ABSTRACT

Computer vision researchers have developed various learning methods based on the bag of words model for image related tasks, including image categorization, image retrieval and texture classification. In this model, images are represented as histograms of visual words (or textons) from a vocabulary that is obtained by clustering local image descriptors. Next, a classifier is trained on the data. Most often, the learning method is a kernel-based one. Various kernels can be plugged in to the kernel method. Popular choices, besides the linear kernel, are the intersection, the Hellinger’s, the $\chi^2$ and the Jensen-Shannon kernels. Recent object recognition results indicate that the novel PQ kernel seems to improve the accuracy over most of the state of the art kernels. The PQ kernel is inspired from a set of rank correlation statistics specific for ordinal data, that are based on counting concordant and discordant pairs among two variables. In this paper, the PQ kernel is used for the first time for the task of texture classification. The PQ kernel is computed in $O(n \log n)$ time using an efficient algorithm based on merge sort. The algorithm leverages the use of the PQ kernel for large vocabularies.

Texture classification experiments are conducted to compare the PQ kernel with other state of the art kernels on two benchmark data sets of texture images. The PQ kernel has the best accuracy on both data sets. In terms of time, the PQ kernel becomes comparable with the state of the art Jensen-Shannon kernel. In conclusion, the PQ kernel can be used to obtain a better pairwise similarity between histograms, which, in turn, improves the texture classification accuracy.

Keywords
kernel methods, rank correlation measure, ordinal measure, visual words, bag of words, textons, texture analysis, texture classification.

1 INTRODUCTION

The classical problem in computer vision is that of determining whether or not the image data contains some specific object, feature, or activity. Particular formulations of this problem are object recognition, image classification, texture classification. Computer vision researchers have recently developed sophisticated methods for such image related tasks. Among the state of the art models are discriminative classifiers using the bag of words (BOW) representation [ZMLS07, SRE+05] and spatial pyramid matching [LSP06], generative models [FFFP07] or part-based models [LSP05a]. The BOW model, which represents an image as a histogram of local features, has demonstrated impressive levels of performance for image categorization [ZMLS07], image retrieval [PCI+07], texture classification [XZY10], or related tasks [IP13].

One of the early approaches of building a vocabulary of features is [LM01]. The main idea in [LM01] is to construct a vocabulary of prototype tiny surface patches, called 3D textons. Textons obtained by k-means clustering are used for texture classification. Textons are elements of texture perception that are widely used in texture analysis. In [XZY10], a novel texture classification method via patch-based sparse texton learning is proposed. The dictionary of textons is learned by applying sparse representation to image patches in the training data set. The authors of [VZ05] model textures by the joint distribution of filter responses. This distribution is represented by the frequency histogram of textons.

This work is focused on improving the BOW model by treating texton histograms as ordinal data. In the BOW model, a vocabulary (or codebook) of textons
A bag of visual words model. The PQ kernel for texton histograms is discussed in Section 3. The efficient algorithm to compute the PQ kernel is presented in Section 4. All the experiments are presented in Section 5. Finally, the conclusions are drawn in Section 6.

2 BAG OF VISUAL WORDS MODEL

In computer vision, the BOW model can be applied to image classification and related tasks, by treating image descriptors as words. A bag of visual words is a vector of occurrence counts of a vocabulary of local image features. This representation can also be described as a histogram of visual words. The vocabulary is usually obtained by vector quantizing image features into visual words. In the context of texture classification, visual words are also referred to as textons.

The BOW model can be divided in two major steps. The first step is for feature detection and representation. The second step is to train a kernel method in order to predict the class label of new image. The feature detection and representation step works as follows. Features are detected using a regular grid across the input image. At each interest point, a SIFT feature [Low99] is computed. This approach is known as dense SIFT [DT05, BZM07]. Next, SIFT descriptors are vector quantized into visual words and a vocabulary (or codebook) of visual words is obtained. The vector quantization process is done by k-means clustering [LM01], and visual words are stored in a randomized forest of k-d trees [PCI07] to reduce search cost. The frequency of each visual word is then recorded in a histogram which represents the final feature vector for the image. The histograms of visual words enter the training step. A kernel method is used for training. Several kernel functions can be used, such as the linear kernel, the intersection kernel, the Hellinger’s kernel, the \( \chi^2 \) kernel, the JS kernel, or the recently introduced PQ kernel [IP13].

It is important to mention that the model described so far ignores spatial relationships between image features. Despite of this fact, visual words showed a high discriminative power and have been used for region or image level classification [CDF+04, FFP05, ZMLS07]. However, the performance can be improved by including spatial information. This can be achieved by dividing the image into spatial bins. The frequency of each visual word is then recorded in a histogram for each bin. The final feature vector of the image is a concatenation of these histograms, which gives a spatial pyramid representation [LSP06] of the image.
3 PQ KERNEL

The PQ kernel was introduced in [IP13]. For the sake of completion, the PQ kernel is also presented next. However, it is worth mentioning that the presentation of the PQ kernel provided in this section follows the original presentation given in [IP13].

All common kernels used in computer vision treat histograms either as finite probability distributions, for instance, the Jensen-Shannon kernel, either as quantitative random variables whose values are the frequencies of different textons in the respective images, for instance, the Hellinger’s kernel (Bhattacharyya’s coefficient) and the $\chi^2$ kernel. Even the linear kernel can be seen as the Pearson’s correlation coefficient if the two histograms are standardized. However, the histograms of textons can also be treated as ordinal data, in which case, the values of his-
tograms will be the ranks of visual words according to their frequencies in the image, rather than the actual values of these frequencies.

An entire set of correlation statistics for ordinal data are based on the number of concordant and discordant pairs among two variables. The number of concordant pairs among two variables (or histograms) $X, Y \in \mathbb{R}^n$ is:

\[
P = \left| \{(i, j) : 1 \leq i < j \leq n, (x_i - x_j)(y_i - y_j) > 0\} \right|.
\]

In the same manner, the number of discordant pairs is:

\[
Q = \left| \{(i, j) : 1 \leq i < j \leq n, (x_i - x_j)(y_i - y_j) < 0\} \right|.
\]

Goodman and Kruskal’s gamma [UC04] is defined as:

\[
\gamma = \frac{P - Q}{P + Q}.
\]

Kendall developed several slightly different types of ordinal correlation as alternatives to gamma. Kendall’s tau-a [UC04] is based on the number of concordant versus discordant pairs, divided by a measure based on the total number of pairs ($n$ is the sample size):

\[
\tau_a = \frac{P - Q}{\frac{m(n-1)}{2}}.
\]

Kendall’s tau-b [UC04] is a similar measure of association based on concordant and discordant pairs, adjusted for the number of ties in ranks. It is calculated as the difference between $P$ and $Q$ divided by the geometric mean of the number of pairs not tied in $X$ and the number of pairs not tied in $Y$, denoted by $X_0$ and $Y_0$, respectively:

\[
\tau_b = \frac{P - Q}{\sqrt{(P + Q + X_0)(P + Q + Y_0)}}.
\]

All the above three correlation statistics are very related. If $n$ is fixed and $X$ and $Y$ have no ties, then $P, X_0$ and $Y_0$ are completely determined by $n$ and $Q$. Actually, all are based on the difference between $P$ and $Q$, normalized differently. Following this observation, the PQ kernel between two histograms $X$ and $Y$ is defined as:

\[
k_{PQ}(X,Y) = 2(P - Q).
\]

The following theorem proves that PQ is indeed a kernel, by showing how to build its feature map.

**Theorem 1** The function denoted by $k_{PQ}$ is a kernel function.

**Proof:** To prove that $k_{PQ}$ is a kernel, the explicit feature map induced by $k_{PQ}$ is provided next. Let $X, Y \in \mathbb{R}^n$ be two histograms of visual words. Let $\Psi$ be defined as follows:

\[
\Psi : \mathbb{R}^n \rightarrow M_{n,n} \quad \Psi(X) = \langle \Psi(X)_{i,j} \rangle_{1 \leq i \leq n, 1 \leq j \leq n},
\]

with

\[
\Psi(X)_{i,j} = \begin{cases} 
1 & \text{if } x_i > x_j \\
-1 & \text{if } x_i < x_j \\
0 & \text{if } x_i = x_j
\end{cases}, \forall 1 \leq i, j \leq n.
\]

Note that $\Psi$ associates to each histogram a matrix that describes the order of its elements.

If matrices are treated as vectors, then the following equality is true:

\[
k_{PQ}(X,Y) = 2(P - Q) = \langle \Psi(X), \Psi(Y) \rangle,
\]

where $\langle \cdot, \cdot \rangle$ denotes the scalar product. This proves that $k_{PQ}$ is a kernel and provides the explicit feature map induced by $k_{PQ}$.

According to [VZ10], the feature vectors of $\gamma$-homogeneous kernels should be $L_2$-normalized. Being linear in the feature space, PQ is a 2-homogeneous kernel and the feature vectors should be $L_2$-normalized. Therefore, in the experiments, the PQ kernel is $L_2$-normalized.

\[
\hat{k}_{PQ}(X,Y) = \frac{k_{PQ}(X,Y)}{\sqrt{k_{PQ}(X,X) \cdot k_{PQ}(Y,Y)}}.
\]

4 EFFICIENT ALGORITHM FOR PQ KERNEL

It is important to note that for vocabularies with more than 1000 textons, the kernel trick should be employed to directly obtain the PQ kernel matrix instead of computing its feature maps, since there is a quadratic dependence between the size of the feature map and the
number of textons. In general, the kernel trick [STC04] refers to using the dual representation given by the kernel matrix of pairwise similarities between samples instead of the primal representation given by the feature maps. This will greatly reduce the space cost of the PQ kernel. The problem of the time complexity remains to be solved. In the dual form, the PQ kernel between two histograms can be directly computed using an algorithm with a time complexity of \( O(n \log n) \), \( n \) being the number of textons. The algorithm described in this section follows the work of [KNR66], which proposed a similar algorithm for computing the Kendall’s tau measure. The algorithm is based on the key insight of Kendall which, in his book [Ken48], proves that the number of discordant pairs \( Q \) between two rankings is equal to the number of interchanges (or swaps), denoted by \( s \), required to transform one ranking into the other. Another important observation is that, given a pair of indices \((i, j)\), interchanging the corresponding values in the same time in the two histograms \( X \) and \( Y \), respectively, does not change the number of discordant pairs. Thus, sorting the two histograms \( X \) and \( Y \) in the same time, using as sorting criteria first the values of \( X \) and second (for ties in \( X \)) the values of \( Y \), will end up with a new pair of variables \( X' \) and \( Y' \) that will have the same number of discordant pairs as \( X \) and \( Y \). At this point, \( X' \) is completely sorted while \( Y' \) is not. To represent the same ranking as \( X' \), \( Y' \) must also be sorted. This implies that the number of swaps needed to sort \( Y' \) will be the number of discordant pairs between \( X' \) and \( Y' \). As described above, the number of swaps needed to sort \( Y' \) is also the number of discordant pairs between \( X \) and \( Y \). A slightly modified merge sort algorithm can be used to sort \( Y' \) while computing the number \( s \). Algorithm 1 computes \( k_{PQ} \) between two histograms \( X, Y \in \mathbb{R}^n \) based on these observations.

**Algorithm 1: PQ Kernel Algorithm**

1. Sort \( X \) and \( Y \) in the same time using merge sort, according to the values in \( X \) in ascending order. If two values in \( X \) are equal, sort them according to \( Y \) in ascending order.
2. Compute the total number of pairs as \( t = n(n - 1)/2 \).
3. Compute the number of equal pairs in \( X \) as \( e_X \).
4. Compute the number of pairs that are equal in both \( X \) and \( Y \) as \( e_{X,Y} \).
5. Sort \( Y \) using merge sort in ascending order and sum the differences of swap positions into \( s \).
6. Compute the number of equal pairs in \( Y \) as \( e_Y \).
7. Finally, compute \( k_{PQ}(X,Y) = 2(t + e_{X,Y} - e_X - e_Y - 2s) \).

Steps 1 and 5 of Algorithm 1 compute the number of discordant pairs \( Q \) between \( X \) and \( Y \), given that \( Q = s \). The number of concordant pairs \( P \) is completely determined by the number of discordant pairs, the total numbers of pairs denoted by \( t = n(n - 1)/2 \), the number of equal pairs in \( X \) denoted by \( e_X \), the number of equal pairs in \( Y \) denoted by \( e_Y \), and the number of pairs that are equal in both \( X \) and \( Y \) denoted by \( e_{X,Y} \). More precisely, \( P \) can be expressed as follows:

\[
P = t + e_{X,Y} - e_X - e_Y - s.
\]

Thus, the difference between \( P \) and \( Q \) can also be expressed in terms of \( t \), \( e_X \), \( e_Y \), and \( e_{X,Y} \), as in step 7 of Algorithm 1.

The analysis of the computational complexity of Algorithm 1 is straightforward. The time for steps 2 and 7 is constant. Steps 1 and 5 are based on merge sort which is known to work in \( O(n \log n) \) time. Because \( X \) and \( Y \) are already sorted, steps 3, 4 and 6 can be computed in linear time with respect to the number of textons \( n \). Consequently, the time complexity of Algorithm 1 is \( O(n \log n) \).

5 EXPERIMENTS

5.1 Data Sets Description

Texture classification experiments are presented on two benchmark data sets of texture images. The first experiment is conducted on the Brodatz data set [Bro66]. This data set is probably the best known benchmark used for texture classification, but also one of the most difficult, since it contains 111 classes with only 9 samples per class. Samples of \( 213 \times 213 \) pixels are cut using a 3 by 3 grid from larger images of \( 640 \times 640 \) pixels. Figure 1 presents three sample images per class of three classes randomly selected from the Brodatz data set.

The second experiment is conducted on the UIUCTex data set of [LSP05b]. It contains 1000 texture images of \( 640 \times 480 \) pixels representing different types of textures such as bark, wood, floor, water, and more. There are 25 classes of 40 texture images per class. Textures are viewed under significant scale, viewpoint and illumination changes. Images also include non-rigid deformations. This data set is available for download at http://www-cvr.ai.uiuc.edu/ponce_grp. Figure 2 presents four sample images per class of four classes representing bark, brick, pebbles, and plaid.

5.2 Implementation and Learning Method

Details about the particularities of the learning framework are given next. In the feature detection and representation step, a variant of dense SIFT descriptors extracted at multiple scales is used [BZM07]. The implementation of the BOW model is mostly based on the VLFeat library [VF08]. Various state of the art kernels are compared with the PQ kernel. The kernels proposed for evaluation are...
the $L_2$-normalized linear kernel, the $L_1$-normalized Hellinger’s kernel, the $L_1$-normalized intersection kernel, the $L_1$-normalized Jensen-Shannon kernel, and the $L_2$-normalized PQ kernel. The norms are chosen according to the authors of [VZ10], that state that $\gamma$-homogeneous kernels should be $L_\gamma$-normalized. It is important to mention that all these kernels are used in the dual form, that implies using the kernel trick to directly build kernel matrices of pairwise similarities between samples.

A state of the art kernel classifier is used in the experiments, namely the Support Vector Machines (SVM), which is very well-suited for binary classification tasks. The SVM [CV95] tries to find the hyperplane that maximally separates the training examples belonging to the two classes. In the experiments, the SVM classifier based on the one versus all scheme is used for the multi-class texture classification tasks. More details about the SVM are discussed in [STC04]. The important fact is that the SVM can be trained in such a way that the feature maps are not needed, only the pairwise kernel matrices being required.

5.3 Brodatz Experiment

Preliminary experiments were performed on the Brodatz data set for parameter tuning. First, a set of experiments were conducted to choose the size of the vocabulary. Vocabularies of 1000, 2000, 3000 and 4000 textons were tested, respectively. The best results were obtained with a vocabulary of 2000 textons. Consequently, this vocabulary was selected for the subsequent experiments. The regularization parameter $C$ of the SVM algorithm was chosen by cross-validation for each kernel, on a subset of texture images from the Brodatz data set.

Table 1 compares the accuracy rate of the SVM based on the PQ kernel with the accuracy rates of the SVM based on various state of the art kernels, using 3 random samples per class for training. The accuracy rates presented in Table 1 are actually averages of accuracy rates obtained over 50 runs for each method, in order to reduce accuracy variation induced by the random selection of training and testing samples. The accuracy of the state of the art kernels is well above the accuracy of the baseline linear kernel. More precisely, the state of the art kernels are roughly $2 - 3\%$ better than the baseline method. However, the BOW model based on the linear kernel achieves comparable results with other state of the art techniques, such as [LSP05b]. In [LSP05b], the accuracy rate reported on the Brodatz data set using the same setup with 3 training samples per class is 88.15%. Among the state of the art kernels, the PQ kernel gives the best accuracy rate. Indeed, the accuracy of the PQ kernel (92.94\%) is roughly $0.6 - 0.8\%$ above the accuracy rates of the other state of the art kernels. The empirical results on the Brodatz data set show that the PQ kernel can outperform the other evaluated kernels for the task of texture classification.

Table 1 also provides the time required by each kernel to produce the kernel matrix that contains pairwise similarities between all the texture samples from Brodatz. Thus, the kernel matrix has 999 rows and 999 columns. The time was measured on a computer with Intel Core i7 2.3 GHz processor and 8 GB of RAM memory using a single Core. Using the efficient algorithm that requires $O(n \log n)$ time to compute the PQ kernel, the
5.4 UIUCTex Experiment

As in the previous case, preliminary experiments were performed on the UIUCTex data set for parameter tuning. Various vocabularies of 1000, 2000, 3000 and 4000 textons were tested, respectively. On this data set, the accuracy seems to improve as the vocabulary dimension is increased. Consequently, the vocabulary of 4000 textons was selected for the subsequent experiments. The regularization parameter $C$ of the SVM algorithm was chosen by cross-validation for each kernel, on a subset of the UIUCTex data set.

Table 2 compares the accuracy rate of the SVM based on the PQ kernel with the accuracy rates of the SVM based on various state of the art kernels, using 20 random samples per class for training. This means that half of the images are used for training, and the other half for testing. The results of the PQ kernel on the UIUCTex data set are consistent with the results obtained on the Brodatz data set. More precisely, the PQ kernel outperforms again the other kernels. The accuracy rate of the PQ kernel is roughly $0.5 - 0.6\%$ above the accuracies of the other kernels.
The dual form in roughly 1
parable to the time of the JS kernel. The PQ kernel is
computationally expensive kernel, but the running time
a single Core. Again, the PQ kernel remains the most
2
Table 2 was measured on a computer with Intel Core i7
ber of textons is 4000 this time. The time reported in
matrix has 1000 rows and 1000 columns, but the num-
all the texture samples from the UIUCTex data set is
nel matrix that contains pairwise similarities between
The time required by each kernel to produce the ker-
result than the linear kernel. The best accuracy rate
91.74% is obtained by the PQ kernel. Overall, the
empirical results indicate that the PQ kernel can have
better performance for texture classification than the
other state of the art kernels.

The time required by each kernel to produce the ker-
el matrix that contains pairwise similarities between
all the texture samples from the UIUCTex data set is
also presented in Table 2. In this experiment, the kernel
matrix has 1000 rows and 1000 columns, but the num-
ber of textons is 4000 this time. The time reported in
Table 2 was measured on a computer with Intel Core i7
2.3 GHz processor and 8 GB of RAM memory using
a single Core. Again, the PQ kernel remains the most
computationally expensive kernel, but the running time
of the PQ kernel computed with Algorithm 1 is com-
parable to the time of the JS kernel. The PQ kernel is
roughly 1.6 times slower than the JS kernel. Compared
to the Brodatz experiment, the vocabulary is twice as
large, containing 4000 textons instead of 2000. How-
ever, the proportion between the time of the PQ kernel
and the time of the JS kernel does not change by much.
The time to compute the quadratic feature maps of the
PQ kernel could not be measured on the same machine,
since the 8 GB of RAM memory is not enough to store
the 1000 feature maps of 16 million features each. This
supports the claim that Algorithm 1 makes the PQ ker-
nel practical for large vocabularies, of more than 1000
words. Another fact to support this claim is that the
object recognition experiments presented in [IP13] are
based on vocabularies of only 500 words, since it is
more expensive to compute feature maps of the PQ ker-
nel. More precisely, the feature maps can be computed
in $O(n^2)$ time, while the PQ kernel can be computed in
the dual form in $O(n \log n)$ time with the efficient al-
gorithm presented in this paper.

<table>
<thead>
<tr>
<th></th>
<th>Lin. $L_2$</th>
<th>Hel. $L_1$</th>
<th>Int. $L_1$</th>
<th>JS $L_1$</th>
<th>PQ $L_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>84.44%</td>
<td>91.24%</td>
<td>88.22%</td>
<td>91.17%</td>
<td>91.74%</td>
</tr>
<tr>
<td>Time (s)</td>
<td>3.4</td>
<td>3.8</td>
<td>18.2</td>
<td>199.8</td>
<td>328.5</td>
</tr>
</tbody>
</table>

Table 2: Accuracy rates and running times of various kernels on the UIUCTex data set using 20 random samples per class for training. The PQ kernel is compared with the state of the art kernels on a vocabulary of 4000 textons. The running time required to compute the pairwise similarity matrix for each kernel is reported in this table.

The paper showed that the PQ kernel can be success-
fully used for texture classification. An efficient al-
gorithm was used for computing the PQ kernel in the
dual form. The algorithm is based on merge sort and
needs only $O(n \log n)$ time. The algorithm leverages
the use of the PQ kernel in BOW models with large
vocabularies. Several texture classification experiments
were conducted to show that the PQ kernel can indeed
be used with large vocabularies of up to 4000 textons.
Furthermore, the empirical results showed that the PQ
kernel can improve the BOW model for texture clas-
sification in terms of accuracy, by treating texton his-
tograms as ordinal data.

In future work, other methods inspired from ordinal
measures can be investigated. For example, the most
common correlation statistic for ordinal data, namely
the Spearman’s rank-order coefficient [UC04], or its
version, namely the Spearman’s footrule, are suc-
cessfully used in text processing [DP09]. It is perfectly
reasonable to use them for image categorization in the
context of the BOW model, since the BOW model is
also inspired from text processing (more precisely, from
text retrieval). Methods to transform such ordinal mea-
sures into kernels would also have to be developed.

7 ACKNOWLEDGMENTS

The contribution of the authors to this paper is equal.

8 REFERENCES


