Deep Fisher Kernels – End to End Learning of the Fisher Kernel GMM Parameters

Vladyslav Sydorov* Mayu Sakurada* Christoph H. Lampert
IST Austria, Klosterneuburg, Austria
{vsydorov|msakurada|chl}@ist.ac.at

Abstract

Fisher Kernels *and* Deep Learning *were two developments with significant impact on large-scale object categorization in the last years. Both approaches were shown to achieve state-of-the-art results on large-scale object categorization datasets, such as ImageNet. Conceptually, however, they are perceived as very different and it is not uncommon for heated debates to spring up when advocates of both paradigms meet at conferences or workshops.*

In this work, we emphasize the similarities between both architectures rather than their differences and we argue that such a unified view allows us to transfer ideas from one domain to the other. As a concrete example we introduce a method for learning a support vector machine classifier with Fisher kernel at the same time as a task-specific data representation. We reinterpret the setting as a multi-layer feed forward network. Its final layer is the classifier, parameterized by a weight vector, and the two previous layers compute Fisher vectors, parameterized by the coefficients of a Gaussian mixture model.

We introduce a gradient descent based learning algorithm that, in contrast to other feature learning techniques, is not just derived from intuition or biological analogy, but has a theoretical justification in the framework of statistical learning theory. Our experiments show that the new training procedure leads to significant improvements in classification accuracy while preserving the modularity and geometric interpretability of a support vector machine setup.

1. Introduction

Object categorization is a core topic of computer vision research, and few other areas have seen as fast progress over the last decade. With the development of patch-based image representations, such as *SIFT* [26], *bag-of-visual words quantization* [12], and *spatial pyramid coding* [21] for the first time global image representations were available that allow reliable decision about local properties of an image, for example if a certain object class is visible or not. Later it was observed that soft and data-dependent encodings, such as *locality constrained linear coding* [39], *super vector encoding* [42], or *Fisher vectors* [30] can improve the categorization accuracy even further. In combination with linear support vector machine classifiers, such mid-level feature representations have become a *de facto* standard for large-scale visual categorization.

stage	operation	type
SVM prediction	sign $f(X)$	non-linear
	$f(X) = \langle w, \phi(X) \rangle$	linear
per image vector, $\psi(X)$	square root, normalize (3)	non-linear
	compute average of $\psi(x_i)$	linear
per descriptor vector, $\psi(x_i)$	multiply by γ_k in (1)/(2)	non-linear
	bracket $\big(\cdot \big)$ in (1)/(2)	linear
preprocessing	L^2-normalization	non-linear
	PCA projection	linear
SIFT	local pooling	non-linear
	gradient filter	linear
image (as multiple overlapping regions)		

Table 1. Schematic description of a Fisher kernel SVM as a 5-layer feed-forward architecture (from bottom to top).

In a parallel development, the interest in *deep learning* methods [3, 6] has increased continuously in the computer vision community over the last years. While conceptually going back at least to the 1980s, these techniques are now rediscovered by the computer vision community since only now it has become possible to build and train deep architectures that are competitive on a variety of visual categorization tasks. This trend culminated in a convolutional neural network winning the 2012 ImageNet Large Scale Visual Recognition Challenge [18], which in the years before had been dominated by hand-crafted system with mid-level features.

In this work, we relate both architectures and show that their differences are not so much structural, but rather in the interpretation which of their parts are fixed and which

* V. Sydorov and M. Sakurada contributed to equal amounts.

are trainable. Our main technical contribution is a training algorithm for support vector machines with Fisher kernels that jointly learns the classifier weight vector and a suitable image representation. The procedure is not only intuitively appealing, but also has a theoretical justification in statistical learning theory.

After training, the architecture is still an instance of a support vector machine with Fisher kernel. This allows us to transfer the learned representation effortlessly to other categorization tasks. In a quantitative evaluation we show that learning the representation indeed leads to improved classification accuracy compared to Fisher kernels with fixed parameters.

2. Deep Fisher Kernel Learning

In this section we first give short summaries of image categorization with Fisher kernel SVMs and with deep architectures, concentrating on convolutional neural networks (CNNs). We then highlight the conceptual similarities between both architectures, showing that the idea of end-to-end training and discriminatively learned feature representations is not limited to neural network architectures, but can also be applied for SVMs with Fisher kernels.

2.1. Image categorization with Fisher kernel SVMs

Fisher kernels were first introduced as a mathematically sound tool for combining generative probabilistic models with discriminative kernel methods [17]. In this work, we concentrate on their practical use for image categorization, following the established setup introduced in [30, 31].

Our base representation of an image is as a set of local descriptors, for example SIFT or one of its variants. The descriptors are PCA-projected to reduce their dimensionality and decorrelate their coefficients. We assume a given K-component Gaussian mixture model (GMM) in descriptor space, $p(x|\pi,\mu,\Sigma) = \sum_{k=1}^{K} \pi^k g_k(x;\mu^k;\Sigma^k)$, where $\pi \in [0,1]^K$ are mixture weights, and each $g_k(x;\mu^k;\Sigma^k)$ is a Gaussian with mean $\mu^k \in \mathbb{R}^D$ and each diagonal covariance matrix, $\Sigma^k = \operatorname{diag}(\sigma^k)$ for $\sigma^k \in \mathbb{R}^D$. For any descriptor, x, we define a vector, $\psi(x) = \big(\mathscr{F}^1(x),\dots,\mathscr{F}^K(x),\mathscr{G}^1(x),\dots,\mathscr{G}^K(x)\big) \in \mathbb{R}^{2KD}$. The subvectors

$$\mathscr{F}^k(x) = \frac{1}{\sqrt{\pi^k}}\gamma_k(x)\left(\frac{x-\mu^k}{\sigma^k}\right) \quad \in \mathbb{R}^D \tag{1}$$

$$\mathscr{G}^k(x) = \frac{1}{\sqrt{2\pi^k}}\gamma_k(x)\left(\frac{(x-\mu^k)^2}{(\sigma^k)^2}-1\right) \quad \in \mathbb{R}^D \tag{2}$$

are the gradients of $p(x|\pi,\mu,\Sigma)$ with respect to the parameter vectors μ^k and σ^k, respectively, scaled by an empirical estimate of the inverse Fisher information matrix (see [30]). Division and multiplication of vectors should be understood componentwise, and $\gamma_k(x)$ denotes the posterior of the k-th GMM component, $\gamma_k(x) = \frac{\pi^k g_k(x;\mu^k;\Sigma^k)}{\sum_{j=1}^{K}\pi^j g_j(x;\mu^j;\Sigma^j)}$.

To represent an image, $X = \{x_1,\dots,x_M\}$, one averages the vector representations of all descriptors, $\psi(X) = \frac{1}{M}\sum_{i=1}^{M}\psi(x_i)$. The result is called the *Fisher vector* of the image, since for any two images X and X', the inner product $\psi(X)^\top\psi(X')$ is approximately equal to the Fisher kernel, $k(X,X')$, that is induced by the GMM.

In practice, it has been proven useful to postprocess the Fisher vector further: we compute the signed squared of each vector dimension, $d = 1,\dots,2KD$, and normalize such that the resulting vector has unit L^2-norm [31]

$$\phi_d(X) = (\operatorname{sign}\psi_d(X))\sqrt{|\psi_d(X)|} \;/\; \sqrt{\|\psi(X)\|_{L^1}} \tag{3}$$

For simplicity we refer to the resulting vectors again as Fisher vectors and to their inner product as Fisher kernel.

The main advantage of having such an explicitly computable vector representation is that one can use an SVM in its primal (or linear) form. This allows training a state-of-the-art object categorization system for thousands of classes from millions of training images within just a few hours [1].

2.2. Image categorization with CNNs

Convolutional neural networks (CNNs) [22] are feed-forward architecture that consist of multiple interconnected layers. Each layer computes a non-linear function using the outputs of the previous layer as its inputs. For the first layer the input is the image itself. In the last layer each output is associated with one of the target classes, and its value is used as a measure of confidence whether the class is present in the image or not.

Within each layer, multiple computing elements, the *neurons*, operate in parallel. Each neuron computes one output value using the same computational rule as the other neurons in the layer, but based on a different *receptive field*, i.e. subset of the available inputs. The actual computation within each neuron has two phases: first, a linear map is applied to the inputs. For CNNs, these maps are convolutions of the inputs with (learned) filter masks. Afterwards, a non-linear transformation is applied to the result, for example a sigmoid or a thresholding [29], and often also a spatial pooling or subsampling operation is applied. Improved classification results have been reported when additionally a groupwise normalization of several neuron's outputs is performed, e.g. by dividing each neuron's output by the L^p-norm of the outputs of a neighborhood [18].

Training CNNs is usually done by classical *backpropagation* [33], which essentially is a joint stochastic gradient descent optimization of all network parameters. The main challenge lies in computing the gradient in an efficient way. For the last layer this is straight-forward, since these parameters have an immediate effect on the network's output. For parameters in deeper layers (further away from the output), analytic expression can be obtained by repeated invocations

of the chain rule. CNNs also allow for regularization, either explicitly by weight decay [27] or implicitly by early stopping [8].

2.3. Fisher kernel SVMs as deep networks

Comparing the two above procedures, we observe a substantial number of conceptual similarities. Like a CNN, an SVM with Fisher kernel makes predictions for new images by executing a sequence of alternating linear and non-linear steps that start at the raw image and end with a value that can be interpreted as confidence in the class decision. See Table 1 for an illustration. Even technical details, such as the application of a group-wise normalization step after the componentwise non-linearity, have been identified as useful in both settings.

The main difference lies in the parameterization of the computational steps. In the Fisher kernel SVM the operations in each level were designed manually and their parameters are held fixed during training, except for the final layer which implements the SVM classifier. In a CNN, all linear operations are fully parameterized, and the parameter values are learned during the training phase. As a consequence, CNNs are very flexible to adapt to different data sources, but they are also very demanding in computational resources and the amount of training data necessary to achieve good generalization.

Staying within the context of image categorization, it is debatable if all the flexibility that deep architectures offer is truly necessary. For example, it is known that when training a convolutional network on natural image data, the first stages always learn essentially the same functionality: they compute local image gradient orientations and pool them over small spatial regions [24]. This, however, is also how a SIFT descriptors is computed. The last layer of a CNN, on the other hand, can be seen as an ordinary classifier acting on features that were computed by the previous stages.

Consequently, the main difference between the two architectures lies in the intermediate layers: CNNs assume just a parametric form of filters here and learn task-dependent values for the parameters jointly with the classifier. The Fisher kernel SVM uses a set of rules that are parameterized by a fixed GMM that was constructed earlier in a generative way.

In this work we aim at bridging this gap by training Fisher kernel SVMs in a deep way: classifier parameters and GMM parameters are learned jointly from training data.

2.4. Backpropagation in Fisher kernel SVMs

We follow our observation in the previous section and interpret the Fisher kernel SVM as a deep learning architecture. The last three layers are parameterized, one by classifier weight vector, the other two by the GMM that determines the the Fisher kernel. To train it we take the binary SVM objective function with squared Hinge loss, and treat it as a function not just of the weight vector, w, but also of the GMM, $G = (\pi, \mu, \sigma)$,

$$\mathcal{L}(w, G) = \frac{1}{2}\|w\|^2 + \frac{C}{n}\sum_{i=1}^{n} \ell(y_i \langle w, \phi_i \rangle)^2, \qquad (4)$$

where $\ell(t) = \max\{0, 1 - t\}$ is the hinge loss, and the right hand side implicitly depends on G through the procedure of computing $\phi_i = \phi(X_i)$ from the training images X_i.

On first sight, it seems appealing to now minimize $\mathcal{L}$ with respect to both, w and G. However, there is no *a priori* reason why this would be a good idea, i.e. why it would lead to a better classifier than a minimization only with respect to w. It is, for example, imaginable that minimizing over G only leads to overfitting. It is here that we benefit from the fact that the objective (4) corresponds not an arbitrary deep network, but an SVM with Fisher kernel. Returning to the viewpoint of support vector machines as maximum margin classifiers, we use the following theorem from statistical learning theory to obtain a guarantee on the performance of the learned classifier.

Theorem 1 (SVM Radius–Margin Bound, simplified in notation from [11, Theorem 4.22])**.** *Let* $\{(x_1, y_1), \ldots, (x_n, y_n)\} \subset \mathbb{R}^d \times \{\pm 1\}$ *be a set of i.i.d. training samples from an unknown data distribution* $p(x, y)$*. Then there exists a constant* c *such that for all linear classifiers,* $f(x) = \operatorname{sign}(\langle w, x \rangle)$*, the following inequality holds with high probability:*

$$\Pr_{(x,y)\sim p}\{f(x) \neq y\} \leq R^2\|w\|^2 + \frac{c}{n}\sum_{i=1}^{n} \ell(y_i \langle w, x_i \rangle)^2, \qquad (5)$$

where $R \in \mathbb{R}$ *is the radius of the smallest ball centered at the origin that contains all data points.*

For the technical definition of *"with high probability"* and a proof of the theorem, please see the original reference.

The theorem states that a classifier has a low probability of making mistakes on future data, if *1) it has a small loss on the training set*, and *2) its weight vector has a small norm in relation to the data radius*. In ordinary SVM learning the data representation is fixed, so the data radius is constant and it suffices to minimize the norm of w. When the data representation is allowed to change, however, it is crucial not to forget about the influence of the data radius, otherwise the generalization guarantees of Theorem 1 are lost.

In the case of deep learning for Fisher kernel SVMs, we can use Theorem 1 to show that minimizing the objective (4) also with respect to G is theoretically justified: we first observe that for any GMM G, the set of possible Fisher vectors lies within a unit ball around the origin, since the normalization condition (3) ensures $\|\phi\|_{L^2} = 1$. Consequently, the bound (5) holds with $R = 1$ for any GMM,

Algorithm 1 Deep Fisher learning

input training images $X_1, \ldots, X_n$, labels $y_1, \ldots, y_n$.
input initial GMM, $G = (\log \pi, \mu, \log \Sigma)$
input regularization parameter C

1: **repeat**
2: compute Fisher vectors with respect to G:
$\phi_i^G = \phi(x_i; G)$, for $i = 1, \ldots, n$
3: solve SVM for training set $\{(\phi_i^G, y_i)_{i=1,\ldots,n}\}$
$w \leftarrow \operatorname{argmin}_w \frac{1}{2}\|w\|^2 + \frac{C}{n}\ell(w; G)$
with $\ell(w; G) = \sum_{i=1}^n \max\{0, 1 - y_i\langle w, \phi_i^G\rangle\}^2$
4: compute gradients w.r.t. the GMM parameters
$\delta_{\log \pi} = \nabla_{\log \pi}\ell(\cdot), \delta_\mu = \nabla_\mu \ell(\cdot), \delta_{\log \Sigma} = \nabla_{\log \Sigma}\ell(\cdot)$
5: find best step size η^* by line search:
$\eta^* = \operatorname{argmin}_\eta \ \ell(w; G_\eta)$
with $G_\eta = (\log \pi - \eta\delta_{\log \pi},\ \mu - \eta\delta_\mu,\ \log \Sigma - \eta\delta_{\log \Sigma})$
6: update GMM parameters, $G \leftarrow G_{\eta^*}$
7: **until** stopping criterion fulfilled

output GMM G, classifier $f(x) = \operatorname{sign}\langle w, \phi(x; G)\rangle$.

which turns the right hand into the SVM objective (up to constants). The smaller its value, the fewer mistakes we can expect on future data. Therefore, finding a classifier by minimizing Equation (4) with respect to w as well as G is a promising way to find a classifier of high accuracy.

2.5. Algorithm

In this section we introduce our main technical contribution: a procedure for the deep training of SVMs with Fisher kernel. Algorithm 1 shows the steps in pseudocode. The main loop (lines 1–7) iteratively updates the SVM and GMM parameters until a stopping criterion is reached. This could be, e.g., that all parameters have converged, after a predetermined number of steps, or when the classification accuracy on a validation set stops to increase.

The main observation for the algorithm is that minimizing Equation (4) with respect to w for a fixed GMM is a convex optimization problem. In fact, it is a standard SVM problem, for which we can find the unique optimal solution efficiently using existing SVM solvers. We therefore treat this step as a black box subroutine (line 3).

Minimizing the objective with respect to the GMM parameters, even for fixed w, is a non-convex optimization problem that we address by gradient descent. Line 4 computes the gradient of the loss function with respect to the GMM parameters π, μ and σ. Their influence on the loss is indirect through the computed Fisher vector, so as in the case of deep networks, one must make use of the chain rule. Analytic expressions for the gradients are given in the appendix. Unfortunately, evaluating the gradients numerically is computationally expensive, since there are non-trivial couplings between all parameters. A straight-forward implementation has runtime complexity $O(d^2T)$, where d is the dimension of the Fisher vectors, and T is the combined number of SIFT descriptors in all training images. Therefore, updating the GMM can be orders of magnitude slower than just computing all Fisher Vectors, which has runtime complexity $O(dT)$. In the appendix, we also discuss how to mitigate this effect by working with more efficient approximate gradients.

For the gradient update we follow a batch setting with a line search (line 5) to find the most effective step size in each iteration. In combination, Algorithm 1 forms a block-coordinate descent. The objective values decrease monotonically until the algorithm terminates in a local optimum. For any fixed GMM, the SVM parameters are even globally optimal. Therefore, if we stop the algorithm early, we can expect the current solution to be the best so far.

During the optimization it must be ensured that the mixture weights and the Gaussian variances remain positive. We achieve this by internally parameterizing and updating the logarithms of their values, from which the original parameters can be obtained at any time by exponentiation. Another requirement for a valid GMM parameterization is that the mixture weights sum up to 1. Enforcing this constraint naively would require a projection step after every update. We avoid this by deriving the gradients even for unnormalized coefficient, $\tilde{\pi}_k$, that relate to the normalized weights as $\pi_k = \tilde{\pi}_k / \sum_j \tilde{\pi}_j$. We then renormalize the mixture weights when updating the actual GMM parameters (line 6). This is not strictly necessary for the algorithm, but it simplifies the analytic expressions and prevents numerical instabilities.

As mentioned above, the optimization with respect to G is non-convex. Algorithm 1 will find only a local optimum of the objective function, and its quality will depend on the initialization. For deep Fisher learning, we are in the lucky situation that a strong initialization is readily available: we simply use a GMM obtained by unsupervised *expectation maximization*, as typically used for computing Fisher kernels anyway. This can be seen as another overlap with deep learning, where layer-wise unsupervised pre-training is a common technique for finding good initializations [16].

Algorithm 1 has two relevant outputs: a classifier that has been trained specifically for the problem at hand, and a Gaussian mixture model that was trained such that the Fisher kernel it induces works well with an SVM classifier. It seems likely that such a kernel would be useful also in a standard Fisher kernel setup with no further deep training. In Section 4 we test this experimentally.

2.6. Extensions

The above section describes the most simple setup of a single binary classification task. Extensions to multi-

class classification, regression, or structured prediction, are straight-forward, since all of these can be formulated with linearly parameterized maximum-margin objectives [19].

Another promising direction is *multi-task* learning: given L learning tasks with objective functions $\mathcal{L}_1, \ldots, \mathcal{L}_L$, we form a new, joint objective function for all tasks,

$$\mathcal{L}_{MT}(w_1, \ldots, w_L, G) = \sum_{l=1}^{L} \mathcal{L}(w_l, G). \qquad (6)$$

Note that each task has its own weight vector, w_l, but all tasks share the GMM, G, so they rely on the same Fisher kernel. We can minimize Equation (6) by a variant of Algorithm 1: for fixed G, all w_l can be updated in parallel, since the SVM are decoupled. In the update of G, the gradient of (6) is just the sum of the gradients of the individual task.

Note that for ordinary SVMs with fixed kernel, multi-task learning by combining the objectives additively as in Equation (6) is pointless. The resulting optimization problem would be completely decoupled with respect to the unknown $w_1, \ldots, w_L$, so the solutions from joint learning are the same as when learning each task separately. In the deep Fisher learning, however, information flows between tasks through the shared G. The main advantage of multi-task training in this form, however, is not increased accuracy compared to learning separate GMMs (that might happen or not), but the fact that the information from all tasks is combined into a single learned kernel function. It is easier to transfer this kernel to new problems than the L kernels that are learned when training L task separately.

3. Related Work

In this section we discuss relevant prior work with focus on methods that also aim at learning kernels or representations for SVM classifiers. For a broader overview, see, e.g., these overviews on object categorization [32], image kernels [19], and deep representation learning [4, 23].

While SVM classifiers traditionally assume a fixed kernel on top of a fixed data representation, many methods have been proposed in the context of computer vision to learn the data representation automatically. Recent examples include the learning of dictionaries for bag-of-words representations [14, 28], compact binary codes [5, 41], or reuse the outputs of other discriminatively trained classifiers [15, 20, 25]. It has also been proposed to learn optimal image descriptors [40], or optimal strategies for feature quantization and spatial pooling [7]. The resulting methods typically consist of two-layered architectures that are trained either stage-wise (first the representation, then the classifier) or by alternating between both stages.

An alternative path is to aim at learning a task-specific kernel function instead of changing the data representation. This was studied systematically in [9], which introduces a framework for simultaneously learning the kernel parameters and the classifier by minimizing a joint objective function. *Multiple kernel learning* [2, 38] is a special case of this idea which restrict the new kernel to a weighted linear combination of a set of base kernels. Later extensions allow also for more complex dependencies [14, 37].

Deep Fisher kernel learning stands in the tradition of both of the above categories. In a kernel view it is a particular instance of the general kernel learning framework [37], but with the advantage that the layer-wise and feed-forward setup allows for analytic expressions of all gradients. Compared to the view of representation learning, which are often ad-hoc constructions, it has the benefit that the minimization of the objective function can be justified by a generalization bound.

The main advantage of deep Fisher learning over earlier work, however, is that by working with Fisher kernels we start from a particularly strong baseline: even without learning the GMM, Fisher kernels provide state-of-the-art results in image categorization tasks. As our experiments in Section 4 will show, learning the GMM parameters leads to substantial further improvements, even when using only moderate amounts of training data. This is in contrast to feature learning methods that use simple architectures. These allow efficient training, but result in representations that typically do not improve significantly over state-of-the-art hand designed representations. It is also different from the situation for very complex models, including general deep belief networks, which need large amounts of training data to find good representations and classifiers.

We see the deep learning of Fisher kernels as a compromise, located in a sweet spot between the two extremes. We build on a model that is known to be powerful, but by using prior knowledge about images, such as SIFT descriptors and the established non-linearities of Fisher kernels, we keep the number of parameters reasonable, so we are able to train successfully even on moderately sized datasets.

Alternative approaches for improving Fisher kernel classifiers have been proposed. In [36] a replacement for the Fisher information matrix is learned for Fisher kernels induces by a Markov model (HMM) or a Markov random field (MRF). In [35] the Fisher vector construction is applied recursively, thereby creating a deeper pipeline than the five steps we use. Both approaches are orthogonal to ours and it will be interesting to see if even better results can be achieved by a combined setup.

4. Experiments

We perform experiments on the PASCAL VOC2007 dataset [13]. With approximately 2500 training, 2500 validation and 5000 test images of 20 classes it is of medium size and allows us to study two questions in detail: *1) how does deep learning of the Fisher kernel affect the classifica-*

(a) deep training

class	base	deep	diff.
aeroplane	67.3	**70.5**	+3.3
bicycle	52.8	**56.9**	+4.1
bird	46.6	**50.6**	+4.1
boat	59.6	**59.9**	+0.3
bottle	26.1	**26.6**	+0.5
bus	52.6	**55.1**	+2.5
car	75.1	**77.7**	+2.6
cat	48.0	**51.5**	+3.5
chair	50.3	**51.5**	+1.2
cow	27.0	**30.4**	+3.4
diningtable	37.4	**44.5**	+7.1
dog	39.9	**40.7**	+0.8
horse	66.6	**68.5**	+1.9
motorbike	64.3	**65.3**	+1.0
person	78.1	**81.0**	+2.9
pottedplant	19.4	**22.0**	+2.6
sheep	26.6	**29.6**	+3.0
sofa	47.2	**49.3**	+2.1
train	70.9	**72.4**	+1.6
tvmonitor	47.0	**50.2**	+3.1
average	50.1	**52.7**	+2.6

(b) kernel transfer

k_{base}	k_{deep}	diff.
74.2	**77.6**	+3.4
56.6	**63.0**	+6.4
49.8	**53.9**	+4.2
61.7	**62.5**	+0.8
28.4	27.6	–0.8
57.3	**60.2**	+2.8
76.3	**79.7**	+3.4
55.1	**56.8**	+1.6
48.8	**49.7**	+0.9
40.9	**44.2**	+3.3
45.5	**47.7**	+2.2
39.4	**42.8**	+3.3
73.8	**76.8**	+3.0
65.9	**69.5**	+3.7
82.1	**84.4**	+2.2
22.5	**23.7**	+1.3
28.0	**29.9**	+1.9
44.5	**46.2**	+1.7
75.9	**78.8**	+2.9
48.2	**52.0**	+3.9
53.7	**56.3**	+2.6

Table 2. Results of deep Fisher training (average precision on PASCAL VOC2007 in %). (a): Learning task-specific kernels improves the results for all classes, often by a large margin. (b): The learned kernels also lead to significant improvements when used in an ordinary SVM setup without further deep learning.

tion accuracy, and *2) how do the learned kernels perform in a traditional SVM scenario*?

4.1. Image features

We use feature representations as they are currently state-of-the-art for SVM-based object categorization, following the descriptions in [1, 10]. Each image is represented by a set of approximately 10,000 local image descriptors of 64 dimensions. To obtain these we first convert color images to gray scale, perform contrast normalization, and scale each image isotropically to at most 100,000 pixels. We then extract 128-dimensional SIFT descriptors of 5 scales from a dense grid with 4×4 pixel spacing. The descriptors are preprocessed by taking the square root of every entry, followed by a projection to 64 dimension using a matrix obtained previously by PCA on a separate dataset. From the local descriptors we form a 4096-dimensional Fisher vector for each image based on a 32-component GMM that was either obtained prior to learning by expectation maximization, or that is learned by Algorithm 1.

Even though the features are rather low-dimensional and based on only a single descriptor type, they provide a powerful representation for image categorization. As our later result show, linear SVMs with Fisher vectors constructed in the above way achieve classification accuracies on par with the state-of-the-art for features of this dimension [10]. With a mean average precision score of 53.7, already the baseline system would have ranked 4th out of 17 participating systems in the original PASCAL VOC challenge, despite the fact that most participants used multiple descriptor types, and higher-dimensional features or non-linear classifiers. While higher numbers than ours have been reported in the literature, these are typically the result of higher-dimensional feature vectors or of combining multiple feature type, such as SIFT and color histograms. A detailed study of the influence of such factors can be found in [34].

4.2. Results

In a first set of experiments we directly compare the classification accuracy of the baseline setup and of the deep training. Table 2(a) contains the results when using Algorithm 1 for 100 iterations on the *train* part of the PASCAL VOC dataset, and evaluating the resulting classifier on the *val* part. One sees that deep training improves the classification quality for all classes, and often by a large margin. A more detailed discussion and learning curves can be found in the supplemental material.

In a second set of experiments we evaluate the possibility of reusing the learned data representation for future tasks. For this we train ordinary Fisher kernel SVMs on the *trainval* part of the data and evaluate them on the *test* part. As baseline, we use the Fisher kernel computed with respect to the default GMM. For the learned model, we select for each class the GMM with best results on the validation set in the previous experiments. Table 2(b) shows the results. Again, the learned Fisher kernels are clearly superior to the base kernel, improving the classification quality in all cases but one. The overall improvement is as big as in the previous setting, indicating that the positive effect of learning the representation is orthogonal to the positive effect of a larger available training set.

We also performed an experimental evaluation of the multi-task learning introduced in Section 2.6. For this we repeat the experiments of the previous paragraph, but now based on Equation (6), such that the classifiers of all 20 classes share a common kernel. The results in Table 3(a) show that this procedure also clearly improves the classification accuracy, with average precision scores typically between the baseline and the task-specific training. Table 3(b) reports the results of using the resulting (single) kernel in a regular SVM task. Again, the improvement is substantial but lower than when training a separate kernel for each task. We interpret the above result as an indication that even the approximately 2500 training image of the PASCAL VOC training set are already sufficient to train strong per-task models, so the additional regularization induced by shar-

(a) multi-task deep training

class	base	deep	diff.
aeroplane	66.8	**68.7**	+1.9
bicycle	53.0	**55.4**	+2.4
bird	46.5	**50.5**	+4.0
boat	59.4	**59.5**	+0.1
bottle	25.7	**26.0**	+0.3
bus	52.3	**52.6**	+0.3
car	74.6	**75.6**	+1.0
cat	49.1	**52.0**	+3.0
chair	45.2	**47.0**	+1.8
cow	30.2	**33.7**	+3.5
diningtable	**39.8**	39.0	–0.7
dog	37.5	**39.7**	+2.2
horse	68.1	**69.5**	+1.4
motorbike	64.3	**65.9**	+1.6
person	78.6	**80.4**	+1.8
pottedplant	17.5	**18.8**	+1.4
sheep	26.2	**27.9**	+1.7
sofa	44.3	44.3	0.0
train	71.2	**73.4**	+2.3
tvmonitor	**48.2**	48.1	–0.1
average	49.9	**51.4**	+1.5

(b) kernel transfer

k_{base}	k_{deep}	diff.
74.2	**75.7**	+1.5
56.6	**58.9**	+2.3
49.8	**52.8**	+3.1
61.7	61.2	–0.5
28.4	28.4	0.0
57.3	**58.6**	+1.3
76.3	**77.5**	+1.2
55.1	**55.7**	+0.6
48.8	**49.5**	+0.7
40.9	**43.4**	+2.5
45.5	**46.4**	+0.9
39.4	**40.3**	+0.9
73.8	**74.7**	+0.9
65.9	**67.7**	+1.9
82.1	**83.5**	+1.4
22.5	**22.9**	+0.4
28.0	**32.4**	+4.4
44.5	**45.2**	+0.7
75.9	**76.7**	+0.7
48.2	**48.6**	+0.4
53.7	**55.0**	+1.3

Table 3. Results of multi-task deep Fisher training (average precision on PASCAL VOC2007 in %). (a) Multi-task learning of a shared kernel improves the results for almost all classes. (b) The learned kernels also lead to significant improvements when used in an ordinary SVM setup without further deep learning.

ing the GMM is not required here. However, part of the effect might also be explained by the less flexible model selection step. In the multi-task setting, Equation (6), all tasks share the same regularization constant, whereas in the single-task setting, Equation (4), the regularization strength is determined on a per-task basis.

5. Summary and Discussion

We made two main contributions in this work. The first is conceptual: SVMs with Fisher kernel for image categorization can be interpreted as deep networks, and this view opens possibilities for transferring successful concepts from deep learning to maximum margin learning. The second contribution is algorithmic and demonstrates a practical realization of such transfer: a deep training algorithm that learns an image representation by Fisher vectors together with parameters of an SVM, while staying mathematically grounded in statistical learning theory. We also studied a method for multi-task SVM learning in image categorization, where classifiers act independently, but share their underlying feature representation.

Our experiments show substantial improvements by learning the Fisher kernel, compared to a baseline that itself already provides results comparable to the state-of-the-art. We believe that our observations will be just first steps in a process that will ultimately lead to new hybrid learning models that combine the expressive power of deep architectures with the theoretical guarantees and geometric interpretability of maximum margin methods.

6. Appendix

This appendix gives explicit expression for the gradients of the loss function ℓ with respect to the GMM component. To shorten the notation we drop the explicit dependence on the weight vector, w, the input, x, and the current value of the GMM, $G = (\pi, \mu, \Sigma)$, where it is clear from the context.

The most complex expressions are the derivatives of the Fisher vectors (1) and (2) with respect to the GMM parameters. An elementary but lengthy calculation yields for all GMM component indices, $k, k' = 1, \dots, K$ and vector components indices, $d, d' = 1, \dots, D$:

$$\frac{\partial}{\partial \pi^k} \mathscr{F}_{d'}^{k'} = \frac{\gamma_{k'} \alpha_{d'}^{k'}}{2\pi^k \sqrt{\pi^{k'}}} (\pi^k + \delta_{kk'} - 2\gamma_k) \tag{7}$$

$$\frac{\partial}{\partial \mu_d^k} \mathscr{F}_{d'}^{k'} = \frac{\gamma_{k'}}{\sigma_d^k \sqrt{\pi^{k'}}} \left(\alpha_{d'}^{k'} \alpha_d^k (\delta_{kk'} - \gamma_k) - \delta_{dd'}^{kk'} \right) \tag{8}$$

$$\frac{\partial}{\partial \sigma_d^k} \mathscr{F}_{d'}^{k'} = \frac{\gamma_{k'} \alpha_{d'}^{k'}}{\sigma_d^k \sqrt{\pi^{k'}}} \left(\left((\alpha_d^k)^2 - 1 \right) (\delta_{kk'} - \gamma_k) - \delta_{dd'}^{kk'} \right) \tag{9}$$

$$\frac{\partial}{\partial \pi^k} \mathscr{G}_{d'}^{k'} = \frac{\gamma_{k'} ((\alpha_{d'}^{k'})^2 - 1)}{2\pi^k \sqrt{2\pi^{k'}}} [\pi^k + \delta_{kk'} - 2\gamma_k] \tag{10}$$

$$\frac{\partial}{\partial \mu_d^k} \mathscr{G}_{d'}^{k'} = \frac{\gamma_{k'} \alpha_d^k}{\sigma_d^k \sqrt{2\pi^{k'}}} \left[\left((\alpha_{d'}^{k'})^2 - 1 \right) (\delta_{kk'} - \gamma_k) - 2\delta_{dd'}^{kk'} \right] \tag{11}$$

$$\frac{\partial}{\partial \sigma_d^k} \mathscr{G}_{d'}^{k'} = \frac{\gamma_{k'}}{\sigma_d^k \sqrt{2\pi^{k'}}} \times \left[((\alpha_{d'}^{k'})^2 - 1)((\alpha_d^k)^2 - 1)(\delta_{kk'} - \gamma_k) - 2\delta_{kk'}^{dd'} (\alpha_d^k)^2 \right] \tag{12}$$

where $\alpha^k := \frac{x - \mu^k}{\sigma^k}$, and $\delta_{ab} = 1$ if $a = b$, and 0 otherwise, and $\delta_{cd}^{ab} = \delta_{ab} \delta_{cd}$.

Stacking the above expression and averaging them over all descriptors in an image, we obtain $\nabla \psi$, the gradient of the unnormalized per-image Fisher vector. From this the gradient of the normalized Fisher vector (Equation (3)) is obtained by an invocation of the chain rule:

$$\nabla \phi_d = \left(\frac{\nabla \psi_d}{2\psi_d} - \frac{\sum_{d'} \operatorname{sign}(\psi_{d'}) \nabla \psi_{d'}}{2 \|\psi\|_{L^1}} \right) \phi_d, \tag{13}$$

where the gradient acts with respect to all parameters, (π, μ, Σ). The gradient of the loss term (Equation (4)) follows by applying the chain rule one more time,

$$\nabla \ell(w, G) = -2 \left\langle w, \sum\nolimits_{i=1}^{n} a_i y_i \nabla \phi(X_i) \right\rangle, \tag{14}$$

where $a_i = \max\{0, 1 - y\langle w, \phi(X_i)\rangle\}$ for any image X_i, and the inner product is taken with respect to the $2KD$ components of w. When gradients in the logarithmic domain are required, we use the identity $\frac{\partial}{\partial \log t} f(t) = t \frac{\partial}{\partial t} f(t)$.

Note that while computing the gradient with the above expressions is computationally costly, there are multiple ways to accelerate it. First, one observes that the leading constant of each expression (7)–(13) contains a term $\gamma_{k'}$. We suggest to compute the gradient terms only if this value exceeds a threshold, e.g. 10^{-5}. A significant speedup can also be obtained by subsampling the number of descriptors used from each image to form the gradient. For our experiments, we used a fraction 10% at no noticeable loss of prediction quality. In fact, the quality improve in some cases, potentially because a slightly randomized gradient helps the algorithm to escape shallow local minima.

Acknowledgement. This work was in parts funded by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement no 308036: *"Life-long learning of visual scene understanding" (L3ViSU).*

References

[1] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. Good practice in large-scale learning for image classification. *PAMI*, 2013. 2, 6

[2] F. R. Bach, G. R. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *ICML*, 2004. 5

[3] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1), 2009. 1

[4] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *PAMI*, 35, 2013. 5

[5] A. Bergamo, L. Torresani, and A. W. Fitzgibbon. PiCoDes: Learning a compact code for novel-category recognition. In *NIPS*, 2011. 5

[6] C. M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, 1995. 1

[7] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce. Learning mid-level features for recognition. In *CVPR*, 2010. 5

[8] R. Caruana, S. Lawrence, and L. Giles. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. *NIPS*, 2001. 3

[9] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine learning*, 46(1-3), 2002. 5

[10] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *BMVC*, 2011. 6

[11] N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press, 2000. 3

[12] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *ECCV Workshop on Statistical Learning in Computer Vision*, 2004. 1

[13] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The Pascal visual object classes (VOC) challenge. *IJCV*, 88(2), 2010. 5

[14] P. V. Gehler and S. Nowozin. Let the kernel figure it out; principled learning of pre-processing for kernel classifiers. In *CVPR*, 2009. 5

[15] P. V. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *ICCV*, 2009. 5

[16] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7), 2006. 4

[17] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *NIPS*, 1999. 2

[18] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 2

[19] C. H. Lampert. Kernel methods in computer vision. *Foundations and Trends in Computer Graphics and Vision*, 4(3), 2009. 5

[20] C. H. Lampert, H. Nickisch, and S. Harmeling. Attribute-based classification for zero-shot visual object categorization. *PAMI*, 2013. 5

[21] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006. 1

[22] Y. LeCun and Y. Bengio. Convolutional networks for images, speech, and time series. In *The handbook of brain theory and neural networks*, volume 3361. MIT Press, 1995. 2

[23] Y. LeCun, K. Kavukcuoglu, and C. Farabet. Convolutional networks and applications in vision. In *International Symposium on Circuits and Systems (ISCAS)*, 2010. 5

[24] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*, 2009. 3

[25] L.-J. Li, H. Su, L. Fei-Fei, and E. P. Xing. Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *NIPS*, 2010. 5

[26] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2), 2004. 1

[27] J. Moody, S. Hanson, A. Krogh, and J. A. Hertz. A simple weight decay can improve generalization. *NIPS*, 4, 1995. 3

[28] F. Moosmann, B. Triggs, and F. Jurie. Fast discriminative visual codebooks using randomized clustering forests. *NIPS*, 2007. 5

[29] V. Nair and G. E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *ICML*, 2010. 2

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

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

[32] A. Pinz. Object categorization. *Foundations and Trends in Computer Graphics and Vision*, 1(4), 2005. 5

[33] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088), 1986. 2

[34] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the Fisher vector: Theory and practice. *IJCV*, 105(3), 2013. 6

[35] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep Fisher networks for large-scale image classification. In *NIPS*, 2013. 5

[36] L. van der Maaten. Learning discriminative Fisher kernels. In *ICML*, 2011. 5

[37] M. Varma and B. R. Babu. More generality in efficient multiple kernel learning. In *ICML*, 2009. 5

[38] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *CVPR*, 2009. 5

[39] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010. 1

[40] S. Winder, G. Hua, and M. Brown. Picking the best daisy. In *CVPR*, 2009. 5

[41] L. Yang, R. Jin, R. Sukthankar, and F. Jurie. Unifying discriminative visual codebook generation with classifier training for object category recognition. In *CVPR*, 2008. 5

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