Kernel Descriptors in comparison with Hierarchical Matching Pursuit

Jan Reubold
Department of Computer Science
Technische Universitaet Darmstadt
64283 Darmstadt
`j.reubold@gmail.com`

Abstract

In this paper a state of the art sparse coding algorithm for image classification, namely Hierachical Matching Pursuit(HMP), is compared to state of the art algorithms using kernel methods(Efficient Match Kernels, Kernel Descriptors and Hierarchical Kernel Descriptors).
HMP is faster and achieves slightly better results than the other algorithms when run over several test-sets. But on the downside, there are set-ups, where the performance of HMP declines. We review and explain all four algorithms and investigate, in which set-ups HMP shouldn't be applied.

Introduction

Today there are many different problem-solving approaches for visual object recognition. It is one of the main research fields in computer vision, machine learning and robotics. One of the difficulties with this task is to make an algorithm suitable for large images with a high-dimensional space.
An insight of the last years is, that the key for a good object recognition algorihtm is learning rich features. Today there are approaches to learn these features with machine learning methods, e.g. Deep belief networks(DBNs)[14] or Convolutional neural networks(CNNs)[15].
We take a closer look at Efficient Match Kernels(EMK), Kernel Descriptors(KD), Hierarchical Kernel Descriptors(HKD) and the Hierarchical Matching Pursuit-algorithm(HMP). These algorithms have shown state of the art results in some experiments of object recognition. Three of them, namely EMK, KD and HKD, use kernel methods for the feature extraction, whereas HMP uses sparse coding.

1 Kernel Methods

Bag of words(BOW)[21] is a very popular feature for visual object recognition. It is simple, computational efficient and robust to scale and rotation variance.
A deficiency of BOW is that it represents each local feature with the closest visual word. According to Liefeng Bo et al[5] this quantization can be too coarse and can lead to a decrease in performance. BOW with a linear SVM can be viewed as spatial match kernels[5].

A formulation of BOW:
$X = \{x_1, ..., x_p\}$ is a set of local features,
$V = \{v_1, ..., v_d\}$ is the dictionary
$\mu(x) = \{\mu_1(x), ..., \mu_d(x)\}^\top$ is a binary indicator vector,
with $\mu_i(x) = 1$, if $x \in R(v_i)$, otherwise $\mu_i(x) = 0$,

where $R(v_i) = \{x : ||x - v_i|| \leq ||x - v||, \forall v \in V\}$,
$\bar{\mu}(X) = \frac{1}{|X|} \sum\limits_{x \in X} \mu(x)$

The resulting kernel function:

$$K_B(X,Y) = \bar{\mu}(X)^\top \bar{\mu}(Y) = \frac{1}{|X||Y|} \sum_{x \in X} \sum_{y \in Y} \mu(x)^\top \mu(y) = \frac{1}{|X||Y|} \sum_{x \in X} \sum_{y \in Y} \delta(x,y), \quad (1)$$

with

$$\delta(x,y) = \begin{cases} 1, & x, y \subset R(v_i), \exists i \in \{1, ..., D\} \\ 0, & otherwise \end{cases}$$

To address the problem of too coarse quantization mentioned above, continuous kernel functions can be used:

$$K_S(X,Y) = \frac{1}{|X||Y|} \sum_{x \in X} \sum_{y \in Y} k(x,y). \quad (2)$$

There are already some kernels invented, e.g. there is the sum match kernel by Haussler[13], which adds local kernels over all combinations of two different sets of features, or neighborhood kernels[20]. They integrate the spatial location into a sum match kernel.
The problem with the kernels mentioned are their significant computational costs. They need to explicitly evaluate the full kernel matrix. For this reason they are impractical for large datasets. The question we further pursue is, how to apply kernels descriptors without the need to compute the full kernel matrix.

1.1 Efficient Match Kernel

Efficient Match Kernels(EMK) map local features to a low dimensional feature space. The resulting features are then averaged to get a so called set-level feature.
EMK can be seen as a two-stages operation. The first is to map the features to a new feature space $[x \overset{\phi}{\to} \phi(x)]$, and the second is to reconstruct $\phi(x)$ with the matrix D (defined below).

To apply the first step, the match kernel of eq. (2) can be simplified as:

$$K_S(X,Y) = \bar{\phi}(X)^\top \bar{\phi}(Y), \quad (3)$$

with

$$\bar{\phi}(X) = \frac{1}{|X|} \sum_{x \in X} \phi(x).$$

Eq. (3) is called an EMK and it can be applied to all finite dimensional kernels as shown in the paper.
EMKs allow us to first compute $\bar{\phi}(X)$, as this is a finite vector, and then apply a linear classifier on the resulting set-level features.
Classification with EMK is linear in the number of images and also in the number of local features.

Now the second step has to be applied. An approximation of $\phi(x)$ is needed. So local feature maps have to be learned, whose inner products retain the values of the kernel functions as exact as possible.
Therefore the dictionary D, $D = \{\phi(z_1), ..., \phi(z_d)\} \in R^{h \times d}$, with its d feature vectors, $Z = [z_1, ..., z_D]$, which span the low dimensional space, has to be selected. But let's first start with the approximation.
$\phi(x)$ can be approximated by minimizing the problem:

$$\bar{v}_x = \min_{v_x} ||\phi(x) - Dv_x||^2, \quad (4)$$

which can be solved analytically as a convex quadratic program:

$$\bar{v}_x = (D^\top D)^{-1}(D^\top \phi(x)). \quad (5)$$

The local kernel derived from the projected vectors:

$$k_l(x,y) = \widetilde{\phi}(x)^\top \widetilde{\phi}(y) = [D\bar{v}_x]^\top [D\bar{v}_y] = k_z(x)^\top K_{ZZ}^{-1} k_z(y), \quad (6)$$

where $\widetilde{\phi}(x) = D\bar{v}_x$, which is the approximation of $\phi(x)$.
k_z is a $d \times 1$ vector with $\{k_z\}_i = k(x, z_i)$ and K_{ZZ} is a $d \times d$ matrix with $\{K_{ZZ}\}_{ij} = k(z_i, z_j)$.
So $\bar{\phi}(X)$ can be computed like this:

$$\bar{\phi}(X) = \frac{1}{|X|} D \sum_{x \in X} \overline{v}_x \ . \tag{7}$$

As mentioned before, now the dictionary D has to be selected. Kernel principal component analysis(KPCA)[22] with a pool of randomly selected local features F and the d topmost eigenvectors as basis vectors, cannot be used, since KPCA becomes too expensive for large F, which is often required for good accuracy.
Instead it is preferable to use a constrained singular value decomposition(CKSVD):

$$argmin_{V,Z} R(V,Z) = \frac{1}{F} \sum_{i=1}^{F} ||\phi(x_i) - Dv_i||^2 \tag{8}$$

Its partial derivations is:

$$\frac{\partial R(V,Z)}{\partial v_i} = 2D^\top D v_i - 2D^\top \phi(x_i)$$

with $V = [v_1, ...v_F]$.

To make the optimization function computationally efficient, the parameter space has to be reduced. V has to be eliminated from this minimization problem.
When the partial derivations are set to zero and Z is fixed, a linear system with respect to v_x can be created, with the optimal analytic solution $\overline{v}_x$. Instead of directly optimizing $R(V, Z)$, this solution can be put into the minimization problem and then an equivalent optimization problem can be solved:

$$argmin_Z R^*(Z) = -\sum_{i=1}^{F} k_z(x_i)^\top K_{ZZ}^{-1} k_z(x_i) \tag{9}$$

With any gradient descent algorithm eq. (9) can be solved and the corresponding EMK is set up.

According to Liefen Bo et al[2] a problem with EMK is that it performs best when it is used with non-linear SVMs.
It uses hand-crafted SIFT features as the basic building blocks, which can also be modified as shown in the next subsection.

1.2 Kernel Descriptors

Based on the work of EMK a general framework was proposed in NIPS 2010[2], called kernel descriptors(KDs), to extract low level features from image patches.
For visual object recognition, orientation histograms such as SIFT[18] and HOG[11] are very common and successful features. According to the paper the match kernel in eq. (10) provides a kernel view of HOG features:

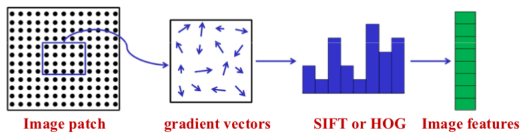

Figure 1: The process of how to obtain a sift/hog feature from an image patch

$$K_h(P,Q) = \sum_{z \in P} \sum_{z' \in Q} \widetilde{m}(z)\widetilde{m}(z')\delta(z)^\top \delta(z') \tag{10}$$

with

$$\delta_i(z) = \begin{cases} 1, & \lfloor \frac{d\theta(z)}{2\pi} \rfloor = i - 1 \\ 0, & otherwise \end{cases} \quad \text{for hard binning}$$

or

$$\delta_i(z) = \max(\cos(\theta(z) - a_i)^9, 0) \ \ \text{for soft binning}$$

where the gradient orientation is discretized into a d-dimensional vector $\delta(z) = [\delta_1(z), ..., \delta_d(z)]$ and a_i is the center of the i-th bin.

This equation only uses one pair of image patches (P and Q), but it is straightforward to do this for sets of image patches.

Now that there is a kernel view of HOG it would be desirable to modify this kernel further to not suffer from discretization errors.

So here is a modified kernel presented in the paper:

$$K_{grad}(P,Q) = \sum_{z\in P}\sum_{z'\in Q} \widetilde{m}(z)\widetilde{m}(z')k_o(\widetilde{\theta}(z),\widetilde{\theta}(z'))k_p(z,z') \tag{11}$$

with k_o a gaussian position kernel over orientations: $k_o(\widetilde{\theta}(z),\widetilde{\theta}(z')) = \exp(-\gamma_o||\widetilde{\theta}(z) - \widetilde{\theta}(z')||^2$ with $\widetilde{\theta}(z) = [\sin(\theta(z))\cos(\theta(z))]$

k_p is a gaussian kernel over positions, which measures the spatial distance between two pixels, and z denotes the 2D position, normalized to $[0,1]$, of the pixel in the image patch.

So it doesn't use hand-crafted SIFT features like EMK, but instead uses more accurate features. It designs and learns them from scratch using kernel methods.
The kernel view of orientation histograms gives a simple and unified way of turning pixel attributes into patch-level features. When applied to a bunch of different pixel attributes, like gradient, color and shape, the combination of all match kernels, which are already strong in their own ability, is much better than the best individual result[2].

To build such a low level feature extractor three steps have to be applied: (1) design a match kernel using pixel attributes, (2) learn the basis vectors using KPCA, (3) construct kernel descriptors by projecting the features vectors to a low dimensional space.
Step (2) and (3) consist of the two stages mentioned in the EMK-section with a slight modification. To find a good projection from the high dimensional, potentially infinite, feature space to the low dimensional feature space, a modified approach is used. (1) To find an approximation of the match kernels, sufficient basis vectors are uniformly and densely sampled from the support region. (2) KPCA is used to learn compact basis vectors. Unlike the approach used in the EMK-section with CKSVD no local minima are involved.

The kernels in eq. (11) can be rewritten as inner products, e.g. k_o can be rewritten as:

$$k_o(\widetilde{\theta}(z),\widetilde{\theta}(z')) = \phi_o(\widetilde{\theta}(z))^\top \phi_o(\widetilde{\theta}(z')) \tag{12}$$

So a feature over image patches can be derived:

$$F_{grad}(P) = \sum_{z\in P} \widetilde{m}(z)\phi_o(\widetilde{\theta}(z)) \otimes \phi_p(z) \tag{13}$$

$\otimes$ is the Kronecker product and $K_{grad}(P,Q) = F_{grad}(P)^\top F_{grad}(Q)$, with F_{grad} as an infinite vector, because of the use of gaussian kernels.

Unlike EMK, the basis vectors for the approximation of the kernel function are computed directly, without requiring any image.
The basis vectors are dense and uniformly sampled over the support region, using a fine grid. When sufficient basis vectors are sampled, the approximation of ϕ can be achieved by its projection into the lower dimensional space. As already shown in the EMK-section[eq. (4-7)], this procedure is equivalent to using a finite dimensional kernel. So to get the approximation of the kernels eq. (7) has to be applied:

$$\widetilde{F}_{grad}(P) = \sum_{z\in P} \widetilde{m}(z)\widetilde{\phi}_o(\widetilde{\theta}(z)) \otimes \widetilde{\phi}_p(z) \tag{14}$$

For $\widetilde{\phi}_o(\widetilde{\theta}(z))$ e.g.:

$$\widetilde{\phi}_o(\widetilde{\theta}(z)) = D\overline{v}_{\widetilde{\theta}(z)}$$

Because of the Kronecker product the dimensionality of the corresponding matrix can get too high for practical use. To make it more practical the dimension of the low-dimensional feature space has to be downsized further. Because orthogonality is not guaranteed by the sampling approach, the basis vectors have a certain amount of redundance. So KPCA can be applied over joint basis vectors: $\{\phi_o(x_1) \otimes \phi_p(y_1), ..., \phi_o(x_{d_o}) \otimes \phi_p(y_{d_p})\}$, where $\phi_p(y_s)$ are basis vectors for k_p, and d_p their quantity, with the t-th kernel principal component:

$$PC^t = \sum_{i=1}^{d_o} \sum_{j=1}^{d_p} \alpha_{ij}^t \phi_o(x_i) \otimes \phi_p(y_j) \tag{15}$$

where α_{ij}^t is learned with KPCA: $K_c\alpha^t = \lambda^t\alpha^t$, where K is a centered kernel matrix:
$[K_c]_{ijst} = k_o(x_i, x_j)k_p(y_s, y_t) - 2\sum_{i',s'} k_o(x_{i'}, x_j)k_p(y_{s'}, y_t) + \sum_{i',j',s',t'} k_o(x_{i'}, x_{j'})k_p(y_{s'}, y_{t'})$

The gradient kernel descriptor(eq. (12)) now has the form:

$$\overline{F}^t_{grad}(P) = \sum_{i=1}^{d_o} \sum_{j=1}^{d_p} \alpha_{ij}^t \left\{ \sum_{z \in P} \widetilde{m}(z) k_o(\widetilde{\theta}(z), x_i) k_p(z, y_j) \right\} \tag{16}$$

KDs perform best when used with EMK in combination with a modified version of the classical orientation histograms. On the downside, the computation takes much longer than SIFT. For a typical 300×300 image, the computation takes 4.0 seconds for the shape kernel function and 1.5 seconds for the gradient kernel function, compared to 0.4 seconds for SIFT.
And another problem is again, that it performs at its best when using in combination with non-linear SVMs.
In the following subsection we will review a paper that speeds up the whole process.

1.3 Hierarchical Kernel Descriptors

In the paper "Object Recognition with Hierarchical Kernel Descriptors"[4] Liefeng Bo et al who invented EMKs and KDs propose a modification of their previous work. The idea is to apply KDs recursively, layer by layer, to obtain high level features by aggregating lower level patch features. (fig. 1)
Hierarchical kernel descriptors(HKD) apply the KDs over the pixels of the patches in the first stage and over the resulting features in the following stages. This can be applied, until the final image-level features are obtained.
The new match kernel also uses the spatial information and the patch variations to generate the features:

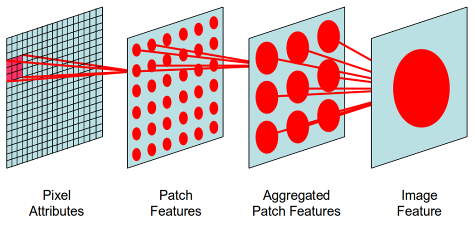

Figure 2: Layer by layer KDs are applied to obtain high level features

$$K(\overline{P}, \overline{Q}) = \sum_{A \in \overline{P}} \sum_{A' \in \overline{Q}} \widetilde{W}_A \widetilde{W}_{A'} k_F(F_A, F_{A'}) k_C(C_A, C_{A'}) \tag{17}$$

$$= \sum_{A \in \overline{P}} \sum_{A' \in \overline{Q}} \widetilde{W}_A \widetilde{W}_{A'} [\phi_F(F_A) \otimes \phi_C(C_A)]^\top [\phi_F(F_{A'}) \otimes \phi_C(C_{A'})]$$

with $k_F(F_A, F_{A'}) = \exp(-\gamma_F ||F_A - F_{A'}||^2)$, with F_A as the gradient, shape or color kernel descriptor and $k_C(C_A, C_{A'}) = \exp(-\gamma_C ||C_A - C_{A'}||^2)$, with C_A the 2D center position of a patch(normalized to $[0, 1]$) and k_C measures the spatial relationship. $\widetilde{W}_A = W_A / \sqrt{\sum_{A \in \overline{P}} W_A^2 + \epsilon_h}$, with ϵ_H a small positive constant and W_A is the average of gradient magnitudes and standard deviations for gradient and shape kernel descriptor and always 1 for the color kernel descriptor.

Again, the basis vectors for the compact low dimensional features have to be learned. The new match kernel can be rewritten(eq. (17), second line), so that the containing kernels can be represented as inner products. Now $\phi_F(F_A) \otimes \phi_C(C_A)$ has to approximated with a set of basis vectors.
ϕ_C is easy to approximate. The basis vectors can be sampled over a 5×5 grid, with $D_C = \phi_C(X_1), ..., \phi_C(X_{d_C})$ and $d_C = 25$, because C_A is only two-dimensional.
ϕ_F however cannot be sampled on a dense and uniform grid, because F_A is in high-dimensional

space. To get the basis vectors for ϕ_F the patch-level features of the training images are used in combination with K-means. The resulting centers then are the new basis vectors $D_F = \phi_F(Y_1), ..., \phi_F(Y_{d_F})$.
So the first layer of the pyramid can be computed efficiently. Now the rest of the pyramid has to be computed.
The dimensionality of the second layer has the size of the total number of the joint basis vectors of the first layer. For an accurate approximation, the number of the basis vectors of ϕ_F should be relatively high. If for example the number of basis vectors would be 5000 the size of the KD would be $25 \times 5000 = 125000$, too big for practical use.
So again, KPCA has to be applied:

$$\frac{[D_F \otimes D_C\beta]^\top [D_F \otimes D_C\beta]}{\beta^\top \beta} = \frac{\beta^\top K_F \otimes K_C\beta}{\beta^\top \beta} \tag{18}$$

with the optimal β equal to the eigenvector with the largest eigenvalue:

$$K_F \otimes K_C\beta = \lambda\beta \tag{19}$$

Due to the high dimensionality of the resulting matrix, $K_F \otimes K_C$, Liefeng Bo et al[4] have proposed a faster algorithm to find the needed eigenvectors. After that, the second layer kernel descriptors have the form:

$$\overline{F}^t(\overline{P}) = \sum_{i=1}^{d_F} \sum_{j=1}^{d_C} \beta_{ij}^t \left\{ \sum_{A \in \overline{P}} \widetilde{W}_A k_F(F_A, Y_i) k_C(C_A, X_j) \right\} \tag{20}$$

But again, HKD performs best with a combination of all three match kernels.
In the next section we will look at the difficulty from an other point of view. Instead of kernel methods an approach using sparse coding will be presented.

2 Hierarchical Matching Pursuit

Hierarchical Matching Pursuit(HMP)[3] is a sparse coding algorithm which learns the features and therefore the filters/kernels from scratch. HMP is scalable, so it can also handle full-size images.

HMP first learns a dictionary from a large set of image patches from the training images. After the dictionary is learned, it applies the matching pursuit encorder layer by layer, to extract the sparse codes and so obtain a feature hierarchy. To obtain the resulting patch-level features, spatial pyramid max pooling is used.
In the paper[3] they preferred using a two-layer HMP. When used with a single layer the performance was a good deal worse and when using a three-layer HMP, the increase in performance was negligible.
Thereafter a linear SVM can be used, which, according to Liefeng Bo et al[3], matches the performace of non-linear SVMs.

For learning the dictionary K-SVD[1] is used. The before mentioned matching pursuit encoder, which is recursively run, consists of 3 modules: (1) Batch (tree) orthogonal matching pursuit, (2) spatial pyramid max pooling and (3) contrast normalization.
According to the paper each of these steps increases the results significantly. All three will be covered in the following subsections.

2.1 Dictionary Learning with K-SVD

K-SVD is used to obtain a dictionary $D = [d_1, ..., d_m] \in R^{h \times m}$ from a large training-set of image patches, $Y = [y_1, ...y_n] \in R^{h \times n}$, obtained from the training images. It generalizes the idea of K-Means. With $K = 1$, K-SVD exactly reproduces the K-Means algorithm.

To obtain D, the following reconstruction error has to be minimized:

$$\min_{D,X} ||Y - DX||_F^2 \quad s.t. \, \forall i, \; ||x_i||_0 \leq K \tag{21}$$

where $X = [x_1, ..., x_n] \in R^{m \times n}$ is the sparse code matrix.
This is solved in an alternating manner. First D is fixed and the sparse code X is modified:

$$\min_{x_i} ||y_i - Dx_i||^2 \quad s.t. \, ||x_i||_0 \leq K \tag{22}$$

This subproblem has to be solved for all $x \in X$. It is combinational and highly non-convex, so for practical use it is only approximated. The orthogonal matching pursuit is used to obtain the approximation, which is described in the next subsection.

In the second stage, each filter and its associated sparse codes are updated simultaneously by SVD:

$$||Y - DX||_F^2 = ||Y - \sum_{j \neq k} d_j x_j^\top - d_k x_k^\top||_F^2 = ||E_k - d_k x_k^\top||_F^2 \tag{23}$$

where x_j are the columns of X and E_k is the residual matrix for the k-th filter.
Now SVD is applied to the residual matrix E_k to obtain the optimal sparse codes and dictionary entries for the k-th filter. This process only uses observations whose sparse code entries have a non-zero value at the k-th entry, to ensure that no new non-zero values are added to the sparse code matrix X.

2.2 Batch (Tree) Orthorgonal Matching Pursuit

Batch tree orthogonal matching pursuit(BTOMP) approximates the solution of eq. (22). It is used for obtaining the dictionary and later to get the sparse codes for the application, e.g. for image classification. It finds the K filters for the sparse codes in a greedy style. At each step it selects the filter with the highest correlations to the latest residual r. At the start r is equal to the observation, $r^0 = y$.
When used to obtain the sparse codes for the application the same dictionary is used for all patches. For that reason, some quantities can be pre-computed to speed up the whole process.
The clue is that x and r aren't explicitly needed for the filter selection:

$$\alpha = D^\top r = D^\top (y - D_I(D_I^\top D_I)^{-1} D_I^\top y) = \alpha^0 - G_I G_{II}^{-1} \alpha_I^0 \tag{24}$$

with $\alpha^0 = D^\top r^0$, where G is $D^\top D$. G_{II} is a sub-matrix, which contains the rows and columns indexed by I.

So α^0 and G can be pre-computed to reduce the costs. But because the computation of G becomes infeasible for a large dictionary, the algorithm puts the dictionary into a tree-structure using K-Means.
K-Means is run over the dictionary to group the filters into o sub-dictionaries $\{D_{g_1}, ..., D_{g_o}\}$ with its centers $C = [c_1, ..., c_o]$.

Algorithm 1 BTOMP

```
Require: $D, C, y, K$
Require: $B = C^\top D, x = 0, I = \emptyset$
Ensure: $x, \quad s.t.\ y \approx Dx$
1: for $k = 1 : K$ do
2:     $g_j : j = argmax_k |\alpha_k|$
3:     $\overline{k} = argmax_{k \in g_j} |d_k^\top r|$
4:     $I = I \cup \overline{k}$
5:     $x_I = (D_I^\top D_I)^{-1} D_I^\top y$
6:     $\alpha = \alpha^0 - B_I x_I$
7:     $r = y - D_I x_I$
8: end for
```

Line 2: the center filter is selected, that best matches the current residual
Line 3: within the sub-dictionary associated to the selected center, the best matched filter is selected
Line 4: the filter is added to the set of already selected filters
Line 5: the sparse codes are updated with incremental Cholesky decomposition
Line 6: the correlation between each center and the current residual ist computed
Line 7: computes the new residual
Line 1: this whole process is repeated K times, until all K filters for the sparse code x are selected

With BTOMP the feature extraction at runtime is significantly decreased[3]

2.3 Spatial Pyramid Max Pooling

Spatial pyramid max pooling is a highly non-linear operator, that extracts the patch-level features of an image, by aggregating the sparse codes, which are spatially close.
Therefore, for every level of the pyramid, it computes the maximum value of each sparse code in

the image cell and concatenates them [eq. (25)].
The first level of the pyramid consists of all sparse codes of the patch, for the second level the patch is subdivided into four quadrants and so on.

$$\widetilde{F}(P) = \left[\max_{j\in P}|x_{j1}|, ..., \max_{j\in P}|x_{jm}|\right] \tag{25}$$

$$F(P) = \left[\widetilde{F}(P_1^1), ..., \widetilde{F}(P_1^{V_1}), ..., \widetilde{F}(P_U^{V_U})\right] \tag{26}$$

where P is an image cell, U is the number of pyramid levels and V_i is the number of cells in the i-th pyramid level.

The resulting patch-level feature is a concatenation of the different spatial cells of the pyramid [eq. (26)].

2.4 Contrast Normalization

Because of local changes in illumination and other effects like that, contrast normalization seems like a good choice to improve the results.

$$\overline{F}(P) = \frac{F(P)}{\sqrt{||F(P)||^2 + \epsilon}} \tag{27}$$

with ϵ a small positive number.

In the paper Liefeng Bo et al have run tests with and without contrast normalization. They showed that the results with contrast normalization applied were about 3% better than the results without. They tested further if the L_1- or the L_2-norm would be better for the normalization and found out that the L_2-norm was consistently better.

3 Conclusion

GS-MKL[24]	LSR[23]	RPP[17]	BM-SVM[7]	SSC[19]
84.3	81.9	81.9	81.3	80.02 ±0.36
NBNN[6]	FC[12]	**HMP**	**KD**	**EMK**
79.23	77.8 ±0.4	76.8 ±0.4	76.4 ±0.4	70.3 ±0.8

Table 1: Caltech 101 results

All of the reviewed algorithms (EMK, KD, HKD and HMP) deliver good performances when run over test-sets(Caltech 101, CIFAR-10, etc). HMP performs best, followed by HKD and KD. EMK trails a bit behind(table 1 and 2).
When looking at the top of the class algorithms from the test-sets of CIFAR-10 and Caltech 101, our reviewed algorithms lack of performance. But when we put a finer point on it, it is made aware that most of the other algorithms have a considerable runtime deficiency, e.g. the MCDNN-algorithm relies on deep convolutional neural networks (DNN).

MCDNN[8]	RFL[16]	SRFDN[10]	HPNN[9]	**HKD**	**KDES-A**
88.79	83.11	82	80.49	80	76

Table 2: CIFAR-10 results

The main difference between the reviewed algorithms also becomes apparent, when we check their runtimes against each other. After the dictionary is computed, HMP takes 0.8 seconds for the entire computation on a typical 300×300 image. Because, as already mentioned in the KD-section, KD and HKD perform best when run with a combination of all three match kernels, the evaluation of just these three kernels takes longer than 5.5 seconds. Also in contrast to HMP they need non-linear

SVMs to achieve their best results.
But as already mentioned in the abstract, there are fields, where HMPs performance decreases significantly. We consider a class that cannot be characterized with just a few significant features, but only with a pattern of a bigger amount of features. Then HMP, as a sparse code, cannot pictures the class characteristics accurate and therefore wouldn't be applicable to this problem.
But when applicable, HMP should be used over the other reviewed algorithms because of its performance considered with its runtime.

References

[1] Michal Aharon, Michael Elad, and Alfred Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. In *IEEE TRANSACTIONS ON SIGNAL PROCESSING*, volume VOL. 54, NOVEMBER 2006.

[2] Liefeng Bo, Dieter Fox, and Xiaofeng Ren. Kernel descriptors for visual recognition. *NIPS*, 2010.

[3] Liefeng Bo, Dieter Fox, and Xiaofeng Ren. Hierarchical matching pursuit for image classification: Architecture and fast algorithms. *NIPS*, 2011.

[4] Liefeng Bo, Dieter Fox, Xiaofeng Ren, and Kevin Lai. Object recognition with hierarchical kernel descriptors. *CVPR*, 2011.

[5] Liefeng Bo and Cristian Sminchisescu. Efficient match kernels between sets of features for visual recognition. *NIPS*, 2009.

[6] Oren Boiman, Eli Shechtman, and Michal Iran. In defense of nearest-neighbor based image classification. *CVPR*, 2008.

[7] Anna Bosch, Andrew Zisserman, and Xavier Munoz. Image classification using random forests and ferns, 2007.

[8] Dan Ciresan, Ueli Meier, and Juergen Schmidhuber. Multi-column deep neural networks for image classification. *CVPR*, 2012.

[9] Dan Ciresan, Ueli Meier, Juergen Schmidhuber, Jonathan Masci, and Luca M. Gambardella. High-performance neural networks for visual object classification. Technical report, Dalle Molle Institute for Arti
cial Intelligence, 2011.

[10] Adam Coates and Andrew Y. NG. Selecting receptive fields in deep networks. *NIPS*, 2011.

[11] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. *CVPR*, 2005.

[12] Peter Gehler and Sebastian Nowozin. On feature combination for multiclass object classification. *ICCV*, 2009.

[13] David Haussler. Convolution kernels on discrete structures. Technical report, University of California at Santa Cruz, 1999.

[14] Geoffrey E. Hinton, Simon Osindero, and Teh Yee-Whye. A fast learning algorithm for deep belief nets. *Neural Computation*, 2006.

[15] Kevin Jarrett, Koray Kavukcuoglu, MarcAurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? *ICCV*, 2009.

[16] Yangqing Jia, Chang Huang, and Trevor Darrell. Beyond spatial pyramids: Receptive field learning for pooled image features, 2012.

[17] Fuxin Li, Joao Carreira, and Cristian Sminchisescu. Object recognition as ranking holistic figure-ground hypotheses. *CVPR*, 2010.

[18] David G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[19] Gabriel L. Oliveira, Erickson R. Nascimento, Antonio W. Vieira, and Mario F. M. Campos. Sparse spatial coding: A novel approach for efficient and accurate object recognition. *ICRA*, 2012.

[20] Mehul Parsana, Sourangshu Bhattacharya, Chiranjib Bhattacharyya, and K. R. Ramakrishnan. Kernels on attributed pointsets with applications. *NIPS*, 2007.

[21] G. Salton and M. J. McGill. Orderless document representation: frequencies of words from a dictionary, 1983.

[22] Bernhard Schoelkopf, Alexander Smola, and Klaus-Robert Mueller. Nonlinear component analysis as a kernel eigenvalue problem. Technical report, Max-Planck-Institut fuer biologische Kybernetik Tuebingen, 1996.

[23] Sinisa Todorovic and Narendra Ahuja. Learning subcategory relevances for category recognition. *CVPR*, 2008.

[24] Jingjing Yang, Yuanning Li, Yonghong Tian, Lingyu Duan, and Wen Gao. Group-sensitive multiple kernel learning for object categorization. *ICCV*, 2009.