] since 2010.

In 2010-2011, the leading approaches to ILSVRC [43, 31] were based on the Fisher Vector [38, 39] (FV) or the closely related Super Vector (SV) [57]. These pipelines involve extracting local descriptors, encoding them with high-order statistics, aggregating them, and feeding them to kernel classifiers (*e.g*. SVMs). However, in 2012 it was shown that Convolutional Neural Networks [30] (CNNs) trained in a supervised fashion with large amounts of labeled data outperformed the FV [28]. CNNs are feed-forward architectures involving multiple computational layers that alternate linear operations such as convolutions and non-linear operations such as max-pooling. Since then CNNs have consistently led the classification task at ILSVRC [28, 56, 46, 48].

Despite their success, deep architectures come with challenges. This includes the requirement for large amounts of training data, their high computational cost which makes training on GPUs [28] or on very large clusters [29] a necessity, or their lack of geometric invariance [17]. This explains why the FV is still very competitive for certain tasks – see *e.g*. the winning system [19] at the Fine-Grained Visual Competition 2013 [3]. Hence, several works [47, 49] have started exploring the combination of FVs and CNNs.

Our **first contribution** is a novel hybrid architecture that combines the best of both worlds – see Fig. 1. Its first layers are unsupervised and involve the computation and dimensionality reduction of high-dimensional FVs. This is followed by a set of supervised fully connected Neural Network (NN) layers – akin to a Multi-Layer Perceptron (MLP) – trained through back-propagation.

The motivation behind our hybrid architecture is threefold. First, since the convolutional layers of the CNN share much with the coding/aggregation steps of the FV (see section 3.3), we wanted to understand whether FVs were competitive with representations learned by the convolutional layers of the CNN. Our experiments show that the accuracy of our architecture comes close to that of the CNN model of Krizhevsky *et al*. [28]. Second, while FVs are designed to work well with linear classifiers (because they correspond to the explicit feature map of the Fisher Kernel), we wanted to understand whether FV classification could be improved with non-linear classifiers. Our work is the first to provide a positive answer thus showing that much still needs to be learned about the FV. Third, in speech recognition MLPs on top of "handcrafted" features were the state-of-the-art until recently [55, 9]. Hence, we wanted to understand how such a combination performed on image classification.

Because it is unpractical to collect large amounts of labeled data for each new task, we are also interested in transferring the mid-level features learned by our architecture. Transferring features derived from image classifiers either to different class sets or even to new tasks (*e.g*. image retrieval or object detection) has been a very active research topic [51, 50, 18, 13, 34, 1, 16]. Our **second contribution** is to show that the mid-level features derived from our hybrid architecture are competitive with those derived from CNNs.

The remainder of the article is organized as follows. Section 2 reviews related works. In section 3 we describe the

proposed architecture and explain how it relates to standard FV and CNN pipelines. Section 4 validates the proposed architecture experimentally. We first show its competitiveness for large-scale classification on ILSVRC'10 and ILSVRC'12. We then show that the mid-level features we derive from the ILSVRC classifiers can be transferred to other classification tasks on PASCAL VOC'07 and VOC'12 and to instance-level image retrieval on Holidays and UKB.

2. Related Work

We now review hybrid systems that combine FVs (or related representations such as the VLAD [26]) and NNs and works that derive representations from image classifiers.

Hybrid systems. Simonyan *et al*. [47] define a FV layer as a set of five operations: i) FV encoding, ii) supervised dimensionality reduction, iii) spatial stacking, iv) ℓ_2 normalization and v) PCA dimensionality reduction. Such FV layers are stacked into a FV network. Peng *et al*. [37] explored a similar idea for action recognition. Sydorov *et al*. [49] on the other hand propose to improve on the FV by jointly learning the SVM classifier and the GMM visual vocabulary. This is inspired by back-propagation as used to learn NN parameters: the SVM gradients are back-propagated to compute the GMM gradients. This idea was extended to the VLAD [36]. Our hybrid architecture is orthogonal to and can be combined with [47, 37] and [49, 36] – see the conclusion. Finally, Gong *et al*. [17] address the lack of geometric invariance of CNNs by extracting CNN activations from large patches, embedding them using VLAD encoding and aggregating them at multiple scales.

Mid-level features from classifiers. A simple approach to compute a representation from a set of image classifiers is to input an image to these classifiers and to stack their outputs into a vector [51, 50]. An issue with this approach is that the number of classes constrains the representation dimension. To control this dimension, Bergamo *et al*. [4] and Gordo *et al*. [18] proposed to learn classifiers with an intermediate hidden layer whose number of units can be parametrized and to represent a new image as the output of this intermediate layer. A natural extension is to learn deeper architectures, *i.e*. architectures with more than one hidden layer, and to use the output of these intermediate layers as features for the new tasks. Krizhevsky *et al*. [28] showed qualitatively that the output of the penultimate layer could be used for image retrieval. This finding was quantitatively validated for a number of tasks including image classification [13, 34, 56, 6, 41], image retrieval [41, 1], object detection [16], and action recognition [34]. We show that the output of the intermediate layers of our architecture can also be used as generic image representations.

3. System Architecture

Fig. 1 illustrates the proposed architecture. The first set of unsupervised layers relies on the FV framework. The resulting representation is then fed to a set of fully connected supervised layers. We first detail the two sets of layers of our architecture. We then highlight the main differences between this architecture and standard FV and CNN pipelines.

3.1. Unsupervised layers

Architecture. Our architecture involves three unsupervised layers that sequentially perform local feature extraction, FV encoding and dimensionality reduction.

Local feature extraction layer (u_1). Our framework is independent of a particular choice of local descriptors. Following [45] we use SIFT and color descriptors but other local descriptors could have been included. We resize images to 100K pixels while keeping the aspect ratio. We extract 24×24 patches every 4 pixels at 5 scales thus leading to approximately 10K patches per image. From each patch we extract two types of local descriptors: 128-dim SIFT descriptors [32] and 96-dim Local Color Statistics (LCS) [8]. Their dimensionality is reduced with PCA respectively to 77-dim and 45-dim. Following [44] we concatenate to these descriptors the xy coordinates as well as the scale of the patches, thus yielding 80-dim and 48-dim descriptors.

FV encoding layer (u_2). Perronnin and Dance [38] applied the Fisher Kernel [22] framework to images by using as a generative model of local descriptors a Gaussian Mixture Model (GMM). This was later refined by Perronnin *et al*. [39] and was shown to be a state-of-the-art patch encoding technique [5, 27, 45]. Given a GMM with K Gaussians, let us denote its parameters by $\lambda = \{w_k, \mu_k, \sigma_k, k = 1 \dots K\}$ with w_k, μ_k, and σ_k respectively the mixture weight, mean vector and standard deviation vector of Gaussian k (assuming a diagonal covariance). In the FV framework, the D-dim descriptor x is embedded with a function $\Phi(x) = [\varphi_1(x), \dots, \varphi_K(x)]$ into a $2KD$-dim space where each function $\varphi_k(x) : \mathbb{R}^D \rightarrow \mathbb{R}^{2D}$ is defined by:

$$\varphi_k(x) = \left[\frac{\gamma(k)}{\sqrt{w_k}}\left(\frac{x-\mu_k}{\sigma_k}\right), \frac{\gamma(k)}{\sqrt{2w_k}}\left(\frac{(x-\mu_k)^2}{\sigma_k^2} - 1\right)\right] \quad (1)$$

and $\gamma(k)$ denotes the soft assignment of descriptor x to Gaussian k. We use a GMM with 1K=1,024 Gaussians. The per-patch FVs are aggregated with sum-pooling, square-rooted and ℓ_2-normalized. One FV is computed on the SIFT descriptors and one on the LCS descriptors. The two FVs are concatenated into a 256K-dim representation.

FV dimensionality reduction layer (u_3). These FVs are too high-dimensional to be processed as is both for memory and computational reasons. To address this issue we use a simple PCA dimensionality reduction followed by whitening and ℓ_2-normalization. PCA followed by whitening has been recently rediscovered as a useful transformation for tasks such as retrieval and classification [23] or detection

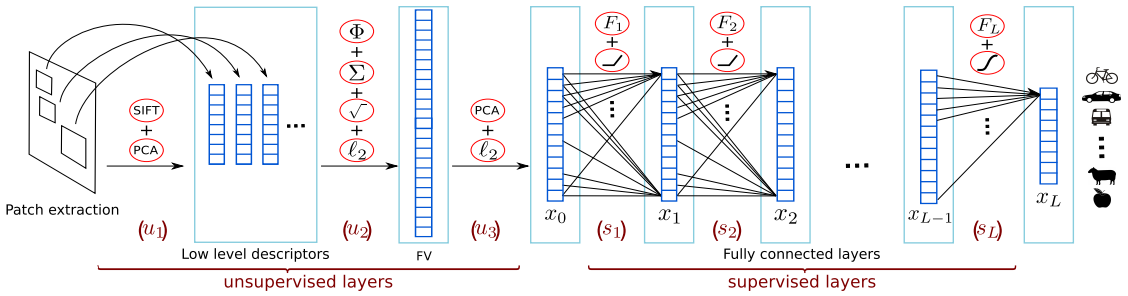


Figure 1. Proposed hybrid architecture. Image patches are described by PCA-projected SIFT descriptors (u_1). These descriptors are then embedded using FV-encoding and aggregated. This image-level representation is normalized by square-rooting and ℓ_2-normalization (u_2). The resulting FV is PCA-projected and re-normalized (u_3). The supervised layers (s_1), (s_2), .. (s_{L-1}) involve a linear projection followed by a reLU. The last layer (s_L) involves a linear projection followed by a softmax or a sigmoid and produces the label estimates. Our architecture can be considered deep as it stacks several unsupervised and supervised layers. To simplify the figure, we assumed that only SIFT descriptors were extracted. However, in practice, we also leverage color descriptors (see text).

[20]. In our experiments, the FV dimensionality is reduced from 256K down to 4K. This value stroke a good compromise between efficiency gain and accuracy loss.

Learning. All previous layers are learned in an unsupervised fashion. A significant advantage of such a strategy is that the corresponding parameters can be learned once and for all on a representative set of unlabeled images. In our experiments, this is done on a subset of 10K images of the ILSVRC'10 training set. The PCA reduction of the local descriptors and the GMM were learned on a subsample of 1M local descriptors randomly extracted from these 10K images. Given that the number of training images was smaller than the FV dimensionality (10K<256K), we learned the FV-level PCA in the dual.

3.2. Supervised layers

The PCA-reduced FVs output by the last unsupervised layer (u_3) are input to the first supervised layer (s_1).

Architecture. The supervised part of our architecture consists of a set of L fully connected layers that involve a linear projection followed by a non-linearity. If x_{n-1} is the input of (s_n), x_n is its output and if σ denotes the non-linearity, we have $x_n = \sigma(F_n(x_{n-1}))$ where $F_n(x) = W_n x + b_n$ and W_n and b_n are the parameters to be learned. For the intermediate hidden layers (s_1) to (s_{L-1}), we use a rectified Linear Unit (reLU) non-linearity $\sigma(x) = \max(0, x)$ which showed improved convergence with respect to sigmoid non-linearities both in computer vision [28] and speech recognition [55, 9]. As for the output of the last layer (s_L), in the mono-label case (*e.g.* ImageNet) we use a softmax non-linearity which given the vector $x = [x(1), \ldots, x(E)]$ performs the following transformation: $x(e) \rightarrow \exp(x(e))/\sum_{i=1}^{E} \exp(x(i))$. This ensures that output scores are non-negative and sum to unity and thus provides a probabilistic-like output. In the multi-label case (*e.g.* PASCAL VOC) we use a sigmoid non-linearity: $x(e) \rightarrow 1/(1+\exp(-x(e)))$. This enables multiple output classes to have strong activations (*i.e.* close to 1). We experimented with a number of supervised layers varying from 1 to 4 (*i.e.* 0 to 3 hidden supervised layers).

Learning. To train the parameters of these supervised layers, we use the standard objective that involves minimizing the cross-entropy between the network output and the ground-truth. We assume that we have N labeled training images with $y_{n,c} = 1$ if image n is tagged with class label c and $y_{n,c} = 0$ otherwise. We denote by $\hat{y}_{n,c}$ the prediction of label c for image n as output by the last supervised layer. By definition $\hat{y}_{n,c} = x_L(c)$. In the mono-label case, we maximize:

$$\sum_{n=1}^{N} \sum_{c=1}^{C} y_{n,c} \log(\hat{y}_{n,c}) \tag{2}$$

while in the multi-label case, we maximize:

$$\sum_{n=1}^{N} \sum_{c=1}^{C} y_{n,c} \log(\hat{y}_{n,c}) + (1 - y_{n,c}) \log(1 - \hat{y}_{n,c}). \tag{3}$$

We initialize the weights from a zero-mean Gaussian. For the optimization, we use back-propagation with mini-batch stochastic gradient descent and a batch size of 128. The learning rate is fixed during an iteration (one pass over the data). Once the validation error does not decrease any more (*i.e.* it decreases by less than 0.1% absolute between two iterations) we divide the step-size by 10. We repeat this procedure twice, *i.e.* we have 3 learning rates. The starting step-size was cross-validated in our experiments. To avoid over-fitting, we use drop-out [28] at the input of all supervised layers. We use the same drop-out rate for all layers although using different values might improve results [9].

3.3. Comparison with FV and CNN pipelines

Relationship with FVs. With respect to the standard large-scale classification FV pipeline [45], our architecture exhibits two significant differences.

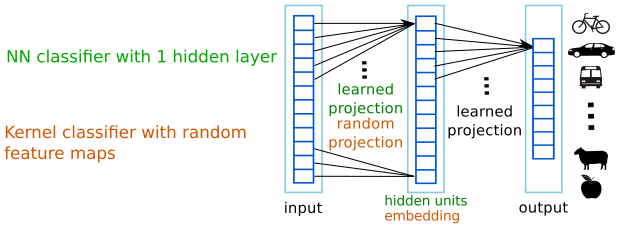


Figure 2. Comparison of a Neural Network classifier with a single hidden layer and a kernel classifier based on random feature maps.

The first one is the PCA that we use to reduce the dimension of the FVs, both for memory and computational reasons. To handle the challenge of high-dimensional vectors, Sánchez and Perronnin [43] use Product Quantization (PQ) [25] which does not reduce the feature dimension. This is justified by their choice of a linear classifier whose capacity is the feature dimension + 1. In the linear case, high-dimensional features are necessary to get a high enough capacity. In our case, since we use a non-linear network, the capacity is not constrained by the dimension of the features that enter the supervised layers.

The second difference is the fact that, in the standard FV classification pipeline, supervised learning relies on kernel classifiers, and generally on linear classifiers. In what follows, we distinguish the linear and non-linear cases. The linear case can be readily interpreted as a special case of our architecture where there is a single supervised layer, *i.e.* L=1. The non-linear kernel case can be understood as an architecture with two layers (L=2) where the parameters of the first layer are not learned but fixed – see for instance [2]. In a two-layer architecture, the first layer can be interpreted as a non-linear mapping Ψ and the second layer as a linear classifier in the embedding space. Interestingly if Ψ involves random Gaussian projections followed by a reLU non-linearity, Ψ is the feature map of the arc-cosine kernel of order one as proposed by Cho and Saul [7]. This shows a close connection between NNs with a single hidden layer and a reLU non-linearity, and arc-cosine kernel classifiers. The main difference is that in the former case both layers are trained in a supervised fashion while in the latter only the second layer is learned in a supervised fashion while the parameters of the first layer are set at random – see Fig. 2. In section 4, we show the superiority of the proposed architecture over the arc-cosine kernel alternative.

Relationship with CNNs. While FVs and CNNs seem quite different at first glance, several authors pointed out that they have a lot in common [47, 49]. Indeed, the FV extraction pipeline can also be interpreted as a series of layers that alternate linear and non-linear operations and can be paralleled with the convolutional layers of the CNN. The main difference between the proposed architecture and the standard CNN [28] is that our first layers are unsupervised while in [28] all layers are learned in a supervised fashion.

An advantage of our architecture is that one can view the learning of the first layers as an unsupervised pre-training stage. Since these layers do not need to be retrained for each task, our architecture is fast to train (see "convergence" in section 4.1). As a corollary, we can train efficiently multiple models for bagging purposes, starting from different random initializations, as the unsupervised layers are fixed. Also, we note that the cost of forward-passing one sample through the unsupervised layers is much higher than the cost of forward-passing through the supervised layers (by a factor $\sim$100). Hence, at inference time when bagging multiple models, we perform only once the costly operations through the unsupervised layers and we repeat only the costless operations through the supervised layers.

4. Experiments

We first provide a detailed analysis of the proposed architecture on ILSVRC'10. We then report results on ILSVRC'12. Finally, we show that we can extract from the intermediate levels of our architecture mid-level features which can be used for classification (on PASCAL VOC'07 and VOC'12) and retrieval (on Holidays and UKB).

4.1. A detailed study on ILSVRC'10

ILSVRC'10 [42] contains 1K classes and $\sim$1.4M images: 1.2M for training, 50K for validation, and 150K for testing. We follow the challenge training/validation/test protocol and evaluation measure, *i.e.* the top-5 error rate.

FV dimensionality reduction. We first quantify the influence of the FV dimensionality reduction in layer (u_3). In this experiment, we use a single supervised layer for classification, *i.e.* a linear logistic classifier. Where we do not perform dimensionality reduction, we use PQ to reduce the memory footprint of the 256K-dim FVs. Using the standard setting of [45] (FV divided into sub-vectors of 8 dimensions and 8 bits per sub-vector), we obtain a 25.5% top-5 error. This is exactly the accuracy reported in [45] (with 1M-dim descriptors in their case). If we perform PCA to 4K-dim, we obtain 27.2% which is only 1.7% higher. However, this 64-fold compression significantly speeds-up subsequent computations in supervised layers.

Model complexity and drop-out. We now study the influence of two key parameters which control the complexity of our supervised layers: the number of supervised layers and the number of units per layer (in all the following experiments, the number of units is constant across hidden layers). As more complex models might more easily overfit, we study the influence of these two parameters jointly with the drop-out rate. Results are summarized in Fig. 3.

We make the following observations. Drop-out does not help when training a linear classifier (0 hidden layer), most certainly because we operate on low-dimensional descriptors (4K) and therefore there is little opportunity to overfit.

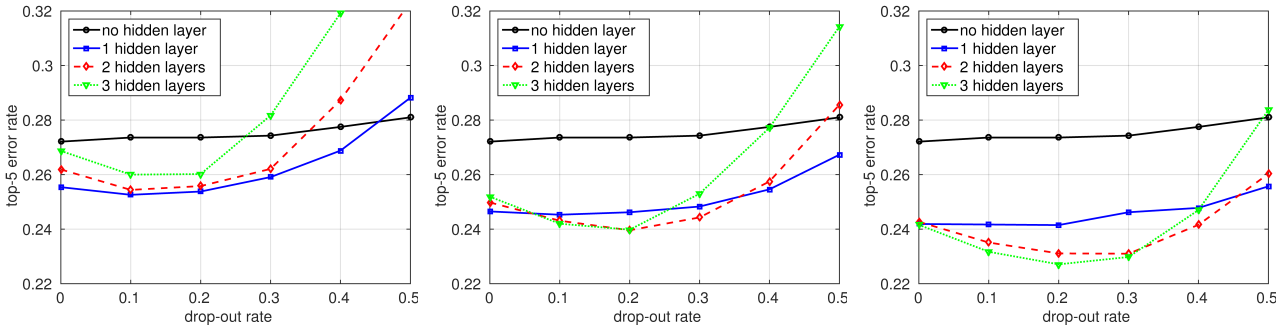


Figure 3. Top-5 error as a function of the drop-out rate for a variable number of hidden layers and a variable number of hidden units per layer: 1,024 (left), 2,048 (middle) and 4,096 (right) units per hidden layer (always the same number of units in each hidden layer).

The 0.5 value which is generally used in CNNs [28] does not work in our architecture and the optimal seems to be around 0.2 whatever the number of hidden layers and units. Finally, more hidden layers does not seem to help for too small a number of hidden units (see Fig. 3 left). However, it makes a difference for a larger number of units (see Fig. 3 right) and the performance saturates with 3 hidden layers.

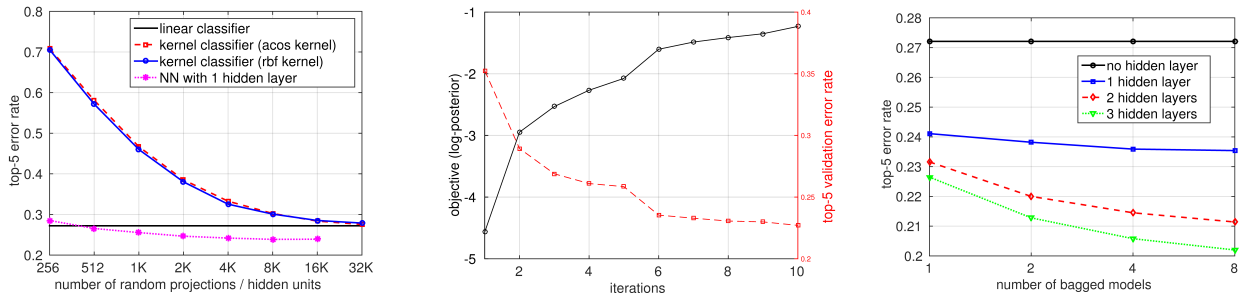
Comparison with kernel methods. In section 3.3, we discussed the link between kernel classifiers and NNs with one hidden layer. We especially highlighted the close relationship between the arc-cosine kernel classifier and NNs with 1 hidden layer employing reLU non-linearities. Therefore, we compare these two alternatives. Since the arc-cosine kernel is not common in computer vision, we also compare with the RBF kernel. Because of the large number of training samples (1.2M), we train the non-linear kernel classifiers in the primal by leveraging explicit feature maps. In the arc-cosine kernel case, we use the feature map proposed by Cho and Saul [7] which involves a random Normal projection followed by a reLU non-linearity. In the RBF kernel case, we use the explicit feature map of Rahimi and Recht [40] which involves a random Normal projection, a random offset and a cosine non-linearity. We cross-validated the kernel bandwidth of the RBF.

Results are reported in Fig. 4(a) as a function of the dimensionality of the feature map (for the arc-cosine and RBF kernels) and of the number of units in the hidden layer (in the NN case). The arc-cosine and RBF kernels perform similarly and significantly worse than a NN with one hidden layer, especially for a small number of random projections / hidden units. Just to perform on par with the linear baseline (NN with no hidden layer), both the arc-cosine and RBF kernels need to project the original samples in a 32K-dim space (8 times larger than the original 4K-dim features). While it might be possible to obtain better results with these kernels by projecting in even higher dimensional spaces, this is much more costly than the NN alternative.

Convergence. We report in Fig. 4(b) the evolution of the objective as well as the validation accuracy over the iterations when learning a model with 3 hidden supervised layers with 4K units and a drop-out rate of 0.2 (the best results reported in Fig. 3 right). We observe that the convergence is much faster than what is reported in [28] (10 passes in our case *vs*. 90 in [28]). We believe that this is because the first layers of our architecture are pre-trained in an unsupervised manner and are kept fixed during the supervised learning process, thus reducing the number of parameters to be learned. Running 24 threads on a Linux server with Intel®Core™ i5-2400 CPUs @ 3.10GHz and 128GBs of RAM, a pass over the data is carried out in $\sim$ 1h with our C++ implementation. Hence, the whole training process takes about 10h on a single server. Using multiple servers, it is possible to train in parallel multiple models from different initializations and to bag their outputs.

Bagging. We consider architectures with 0, 1, 2 and 3 hidden supervised layers. For each architecture, we train 8 different models using different random seed initializations. To combine the output of multiple networks, we multiply the probabilistic outputs. Results are presented in Fig. 4(c) when combining 2, 4 or 8 models. The larger the number of hidden layers, the greater the impact of bagging. This does not come as a complete surprise. Indeed, when there is no hidden supervised layer the objective to be optimized is convex and therefore different initializations lead to the same solution. The larger the number of hidden layers, the greater the opportunity to converge to different solutions. In the case of 3 hidden layers, we go from a top-5 error rate of 22.6% with no bagging to 20.2% with bagging.

Data augmentation. A standard way to avoid overfitting when training classifiers and especially deep architectures is to artificially augment the data by introducing transformed versions of the original images. For instance, Krizhevsky *et al*. [28] uses a predefined set of crops, flips and color perturbations. To avoid manually choosing the transformations, we run the Iterative Transformation Pursuit (ITP) algorithm of Paulin *et al*. [35] on a subset of ILSVRC'10 (using all 1K classes but only 30 images per class) and choose a set of 4 transformations on top of the original images. Hence,



(a) (b) (c)

Figure 4. Top-5 error on ILSVRC'10. (a) Comparison of a linear classifier, a NN with 1 hidden layer and kernel classifiers with explicit feature maps. (b) Log-posterior objective and validation error as a function of the number of passes over the training data. (c) Impact of bagging for NNs with 0, 1, 2 and 4 hidden layers and 4K units per hidden layer. For each model, we chose the best validation drop-out rate.

Proposed hybrid			
hidden units	data augm.	bagging	top-5 error (in %)
4K	no	no	22.6
4K	no	yes	20.2
4K	yes	no	19.8
4K	yes	yes	18.1
8K	yes	yes	**17.6**

State-of-the-art		
[45]	FV	25.5
[35]	FV + ITP	25.1
[47]	deep Fisher network	20.8
[35]	DeCAF + ITP	18.6
[28]	AlexNet CNN	**17.0**

Table 1. Impact of data augmentation and bagging on ILSVRC'10 for an architecture with 3 hidden supervised layers. We also compare with the state-of-the-art.

the number of images is multiplied by 5. Data augmentation is used both at training and test time. At test time, we multiply the predictions corresponding to the different transformations of the images.

In the case where we have no hidden supervised layer (*i.e.* we learn a linear classifier on top of the 4K-dim PCA-reduced FVs), the error rate is decreased from 27.2% with no data augmentation to 25.2% with data augmentation. In the case where we have 3 hidden supervised layers, we report the results in Table 1. We observe a 2.8% improvement without bagging (from 22.8% to 19.8%) and a 2.1% improvement with bagging (from 20.2% to 18.1%).

Learned representations. It is interesting to visualize what is learned by our architecture. For this purpose, we inspected the output of the intermediate unsupervised and supervised layers – see Fig. 5 and its caption. The conclusion is that the output of supervised layers is more semantically consistent than the output of unsupervised layers. This makes for a strong case to use the output of supervised layers as generic mid-level features – see section 4.3.

Comparison with the state-of-the-art. To obtain the best possible results with our architecture, we use three hidden supervised layers with 8K hidden units. We perform data augmentation (original image + 4 transforms) and bag 8 models. This leads to a top-5 error rate of 17.6%. Note that if we perform only training set (but not test set) augmentation we obtain 19.0%. We compare these results with the state-of-the-art in Table 1. We report significantly better results than a FV baseline [45] and the FV + ITP results [35]. Our system is competitive with respect to CNNs [28, 35]. Note that Simonyan *et al.* [47] report a 14.3% top-5 error by combining deep Fisher networks and CNNs. Similarly, we could combine the output of our model with that of a standard CNN.

Proposed hybrid		(Bag-8)	19.8
[42]	Best challenge FV results		26.2
[28]	AlexNet CNN	(Bag-5)	16.4
[56]	Zeiler CNN	(Bag-6)	14.7
[46]	OverFeat CNN	(Bag-7)	13.2
[48]	Simonyan CNN	(Bag-7)	**7.5**

Table 2. Comparison with the state-of-the-art on ILSVRC'12. When relevant, we indicate how many models are bagged. Top-5 error (in %).

4.2. Results on ILSVRC'12

We also benchmark our architecture on ILSVRC'12 [42] for which many results have been reported recently in the literature. This dataset contains 1K classes, ∼1.2M training images and 50k validation images. Since the test labels are not available, we report results on the validation set as is standard practice. We use training set augmentation and bag 8 models. To avoid overfitting our architecture to the validation data, we use the best architecture of ILSVRC'10: three hidden supervised layers with 8K hidden units and a 0.2 drop-out rate in all supervised layers. We obtain a top-5 error rate of 19.8%.

We compare to the state-of-the-art in Table 2. Our accuracy is significantly better than the best FV-based results reported on this dataset (during the challenge). However, it is somewhat below the CNN systems [28, 56, 46] and significantly below the recent very deep architecture by Simonyan

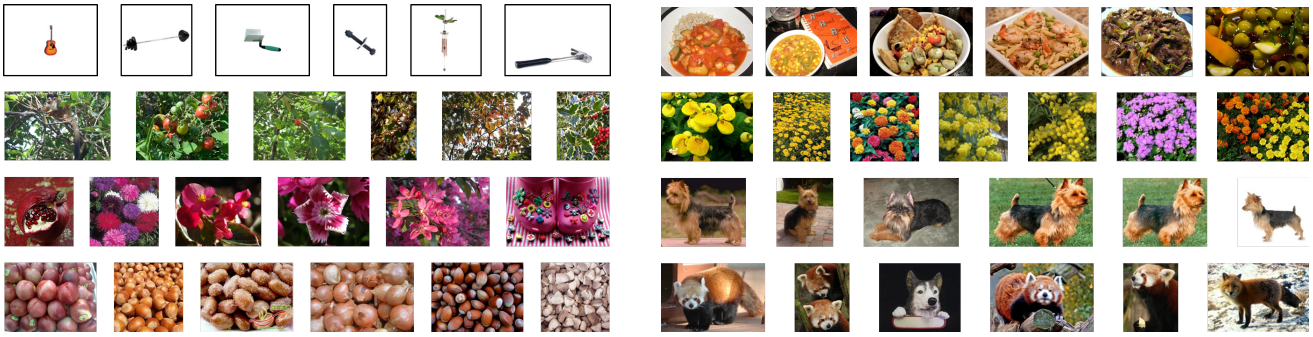


Figure 5. Visualizing the output of intermediate layers for an architecture with 3 hidden supervised layers and 4K units per hidden layer. For a given layer, we show the images with the strongest responses for four random neurons (one line per neuron). Left: x_0 is considered, *i.e.* the output of the last unsupervised layer (u_3). While the neuron outputs are visually consistent (*e.g.* same color), they are not necessarily semantically consistent. This is because all the learning prior to (u_3) is unsupervised. Right: x_3 is considered, *i.e.* the output of the penultimate layer (s_3). Supervised learning makes the neuron outputs much more semantically consistent.

and Zisserman [48]. We still believe that our architecture has merits of its own. Especially, it does not require GPUs or very large clusters [29] to be trained. Training the system from scratch on a small CPU cluster (8 machines) takes ∼10 days: 7 days for the extraction of the PCA-reduced FVs and 3 days for the supervised training of the 8 models. This is to be compared *e.g.* with the 2-3 weeks reported by Simonyan and Zisserman in [48] to train a single model with a four NVIDIA Titan black GPU system.

4.3. Transferring mid-level features

In standard CNN architectures trained on large datasets such as ImageNet, the outputs of intermediate layers have been used as mid-level features in a large variety of new tasks. We now show that our architecture has the same property. We start from a model learned on ILSVRC'12 (3 hidden supervised layers with 4K units and a drop-out rate of 0.2) and study how such a classifier can transfer to new tasks for which we have less (or no) training data. We consider both classification and instance retrieval tasks.

4.3.1 Classification

Experiments are conducted on PASCAL VOC'07 and VOC'12 [14, 15] which contain 20 classes. VOC'07 is split into 2,501 training, 2,510 validation and 4,952 test images. VOC'12 is composed of 5,717 training, 5,823 validation, and 10,991 test images. Following the PASCAL guidelines, we first propose a study on VOC'07 and report a single run on VOC'12 (evaluated on the PASCAL server). On both datasets, we report the mean Average Precision (mAP).

Transfer strategy. We follow the strategy that involves extracting the output of the penultimate layer (x_{L-1} in our case), ℓ_2-normalizing it and feeding it to a classifier [13, 16]. We consider both linear and non-linear classifiers. In the non-linear case, to stay in the spirit of the proposed architecture, we learn a NN with fully connected layers, reLU non-linearities for the hidden layers and sigmoid non-linearities for the last layer (see section 3.2). We cross-validated the number of layers, the number of hidden units and the drop-out rate.

classifier	data augm.	bagging	mAP (in %)
linear	no	no	71.1 ± 0.12
linear	yes	no	72.9 ± 0.19
linear	yes	yes	74.5
non-linear	no	no	73.1 ± 0.13
non-linear	yes	no	74.8 ± 0.25
non-linear	yes	yes	**76.2**

Table 4. mAP on VOC'07 for linear *vs.* non-linear classifiers, with/without data-augmentation and with/without bagging.

Data augmentation and bagging. Since we have a limited amount of labeled training VOC images, we use data augmentation on VOC too. To this end, we apply ITP [35] to select 4 transformations (on top of the original images), and apply them at training and test time. The ITP learning is done on the training and validation sets using a linear classifier. We use the same chosen transformations in both the linear and non-linear cases. We also consider the bagging of multiple systems (8 in our experiments).

Results. We show results in Table 4. As observed for ILSVRC'10, non-linear classifiers improve over linear ones despite the limited amount of training data. Data augmentation and bagging always help. Our best results of 76.2% mAP are obtained by bagging 8 classifiers with 3 hidden layers (2K units per layer).

Comparison with state-of-the-art. We now compare our results with other mid-level representations in Table 5. For VOC'12, we trained the system with the best VOC'07 setting and submitted our results to the PASCAL server[1]. First we report superior performance with respect to the standard FV pipeline [45] and its improved version from [6]. We also

[1]Detailed results available at: http://host.robots.ox.ac.uk:8080/anonymous/ZPBASO.html

Methods	FV	x_0	x_*	Bag-8 + PCA		FV+proj [18]	CNN+VLAD [17]	CNN [1]	CNN [41]
dim	256K	4K	4K	512	4K	512	12K	4K	500
Holidays	82.4	83.5	80.5 ± 0.31	82.7	**84.7**	79.0	80.2	74.9	64.2
UKB	3.35	3.33	3.36 ± 0.01	3.37	**3.43**	3.36	-	**3.43**	3.04

Table 3. Retrieval results on Holidays and UKB. x_* indicates that we use the output of the best intermediate layer.

		VOC'07	VOC'12
Proposed Architecture		76.2	77.7
[45]	baseline FV	63.9	-
[6]	improved FV	68.0	-
[13] from [6]	DeCAF - CNN	73.4	-
[41]	Razavian - CNN	77.2	-
[56]	Zeiler - CNN	-	79.0
[6]	Chatfield - CNN	**82.4**	**83.2**
[34]	Oquab - CNN	77.7	78.7
[21]	He - CNN	80.1	-
[52]	Wei - CNN	81.5	81.7

Table 5. Comparison with other mid-level representations on PASCAL VOC'07 and VOC'12 (mAP in %).

outperform off-the-shelf DeCAF results and perform on par with the results of Razavian *et al*. [41], Oquab *et al*. [34] and Zeiler and Fergus [56]. He *et al*. [21] and Wei *et al*. [52] report better performance but with significantly more complex transfer strategies. [21] uses a spatial pyramid pooling layer, with 50 bins. [52] uses a significantly more complex classification pipeline driven by bounding box proposals. The best results are those of Chatfield *et al*. [6] with a costly architecture. Note that [34] and [52] also report better results with extended class sets (*i.e*. more than the 1,000 classes of ILSVRC'12).

4.3.2 Instance-level retrieval

We study the application of our mid-level features to instance retrieval. We consider two datasets: INRIA Holidays [24] (Holidays) which involves 1,491 images of 500 scenes and the University of Kentucky Benchmark [33] (UKB) which involves 10,200 images of 2,250 objects. We use the standard evaluation measures: mean AP (mAP) for Holidays and 4 times the recall at top 4 on UKB.

Layer choice. We extract and ℓ_2-normalize the output of intermediate layers and compare them with the dot-product. As observed by Yosinski *et al*. [54], the transferability of features decreases as the distance between the base and target tasks increases. For UKB, whose images are closer to the ImageNet ones, we use x_2, while for Holidays, which contains mostly scenes, we use x_1. We also report results for two baselines: using the high-dimensional FVs and the PCA-reduced FVs, *i.e*. x_0.

Bagging. Bagging can be done by fusing the scores of multiple queries. However, the query cost of this late fusion strategy increases linearly with the number of models. Instead, we concatenate the representations produced by the different models, and reduce the dimension of this new representation with PCA-whitening (learned on ILSVRC'10).

Results. All results are reported in Table 3. We also compare with other compact global image representations. We observe that the PCA-reduced FVs obtain very competitive results and that using the best intermediate layer leads to a decrease of performance on Holidays and to only a slight increase on UKB. However, the proposed bagging strategy, that involves PCA and whitening, leads to improvements with a competitive result of 84.7% (resp. 3.43) on Holidays (resp. UKB) for 4K dimensions. [41] reports 84.3% on Holidays and 3.64 on UKB with a costly setting that involves extracting and matching multiple CNN features per image. Our results are obtained by extracting and matching a single descriptor per image. Babenko *et al*. [1] also report better results on Holidays (79.3%) for a CNN that has been retrained using an external dataset of landmarks. This additional information could be used by our architecture as well, *e.g*. by learning an extra set of layers as done on VOC.

5. Conclusion

Summary. We proposed a hybrid architecture for image classification that combines the benefits of FV and CNN pipelines. A first set of unsupervised layers involves the extraction and dimensionality reduction of FVs while the subsequent supervised layers are fully connected. The proposed approach was shown to significantly improve over previous FV systems without incurring the high cost associated with CNNs. Especially, our system does not require training with GPUs or large CPU clusters. We also showed that mid-level representations extracted from our architecture were competitive with the mid-level features extracted from CNN representations, on several target tasks.

Future work. We believe that our current architecture can be improved in its last unsupervised layer, *i.e*. the unsupervised dimensionality reduction of FVs. On the one hand PCA significantly reduces the high-dimensional FVs, and makes the subsequent supervised training and parameter search manageable. On the other hand, FVs contain much fine-grained information [19] that might be discarded by this operation. An alternative inspired by [47] would be to use a supervised dimensionality reduction instead of PCA. Another alternative inspired by [49] would be to back-propagate the gradients to the currently unsupervised layers to fine-tune them. We leave this for future work.

Acknowledgments

This work was done in the context of the Project Fire-ID, supported by the Agence Nationale de la Recherche (ANR-12-CORD-0016).

References

[1] A. Babenko, A. Slesarev, A. Chigorin, and V. S. Lempitsky. Neural codes for image retrieval. In *ECCV*, 2014. 1, 2, 8

[2] Y. Bengio and Y. LeCun. *Large-Scale kernel machines*, chapter Scaling learning algorithms towards AI. MIT press, 2007. 4

[3] A. Berg, R. Farrell, A. Khosla, J. Krause, L. Fei-Fei, J. Li, and S. Maji. Fine-Grained Competition (FGComp). http://sites.google.com/site/fgcomp2013/, 2013. 1

[4] A. Bergamo, L. Torresani, and A. Fitzgibbon. PiCoDeS: Learning a compact code for novel-category recognition. In *NIPS*, 2011. 2

[5] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. *BMVC*, 2011. 2

[6] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: delving deep into convolutional nets. In *BMVC*, 2014. 2, 7, 8

[7] Y. Cho and L. Saul. Kernel methods for deep learning. *NIPS*, 2009. 4, 5

[8] S. Clinchant, G. Csurka, F. Perronnin, and J.-M. Renders. XRCE's participation to imageval. *ImageEval @ CVIR*, 2007. 2

[9] G. Dahl, T. Sainath, and G. Hinton. Improving deep neural networks for LVCSR using rectified linear units and dropout. In *ICASSP*, 2013. 1, 3

[10] J. Deng, A. Berg, K. Li, and L. Fei-Fei. What does classifying more than 10,000 image categories tell us? In *ECCV*, 2010. 1

[11] J. Deng, N. Ding, Y. Jia, A. Frome, K. Murphy, S. Bengio, Y. Li, H. Neven, and H. Adam. Large-scale object classification using label relation graphs. In *ECCV*, 2014. 1

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009. 1

[13] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. DeCAF: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014. 1, 2, 7, 8

[14] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The Pascal visual object classes (VOC) challenge. *IJCV*, 2010. 7

[15] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. 7

[16] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1, 2, 7

[17] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *ECCV*, 2014. 1, 2, 8

[18] A. Gordo, J. Rodríguez-Serrano, F. Perronnin, and E. Valveny. Leveraging category-level labels for instance-level image retrieval. In *CVPR*, 2012. 1, 2, 8

[19] P.-H. Gosselin, N. Murray, H. Jégou, and F. Perronnin. Revisiting the Fisher vector for fine-grained classification. *PRL*, 2014. 1, 8

[20] B. Hariharan, J. Malik, and D. Ramanan. Discriminative decorrelation for clustering and classification. In *ECCV*, 2012. 3

[21] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014. 8

[22] T. Jaakkola and D. Haussler. Exploting generative models in discriminative classifiers. In *NIPS*, 1998. 2

[23] H. Jégou and O. Chum. Negative evidences and cooccurrences in image retrieval: the benefit of PCA and whitening. In *ECCV*, 2012. 2

[24] H. Jégou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. *ECCV*, 2008. 8

[25] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE TPAMI*, 2011. 4

[26] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010. 2

[27] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid. Aggregating local image descriptors into compact codes. *TPAMI*, 2012. 2

[28] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 2, 3, 4, 5, 6

[29] Q. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. Corrado, J. Dean, and A. Ng. Building high-level features using large scale unsupervised learning. In *ICML*, 2012. 1, 7

[30] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Handwritten digit recognition with a back-propagation network. *NIPS*, 1989. 1

[31] Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, L. Cao, and T. Huang. Large-scale image classification: fast feature extraction and SVM training. In *CVPR*, 2011. 1

[32] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 2

[33] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. *CVPR*, 2006. 8

[34] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, 2014. 1, 2, 8

[35] M. Paulin, J. Revaud, Z. Harchaoui, F. Perronnin, and C. Schmid. Transformation pursuit for image classification. In *CVPR*, 2014. 5, 6, 7

[36] X. Peng, L. Wang, Y. Qiao, and Q. Peng. Boosting VLAD with supervised dictionary learning and high-order statistics. In *ECCV*, 2014. 2

[37] X. Peng, C. Zou, Y. Qiao, and Q. Peng. Action recognition with stacked Fisher vectors. In *ECCV*, 2014. 2

[38] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. *CVPR*, 2007. 1, 2

[39] F. Perronnin, J. Sánchez, and T. Mensink. Improving the Fisher kernel for large-scale image classification. *ECCV*, 2010. 1, 2

[40] A. Rahimi and B. Recht. Random features for large-scale kernel machines. *NIPS*, 2007. 5

[41] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *CVPR Deep Vision Workshop*, 2014. 2, 8

[42] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. arXiv, 2014. 1, 4, 6

[43] J. Sánchez and F. Perronnin. High-dimensional signature compression for large-scale image classification. In *CVPR*, 2011. 1, 4

[44] J. Sánchez, F. Perronnin, and T. de Campos. Modeling the spatial layout of images beyond spatial pyramids. In *PRL*, 2012. 2

[45] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the Fisher vector: theory and practice. *IJCV*, 2013. 2, 3, 4, 6, 7, 8

[46] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. OverFeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014. 1, 6

[47] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep Fisher networks for large-scale image classification. In *NIPS*, 2013. 1, 2, 4, 6, 8

[48] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv, 2014. 1, 6, 7

[49] V. Sydorov, M. Sakurada, and C. Lampert. Deep Fisher kernels – End to end learning of the Fisher kernel GMM parameters. In *CVPR*, 2014. 1, 2, 4, 8

[50] L. Torresani, M. Szummer, and A. Fitzgibbon. Efficient object category recognition using classemes. In *ECCV*, 2010. 1, 2

[51] G. Wang, D. Hoiem, and D. Forsyth. Learning image similarity from Flickr groups using stochastic intersection kernel machines. In *ICCV*, 2009. 1, 2

[52] Y. Wei, W. Xia, J. Huang, B. Ni, J. Dong, Y. Zhao, and S. Yan. CNN: single-label to multi-label. arXiv, 2014. 8

[53] J. Weston, S. Bengio, and N. Usunier. Large scale image annotation: learning to rank with joint word-image embeddings. *ECML*, 2010. 1

[54] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. Quantifying the transferability of features in deep neural networks. arXiv, 2014. 8

[55] M. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G. Hinton. On rectified linear units for speech processing. In *ICASSP*, 2013. 1, 3

[56] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*. 2014. 1, 2, 6, 8

[57] Z. Zhou, K. Yu, T. Zhang, and T. Huang. Image classification using super-vector coding of local image descriptors. *ECCV*, 2010. 1