Patch Autocorrelation Features: A translation and rotation invariant approach for image classification

Radu Tudor Ionescu · Andreea Lavinia Ionescu · Josiane Mothe · Dan Popescu

Abstract The autocorrelation is often used in signal processing as a tool for finding repeating patterns in a signal. In image processing, there are various image analysis techniques that use the autocorrelation of an image in a broad range of applications from texture analysis to grain density estimation. This paper provides an extensive review of two recently introduced and related frameworks for image representation based on autocorrelation, namely Patch Autocorrelation Features (PAF) and Translation and Rotation Invariant Patch Autocorrelation Features (TRIPAF). The PAF approach stores a set of features obtained by comparing pairs of patches from an image. More precisely, each feature is the euclidean distance between a particular pair of patches. The proposed approach is successfully evaluated in a series of handwritten digit recognition experiments on the popular MNIST data set. However, the PAF approach has limited applications, because it is not invariant to affine transformations. More recently, the PAF approach was extended to become invariant to image transformations, including (but not limited to) translation and rotation changes. In the TRIPAF framework, several features are extracted from each image patch. Based on these features, a vector of similarity values is computed between each pair of patches. Then, the similarity vectors are clustered together such that the spatial offset between the patches of each pair is roughly the same. Finally, the mean and the standard deviation of each similarity value are computed for each group of similarity vectors. These statistics are concatenated to obtain the TRIPAF feature vector. The TRIPAF vector essentially records information about the repeating patterns within an image at various spatial offsets. After presenting the two approaches, several optical char-

R. T. Ionescu
University of Bucharest, 14 Academiei, Bucharest, Romania, E-mail: raducu.ionescu@gmail.com

A. L. Ionescu, D. Popescu
Politehnica University of Bucharest, 313 Splaiul Independentei Street, Bucharest, Romania, E-mail: andreea.lavinia@ymail.com, dan_popescu_2002@yahoo.com

J. Mothe
École Supérieure du Professorat et de l'Éducation, Université de Toulouse, IRIT, UMR 55005 CNRS, 118, Route de Narbonne, Toulouse, France, E-mail: josiane.mothe@irit.fr

acter recognition and texture classification experiments are conducted to evaluate the two approaches. Results are reported on the MNIST (98.93%), the Brodatz (96.51%), and the UIUCTex (98.31%) data sets. Both PAF and TRIPAF are fast to compute and produce compact representations in practice, while reaching accuracy levels similar to other state-of-the-art methods.

Keywords image autocorrelation · patch-based method · optical character recognition · texture classification · rotation invariant method · translation invariant method · MNIST · Brodatz · UIUCTex

1 Introduction

Artificial intelligence is a vast domain with important applications in many fields (Valipour et al, 2013; Valipour, 2015a,b; Laalaoui and Bouguila, 2015; Ionescu et al, 2016), including computer vision (Szeliski, 2010; Ionescu and Popescu, 2016). The classical problem in computer vision is that of determining whether or not the image data contains some specific object, feature, or activity. Object recognition, face recognition, texture classification and optical character recognition are particular formulations of this problem, the last two of them being approached in the current work. Optical character recognition is widely employed as a form of acquiring digital information from printed paper documents such as passport documents, business cards, or even historical documents, so that they can be electronically stored, edited or searched. On the other hand, texture classification is extremely useful in industrial and biomedical surface inspection, for example for defects or disease identification. Texture classification is also employed for ground classification and segmentation of satellite or aerial imagery. Hence, there are many potential applications of the two frameworks presented in this work.

Computer vision researchers have developed sophisticated methods for various image classification tasks. Virtual SVM (DeCoste and Schölkopf, 2002), boosted stumps (Kégl and Busa-Fekete, 2009), convolutional neural networks (LeCun et al, 1998; Ciresan et al, 2012) or deep Boltzmann machines (Salakhutdinov and Hinton, 2009) are sophisticated state-of-the-art learning frameworks used for optical character recognition. However, simple methods such as the k-Nearest Neighbor (k-NN) model have also obtained very good recognition results, sometimes being much better than more sophisticated techniques. Some of the techniques that fall in this category of simple yet very accurate methods, and worth to be mentioned, are the k-NN models based on Tangent distance (Simard et al, 1996), shape context matching (Belongie et al, 2002), non-linear deformation (Keysers et al, 2007), and Local Patch Dissimilarity (Dinu et al, 2012), respectively. While optical character recognition can successfully be approached with a simple technique, more complex image classification tasks require more sophisticated methods, naturally because the methods have to take into account several aspects such as translation, rotation, and scale variations, illumination changes, viewpoint changes, partial occlusions and noise. Among the rather more elaborate state-of-the-art models used in image classification are bag of visual words (Csurka et al, 2004; Liu et al, 2011; Ionescu et al, 2013), Fisher Vectors (Perronnin and Dance, 2007; Perronnin et al, 2010), and deep learning models (Krizhevsky et al, 2012; Socher et al, 2012; Simonyan and Zisserman, 2014; Szegedy et al, 2015).

This paper presents two feature representation for images that are based on the autocorrelation of the image with itself. The first representation is a simple approach in which each feature is determined by the euclidean distance between a pair of patches extracted from the image. To reduce the time necessary to compute the feature representation, patches are extracted at a regular interval by using a dense grid over the image. This feature representation, which was initially introduced in (Ionescu et al, 2015a), is termed Patch Autocorrelation Features (PAF). Ionescu et al (2015a) have shown that PAF exibits state-of-the-art performance in optical character recognition. However, the PAF approach is affected by affine transformations and it requires additional improvements in order to solve[1] more complex image classification tasks, such as texture classification, for example. Ionescu et al (2015b) have proposed an extension of PAF that is invariant to image transformations, such as translation and rotation changes. Naturally, this extension involves more elaborate computations, but the resulted feature vector is actually more compact, since it involves the vector quantization of pairs of patches according to the spatial offset between the patches in each pair. Instead of directly comparing the patches, the extended approach initially extracts a set of features from each patch. The extended feature representation is termed Translation and Rotation Invariant Patch Autocorrelation Features (TRIPAF).

Several handwritten digit recognition experiments are conducted in this work in order to demonstrate the performance gained by using the PAF representation instead of a standard representation based on raw pixel values (a common approach for optical character recognition). More precisely, experiments are performed using two different classifiers (k-NN and SVM) on original and deslanted images from the MNIST data set. Experiments are conducted on only 1000 images, but also on the entire MNIST data set. The empirical results obtained in all the experiments indicate that the PAF representation is constantly better than the standard representation. The best results obtained with the PAF representation are similar to some of the state-of-the-art methods (Kégl and Busa-Fekete, 2009; Salakhutdinov and Hinton, 2009). The advantage of PAF is that it is easy and fast to compute.

A series of texture classification experiments are also conducted in the present work to evaluate the extended version of PAF, namely TRIPAF, in a more difficult setting. Two popular texture classification data sets are used for the evaluation, specifically Brodatz and UIUCTex. The empirical results indicate that TRIPAF can significantly improve the performance over a system that uses the same features, but extracts them from entire images. By itself, TRIPAF is invariant to rotation and translation changes, and for this reason, it makes sense to combine it with a scale invariant system in order to further improve the performance. As such, the system based on TRIFAF was combined with the bag of visual words (BOVW) framework of Ionescu et al (2014b) through multiple kernel learning (MKL) (Gonen and Alpaydin, 2011). The BOVW framework is based on clustering SIFT descriptors (Lowe, 1999) into visual words, which are scale invariant. The performance level of the combined approach is comparable to the state-of-the-art methods for texture classification (Zhang et al, 2007; Nguyen et al, 2011; Quan et al, 2014).

[1] with a reasonable degree of accuracy

The idea behind the PAF and the TRIPAF approaches is to represent the repeating patterns that occur in the image through a set of features. Thus, they provide a way of representing the image autocorrelation that is useful for learning the repeating patterns in order to classify images. A key difference from previous works using the autocorrelation for texture classification (Horikawa, 2004b,a; Toyoda and Hasegawa, 2007) is that TRIPAF obtains significantly better performance for this task. Moreover, TRIPAF provides a compact representation that can be computed fast.

To summarize, the contributions of this overview paper are:

- It provides an extensive description of PAF, a simple and efficient feature representation for handwritten digit recognition and simple image classification tasks (Section 3);
- It gives an extensive presentation of TRIPAF, a compact, efficient and invariant feature representation for texture classification and more complex image classification task (Section 4);
- It presents classification results obtained with PAF on the MNIST data set (Section 6);
- It presents classification results obtained with TRIPAF on the Brodatz and UIUCTex data sets (Section 7);
- It provides empirical evidence that TRIPAF is invariant to rotation and mirror transformations (Section 7.6).

2 Related Work

2.1 Autocorrelation in Image Analysis

The autocorrelation is a mathematical tool for finding repeating patterns which has a wide applicability in various domains such as signal processing, optics, statistics, image processing, or astrophysics. In signal processing, it is used to find repetitive patterns in a signal over time. Images can also be regarded as spatial signals. Thus, it makes sense to measure the spatial autocorrelation of an image. Certainly, the autocorrelation has already been used in image and video processing (Brochard et al, 2001; Popovici and Thiran, 2001; Horikawa, 2004b; Toyoda and Hasegawa, 2007; Kameyama and Phan, 2013; Yi and Pavlovic, 2013; Haouas et al, 2016). Brochard et al (2001) present a method for feature extraction from texture images. The method is invariant to affine transformations, this being achieved by transforming the autocorrelation function (ACF) and then by determining an invariant criterion which is the sum of the coefficients of the discrete correlation matrix.

A method for using higher order local autocorrelations (HLAC) of any order as features is presented in (Popovici and Thiran, 2001). The method exploits the special form of the inner products of autocorrelations and the properties of some kernel functions used by SVM. Toyoda and Hasegawa (2007) created large mask patterns for HLAC features and constructed multi-resolution features to support large displacement regions. The method is applied to texture classification and face recognition. However, Toyoda and Hasegawa (2007) apply their method on only 32 Brodatz textures and obtain their best accuracy rate of 83.0% with 19 training samples per class. By contrast, TRIPAF is evaluated on all the 111

Brodatz textures using only 3 training samples per class, a considerably more difficult setting. However, the accuracy of TRIPAF (92.85%) is nearly 10% better.

Kernel canonical correlation analysis based on autocorrelation kernels is applied to invariant texture classification in (Horikawa, 2004b). The autocorrelation kernels represent the inner products of the autocorrelation functions of original data. In (Horikawa, 2004a), SVM based on autocorrelation kernels are used for texture classification invariant to similarity transformations and noise. Different from these works (Horikawa, 2004b,a), the TRIPAF framework is evaluated on the entire Brodatz data set, demonstrating good results for more than a few kinds of texture.

Kameyama and Phan (2013) explored the nature of the (Local) Higher-Order Moment kernel of various orders as measures for image similarity. The Higher-Order Moment kernel enables efficient utilization of higher-order autocorrelation features in images. Through sensitivity evaluation and texture classification experiments, the authors found that the studied kernel allows to control the selectivity of the similarity evaluation.

Yi and Pavlovic (2013) propose an autocorrelation Cox process that encodes spatio-temporal context in a video. In order to infer the autocorrelation structure relevant for classification, they adopt the information gain feature selection principle. The authors obtain state-of-the-art performance on action recognition in video.

Haouas et al (2016) focus on the combination of two important image features, the spectral and the spatial information, for remote sensing image classification. Their results show the effectiveness of introducing the spatial autocorrelation in the pixel-wise classification process.

2.2 Patch-based Techniques

As many other computer vision techniques (Efros and Freeman, 2001; Deselaers et al, 2005; Guo and Dyer, 2007; Barnes et al, 2011; Michaeli and Irani, 2014), the PAF map considers patches rather than pixels, in order to capture distinctive features such as edges, corners, shapes, and so on. In other words, the PAF representation stores information about repeating edges, corners, and other shapes that can be found in the analyzed image. For numerous computer vision applications, the image can be analyzed at the patch level rather than at the individual pixel level or global level. Patches contain contextual information and have advantages in terms of computation and generalization. For example, patch-based methods produce better results and are much faster than pixel-based methods for texture synthesis (Efros and Freeman, 2001). However, patch-based techniques are still heavy to compute with current machines, as stated in (Barnes et al, 2011). To reduce the time necessary to compute the PAF representation, patches are compared using a grid over the image. The density of this grid can be adjusted to obtain the desired trade-off between accuracy and speed.

A paper that describes a patch-based approach for rapid image correlation or template matching is (Guo and Dyer, 2007). By representing a template image with an ensemble of patches, the method is robust with respect to variations such as local appearance variation, partial occlusion, and scale changes. Rectangle filters are applied to each image patch for fast filtering based on the integral image representation.

An approach to object recognition was proposed by Deselaers et al (2005), where image patches are clustered using the EM algorithm for Gaussian mixture densities and images are represented as histograms of the patches over the (discrete) membership to the clusters. Patches are also regarded in (Paredes et al, 2001), where they are classified by a nearest neighbor based voting scheme.

Agarwal and Roth (2002) describe a method where images are represented by binary feature vectors that encode which patches from a codebook appear in the images and what spatial relationship they have. The codebook is obtained by clustering patches from training images whose locations are determined by interest point detectors.

Passino and Izquierdo (2007) propose an image classification system based on a Conditional Random Field model. The model is trained on simple features obtained from a small number of semantically representative image patches.

The patch transform, proposed in (Cho et al, 2010), represents an image as a bag of overlapping patches sampled on a regular grid. This representation allows users to manipulate images in the patch domain, which then seeds the inverse patch transform to synthesize a modified image.

In (Barnes et al, 2011), a new randomized algorithm for quickly finding approximate nearest neighbor matches between image patches is introduced. This algorithm forms the basis for a variety of applications including image retargeting, completion, reshuffling, object detection, digital forgery detection, and video summarization.

Patches have also been used for handwritten digit recognition in (Dinu et al, 2012). Dinu et al (2012) present a dissimilarity measure for images that quantifies the spatial non-alignment between two images. With some efficiency improvements, Ionescu and Popescu (2013b) show that their method reaches state-of-the-art accuracy rates in handwritten digit recognition. An extended version of the same method has also been used for texture classification in (Ionescu et al, 2014a).

Michaeli and Irani (2014) present an approach for blind deblurring, which is based on the internal patch recurrence property within a single natural image. While patches repeat across scales in a sharp natural image, this cross-scale recurrence significantly diminishes in blurry images. The authors exploit these deviations from ideal patch recurrence as a cue for recovering the underlying (unknown) blur kernel, showing better performance than other state-of-the-art deblurring frameworks.

3 Patch Autocorrelation Features

The Patch Autocorrelation Features are inspired from the autocorrelation used in signal processing. Instead of quantifying the autocorrelation through a coefficient, the approach described in this section is to store the similarities between patches computed at various spatial intervals individually in a vector. This vector contains the Patch Autocorrelation Features that can be used for image classification tasks in which the amount of variation induced by affine transformations is not significant.

The L_2 euclidean distance is used to compute the similarity between patches, but it can be substituted with any other distance or similarity measure that could possibly work better, depending on the task. The only requirement in the case of the euclidean distance is that the patches should all be of the same size, in order to properly compute the similarity between patches. To reduce the number of parameters that need to be tuned, another constraint to use square shaped patches was also added. Formally, the L_2 euclidean distance between two gray-scale patches X and Y each of $p \times p$ pixels is:

$$\Delta_{L_2}(X, Y) = \sqrt{\sum_{i=1}^{p} \sum_{j=1}^{p} (X_{i,j} - Y_{i,j})^2}, \tag{1}$$

where $X_{i,j}$ represents the pixel found on row i and column j in X, and $Y_{i,j}$ represents the pixel found on row i and column j in Y. At this point, one can observe that the PAF representation contains a quadratic number of features with respect to the number of considered patches. More precisely, if n denotes the number of patches extracted from the image, then the resulted number of features will be $n(n-1)/2$, since each pair of patches needs to be considered once and only once. Thus, the computational complexity of PAF is $O(n^2)$. However, a dense grid is applied over the image to reduce the number of patches n. Extracting patches or local features using a sparse or a dense grid is a popular approach in computer vision (Cho et al, 2010; Ionescu and Popescu, 2015). The density of the grid is directly determined by a single parameter that specifies the spatial offset (in pixels) between consecutive patches. In practice, a good trade-off between accuracy and speed can be obtained by adjusting this parameter.

Algorithm 1: PAF Algorithm

```
1  Input:
2  $I$ - a gray-scale input image of $h \times w$ pixels;
3  $p$ - the size (in pixels) of each square-shaped patch;
4  $s$ - the distance (in pixels) between consecutive patches.
5  Initialization:
6  $n \leftarrow ceil((h - p + 1)/s) \cdot ceil((w - p + 1)/s)$;
7  $\mathcal{P} \leftarrow \emptyset$;
8  $v_i \leftarrow 0$, for $i \in \{1, 2, ..., n(n-1)/2\}$;
9  Computation:
10 for $i = 1 : s : w$ do
11     for $j = 1 : s : w$ do
12         $P \leftarrow I_{i:(i+p-1),j:(j+p-1)}$;
13         $\mathcal{P} \leftarrow \mathcal{P} \cup P$;
14 $k \leftarrow 1$;
15 for $i = 1 : n - 1$ do
16     for $j = i + 1 : n$ do
17         $v(k) \leftarrow \Delta_{L_2}(P_i, P_j)$;
18         $k + +$;
19 Output:
20 $v$ - the PAF feature vector with $n(n-1)/2$ components.
```

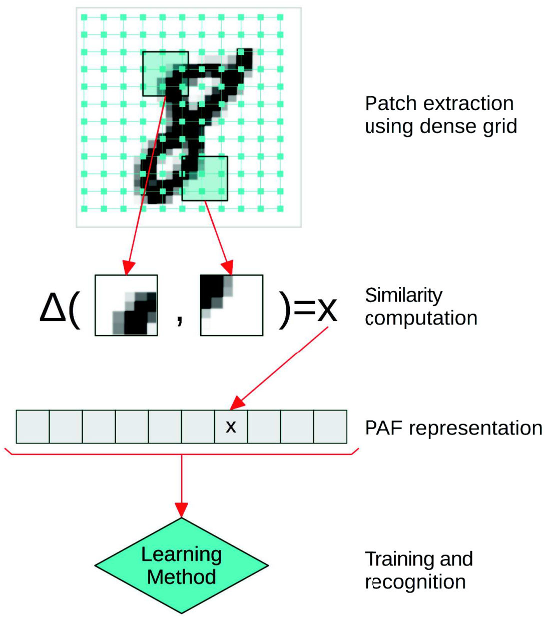


Fig. 1 The classification system based on Patch Autocorrelations Features. The PAF representation is obtained by storing the similarity between pairs of patches that are previously extracted using a dense grid over the input image. The PAF maps of train images are used to learn discriminant features. The trained classifier can then be used to predict class labels for new images represented as PAF vectors.

Algorithm 1 computes the PAF representation for a gray-scale input image I. Notice that gray-scale images are considered for simplicity, but the PAF approach can also be computed on color images. For instance, the PAF representation can be easily adapted to work on color patches. Alternatively, color images can be transformed to gray-scale before any further processing.

The following conventions and mathematical notations are considered in Algorithm 1 and throughout this paper. Arrays and matrices are always considered to be indexed starting from position 1, that is $v = (v_1, v_2, ..., v_{|v|})$, where $|v|$ is the number of components of v. The notations v_i or $v(i)$ are alternatively used to identify the i-th component of v. The sequence $1, 2, ..., n$ is denoted by $1 : n$, but if the step is different from the unit, it can be inserted between the lower and the upper bounds. For example, $1 : 2 : 8$ generates the sequence $1, 3, 5, 7$. Moreover, for a vector v and two integers i and j such that $1 \leq i \leq j \leq |v|$, $v_{i:j}$ denotes the sub-array $(v_i, v_{i+1}, ..., v_j)$. In a similar manner, $X_{i:j,k:l}$ denotes a sub-matrix of the matrix X. Since the analyzed images are reduced to gray-scale, the notion of *matrix* and *image* can be used interchangeably, with the same meaning. In this context, a patch corresponds to a sub-matrix. The set of patches extracted from the input image is denoted by $\mathcal{P} = \{P_1, P_2, ..., P_n\}$.

The first phase of Algorithm 1 (steps 10-13) is to use a dense grid of uniformly spaced interest points in order to extract patches at a regular interval. The patches are stored in a set denoted by $\mathcal{P}$. In the second phase (steps 14-18), the patches are compared two by two using the euclidean distance, and the distance between each pair of patches is then recorded in a specific order in the PAF vector v. More specifically, step 17 of Algorithm 1 is computed according to Equation (1). An important remark is that the features are generated in the same order for every image to ensure that all images are represented in the same way, which is a mandatory characteristic of feature representations used in machine learning. For instance, if the similarity of two patches with the origins given by the coordinate points (x, y) and (u, z) in image I, respectively, is stored at index k in the PAF vector of image I, then the similarity of the patches having the origins in (x, y) and (u, z) in any other image must always be found at index k in the PAF map. This will enable any learning method to find the discriminant features from the PAF vectors. The entire process that involves the computation of the PAF vector for image classification is illustrated in Figure 1.

A quick look at Algorithm 1 is enough to observe that the PAF approach is a simple technique. While it was found to be very effective for rather simple image classification tasks (Ionescu et al, 2015a), the PAF algorithm needs to be adjusted to handle images variations common to more complex image classification tasks. This issue is going to be addressed in the next section.

4 Translation and Rotation Invariant Patch Autocorrelation Features

Several modifications have been proposed in (Ionescu et al, 2015b) in order to transform Patch Autocorrelation Features into an approach that takes into account several image variation aspects including translation and rotation changes, illumination changes, and noise. Instead of comparing the patches based on raw pixel values, a set of features is extracted from each image patch. Depending on the kind of patch features, the method can thus become invariant to different types of image variations. In this work, a set of texture-specific features are used since the approach is evaluated on the texture classification task. These features are described in detail in Section 4.1. The important remark is that a different set of features can be extracted from patches when PAF is going to be used for a different problem.

Rather than computing a single value to represent the similarity between two patches based on the extracted features, the extended PAF approach computes several similarities, one for each feature. More precisely, patches are compared using the Bhattacharyya coefficient between each of their features. Given two feature vectors $f_X, f_Y \in \mathbb{R}^m$ extracted from two image patches X and Y, the vector of similarity values between the two patches $s_{X,Y}$ is computed as follows:

$$s_{X,Y}(i) = \sqrt{f_X(i)} \cdot \sqrt{f_Y(i)}, \forall i \in \{1, 2, ..., m\}. \tag{2}$$

Note that each component of $s_{X,Y}$ is independently computed in the sense that it does not depend on the other features. Therefore, $s_{X,Y}$ can capture different aspects about the similarity between X and Y, since each feature can provide different information. The same features are naturally extracted from each patch, thus $|f_X| = |f_Y| = m$ for every pair of patches (X, Y).

In the basic PAF approach, the next step would be to concatenate the similarity vectors generated by comparing patches two by two. This would be fine, as long as the method relies entirely on the features to achieve invariance to different image transformations. However, further processing can be carried out to ensure that PAF remains invariant to translation and rotation, even if the extracted features are not. Unlike the basic PAF approach, the pairs of patches are vector quantized by the spatial offset between the patches of each pair. Given two patches X and Y having the origins in (x, y) and (u, z), respectively, the spatial offset o between X and Y is measured with the help of the L_2 euclidean distance between their origins:

$$o(X, Y) = \sqrt{(x-u)^2 + (y-z)^2}. \tag{3}$$

In order to cluster pairs of patches together, the spatial offsets are rounded to the nearest integer values. Given two pairs of patches (X, Y) and (U, V), they are clustered together only if $\lfloor o(X, Y) \rceil = \lfloor o(U, V) \rceil$, where $\lfloor x \rceil$ is the rounding function of $x \in \mathbb{R}$, that returns the nearest integer value to x. It is important to note that the similarity vector determined by a pair of patches is included in the cluster, not the patches themselves. Formally, a *cluster* (or a group) of similarity vectors between patches extracted at a given spatial offset k is defined as follows:

$$C_k = \{s_{P_i,P_j} \mid \lfloor o(P_i, P_j) \rceil = k, \forall (P_i, P_j) \in \mathcal{P} \times \mathcal{P}, 1 \leq i < j \leq |\mathcal{P}|\}, \tag{4}$$

where $\mathcal{P} = \{P_1, P_2, ..., P_{|\mathcal{P}|}\}$ is the set of patches extracted from the input image, $o(P_i, P_j)$ is the offset between P_i and P_j determined by Equation (3), and s_{P_i,P_j} is the similarity vector between patches P_i and P_j computed as in Equation (2).

In each cluster, the similarity vectors are computed between patches that reside at a certain spatial offset, in all possible directions. The exact position of each patch is simply disregarded in the clustering process. When the image is translated or rotated, these clusters remain mostly unaffected, because the spatial offsets between patches are always the same. Obviously, the patches extracted from an image will not be identical to the patches extracted from the same image after applying a rotation, specifically because the patches are extracted along the vertical and horizontal axes of the image based on a fixed grid, as illustrated in Figure 2. As such, the clusters may not contain the very same patches when the image is rotated, but in principle, each cluster should capture about the same information, since the distance between patches is always preserved. Therefore, the method can be safely considered as translation and rotation invariant. The final step is to find a representation for each of these clusters. The mean and the standard deviation are computed for each component of the similarity vectors within a group. Finally, the Translation and Rotation Invariant Patch Autocorrelation Features (TRIPAF) are obtained by concatenating these statistics for all the clusters C_1, C_2, ..., C_d, in this specific order, where d is a constant integer value that determines the number of clusters. Figure 2 illustrates the entire process required for computing the TRIPAF vector.

It is important to note that in order to make sure all the images in a set $\mathcal{I}$ are represented by vectors of the same length, each image needs to be resized such that its diagonal is equal to the constant integer value d:

$$\sqrt{{w_I}^2 + {h_I}^2} = d, \forall I \in \mathcal{I}, \tag{5}$$

where w_I and h_I are the width and the height of image I, respectively. The number of components of the TRIPAF vector is $O(md)$, where m represents the number of

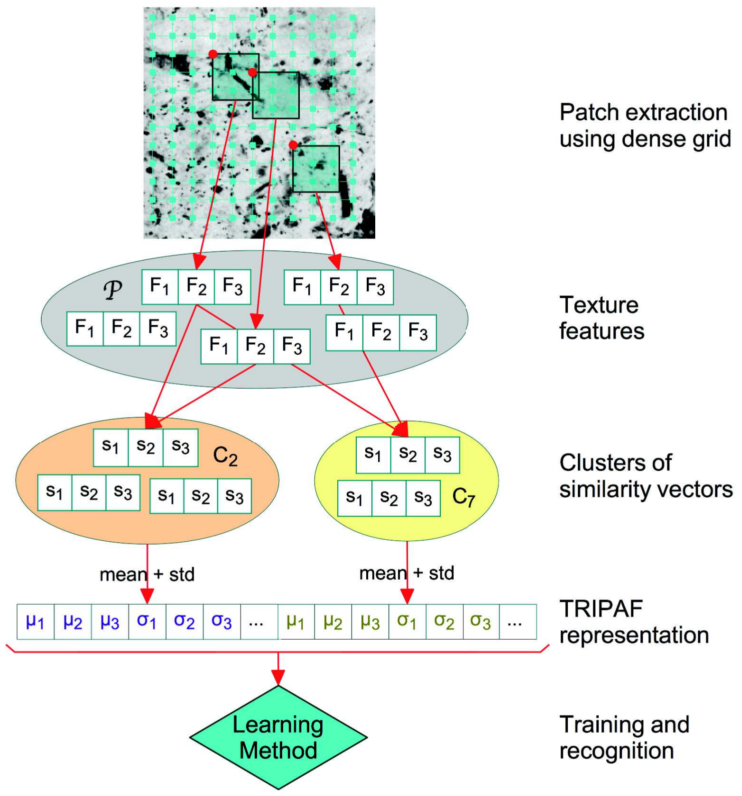


Fig. 2 The classification system based on Translation and Rotation Invariant Patch Autocorrelations Features. In a first stage, feature vectors are computed from patches extracted by applying a grid over the image, and then, the resulted feature vectors are stored in a set $\mathcal{P}$. Then, similarity vectors are computed and subsequently clustered according to the spatial offsets between patches. Finally, the TRIPAF vector is generated by computing the mean (μ) and the standard deviation (σ) of each component of the similarity vectors within each cluster. The TRIPAF maps of train images are then used to learn discriminant features. The trained classifier can be used to predict class labels for new images represented as TRIPAF vectors. For better readability, not all arrows have been represented in this figure.

features extracted from patches and d is a positive integer value that controls the number of pairs of patches per cluster. Actually, d can be considered as a parameter of TRIPAF that can be usually adjusted in practice, in order to obtain a desired trade-off between accuracy and space. Having an extra parameter that needs to be adjusted based on empirical observations is not necessarily desired. Therefore, to reduce the number of parameters that need to be tuned, d can be set to the average diagonal size of all the images in $\mathcal{I}$. An interesting detail about TRIPAF is that the clusters of similarity vectors naturally tend to become more and more sparse as the spatial offset between patches grows. Certainly, cluster C_1 is the most dense cluster, while C_d contains only two components that correspond to the patches situated in the four corners of the image. Being so far apart, the patches that form the sparse clusters no longer bring any useful information about the image.

This hypothesis was empirically tested in the context of texture classification. The accuracy of TRIPAF remains at the same level when the representation is reduced to half of its size by removing the sparse clusters $C_d, C_{d-1}, ..., C_{\lfloor d/2 \rceil}$. Another constant value $c \in (0, 1]$ is introduced to control the proportion of dense clusters that is taken into consideration. More precisely, only the first $\lfloor c \cdot d \rceil$ clusters will be kept in the final TRIPAF vector.

Algorithm 2: TRIPAF Algorithm

```
1  Input:
2  $I$ - a gray-scale input image of $h \times w$ pixels;
3  $p$ - the size (in pixels) of each square-shaped patch;
4  $s$ - the distance (in pixels) between consecutive patches;
5  $d$ - the number of clusters;
6  $c$ - a constant value in the range $(0, 1]$;
7  $\mathcal{F} = \{F_1, F_2, ...., F_m \mid F_i : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}, \forall i\}$ - a set of $m$ feature extraction functions.
8  Initialization:
9  $I \leftarrow$ resized image $I$ such that $\sqrt{(w)^2 + (h)^2} = d$;
10 $(h, w) \leftarrow size(I)$;
11 $n \leftarrow ceil((h - p + 1)/s) \cdot ceil((w - p + 1)/s)$;
12 $\mathcal{P} \leftarrow \emptyset$;
13 $C_i \leftarrow \emptyset$, for $i \in \{1, 2, ..., \lfloor c \cdot d \rceil\}$;
14 $v_i \leftarrow 0$, for $i \in \{1, 2, ..., m \cdot \lfloor c \cdot d \rceil\}$;
15 Computation:
16 for $i = 1 : s : h$ do
17     for $j = 1 : s : w$ do
18         $P \leftarrow I_{i:(i+p-1),j:(j+p-1)}$;
19         for $k = 1 : m$ do
20             $f_P(k) \leftarrow F_i(P)$;
21         $\mathcal{P} \leftarrow \mathcal{P} \cup f_P$;
22 for $i = 1 : n - 1$ do
23     for $j = i + 1 : n$ do
24         if $\lfloor o(P_i, P_j) \rceil \leq \lfloor c \cdot d \rceil$ then
25             for $k = 1 : m$ do
26                 $s_{P_i,P_j}(k) \leftarrow \sqrt{f_{P_i}(k)} \cdot \sqrt{f_{P_j}(k)}$;
27             $C_{\lfloor o(P_i,P_j) \rceil} \leftarrow C_{\lfloor o(P_i,P_j) \rceil} \cup s_{P_i,P_j}$;
28 $k \leftarrow 1$;
29 for $i = 1 : \lfloor c \cdot d \rceil$ do
30     for $j = 1 : m$ do
31         $v(k) \leftarrow mean(s(j))$, for $s \in C_i$;
32         $v(k + 1) \leftarrow std(s(j))$, for $s \in C_i$;
33         $k \leftarrow k + 2$;
34 Output:
35 $v$ - the TRIPAF feature vector with $m \cdot \lfloor c \cdot d \rceil$ components.
```

The TRIPAF representation is computed as described in Algorithm 2. The same mathematical conventions and notations provided in Section 3 are also used in Algorithm 2. On top of these notations, two predefined functions are used to compute the mean and the standard deviation (as defined in literature), namely *mean* and *std*. The TRIPAF algorithm can be divided into three phases. In the first phase (steps 16 -21), feature vectors are computed from patches extracted by applying a grid of uniformly spaced interest points, and then, the resulted feature vectors are stored in the set $\mathcal{P}$. The notation $\mathcal{P}$ is used by analogy to Algorithm 1, even if it is now used to store feature vectors instead of patches. In the second phase (steps 22 -27), the similarity vectors are computed and subsequently clustered according to the spatial offsets between patches. In the third phase (steps 28 -33), the TRIPAF vector v is generated by computing the mean and the standard deviation of each component of the similarity vectors within each cluster. A close view reveals that Algorithm 2 is more complex than Algorithm 1, but it is also more flexible, in the sense that it can easily be adapted for a variety of image classification tasks, simply by changing the set of features $\mathcal{F}$. The set of texture-specific features used for the texture classification experiments are described next.

4.1 Texture Features

The mean and the standard deviation are the first two statistical features extracted from image patches. These two basic features can be computed indirectly, in terms of the image histogram. The shape of an image histogram provides many clues to characterize the image, but the features obtained from the histogram are not always adequate to discriminate textures, since they are unable to indicate local intensity differences. Therefore, more complex features are necessary.

One of the most powerful statistical methods for textured image analysis is based on features extracted from the Gray-Level Co-Occurrence Matrix (GLCM) proposed by Haralick et al (1973). The GLCM is a second order statistical measure of image variation and it gives the joint probability of occurrence of gray levels of two pixels, separated spatially by a fixed vector distance. Smooth texture gives a co-occurrence matrix with high values along diagonals for small distances. The range of gray level values within a given image determines the dimensions of a co-occurrence matrix. Thus, 4 bits gray level images give 16×16 co-occurrence matrices. Relevant statistical features for texture classification can be computed from a GLCM. The features proposed by Haralick et al (1973), which show a good discriminatory power, are the contrast, the energy, the entropy, the homogeneity, the variance and the correlation.

Another feature that is relevant for texture analysis is the fractal dimension. It provides a statistical index of complexity comparing how detail in a fractal pattern changes with the scale at which it is measured. The fractal dimension is usually approximated. The most popular method of approximation is box counting (Falconer, 2003). The idea behind the box counting dimension is to consider grids at different scale factors over the fractal image, and count how many boxes are filled over each grid. The box counting dimension is computed by estimating how this number changes as the grid gets finer, by applying a box counting algorithm. An efficient box counting algorithm for estimating the fractal dimension was proposed by Popescu et al (2013). The idea of the algorithm is to skip the computation for coarse grids, and count how many boxes are filled only for finer grids. The TRIPAF approach includes this efficient variant of box counting.

Daugman (1985) found that cells in the visual cortex of mammalian brains can be modeled by Gabor functions. Thus, image analysis by the Gabor functions is similar to perception in the human visual system. A set of Gabor filters with different frequencies and orientations may be helpful for extracting useful features from an image. The local isotropic phase symmetry measure (LIPSyM) of Kuse et al (2011) takes the discrete Fourier transform of the input image, and filters this frequency information through a bank of Gabor filters. Kuse et al (2011) also note that local responses of each Gabor filter can be represented in terms of energy and amplitude. Thus, Gabor features, such as the mean-squared energy and the mean amplitude, can be computed through the phase symmetry measure for a bank of Gabor filters with various scales and rotations. These features are relevant because Gabor filters have been found to be particularly appropriate for texture representation and discrimination. The Gabor features are also used in the TRIPAF algorithm.

5 Learning Methods

There are many supervised learning methods that can be used for classification, such as Naïve Bayes classifiers (Manning et al, 2008), neural networks (Bishop, 1995; Krizhevsky et al, 2012; LeCun et al, 2015), Random Forests (Breiman, 2001), kernel methods (Shawe-Taylor and Cristianini, 2004) and many others (Caruana and Niculescu-Mizil, 2006). However, all classification systems used in this work rely on kernel methods to learn discriminant patterns. Therefore, a brief overview of the kernel functions and kernel classifiers used either for handwritten digit recognition or texture classification is given in this section. For the texture recognition experiments, the TRIPAF representation is combined with a bag of visual words representation to improve performance, particularly the bag of visual words representation described in (Ionescu et al, 2014b), which is based on the PQ kernel. Details about the BOVW model are also given in this section.

5.1 Kernel Methods

Kernel-based learning algorithms work by embedding the data into a Hilbert space, and searching for linear relations in that space using a learning algorithm. The embedding is performed implicitly, that is by specifying the inner product between each pair of points rather than by giving their coordinates explicitly. The power of kernel methods lies in the implicit use of a Reproducing Kernel Hilbert Space (RKHS) induced by a positive semi-definite kernel function. Despite the fact that the mathematical meaning of a kernel is the inner product in a Hilbert space, another interpretation of a kernel is the pairwise similarity between samples.

The kernel function offers to the kernel methods the power to naturally handle input data that is not in the form of numerical vectors, such as strings, images, or even video and audio files. The kernel function captures the intuitive notion of similarity between objects in a specific domain and can be any function defined on the respective domain that is symmetric and positive definite. For images, many such kernel functions are used in various applications including object recognition, image retrieval, or similar tasks. Popular choices are the linear kernel, the intersection kernel, the Hellinger's kernel, the χ^2 kernel or the Jensen-Shannon kernel. Other state of art kernels are the pyramid match kernel of Lazebnik et al (2006) and the PQ kernel of Ionescu and Popescu (2013a). Some of these kernels are also used in different contexts in the experiments presented in this work, specifically the linear kernel, the intersection kernel and the PQ kernel. These three kernels are described in more detail next.

For two feature vectors $X, Y \in \mathbb{R}^n$, the linear kernel is defined as:

$$k(X, Y) = \langle X, Y \rangle, \tag{6}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product.

The intersection kernel is given by component-wise minimum of the two feature vectors:

$$k(X, Y) = \sum_{i}^{n} \min \{x_i, y_i\}. \tag{7}$$

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 $X, Y \in \mathbb{R}^n$ is:

$$P = |\{(i, j) \, : \, 1 \leq i < j \leq n, \, (x_i - x_j)(y_i - y_j) > 0\}|. \tag{8}$$

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

$$Q = |\{(i, j) \, : \, 1 \leq i < j \leq n, \, (x_i - x_j)(y_i - y_j) < 0\}|. \tag{9}$$

Among the correlation statistics based on counting concordant and discordant pairs are *Goodman and Kruskal's gamma*, *Kendall's tau-a* and *Kendall's tau-b* (Upton and Cook, 2004). These three correlation statistics are very related. More precisely, all are based on the difference between P and Q, normalized differently. In the same way, the PQ kernel (Ionescu and Popescu, 2013a) between two histograms X and Y is defined as:

$$k_{PQ}(X, Y) = 2(P - Q). \tag{10}$$

The PQ kernel can be computed in $O(n \log n)$ time with the efficient algorithm based on merge sort described in (Ionescu and Popescu, 2015). An important remark is that the PQ kernel is suitable to be used when the feature vector can also be described as a histogram. For instance, the PQ kernel improves the accuracy in object recognition (Ionescu and Popescu, 2013a, 2015) and texture classification (Ionescu et al, 2014b) when the images are represented as histograms of visual words.

After embedding the features with a kernel function, a linear classifier is used to select the most discriminant features. Various kernel classifiers differ in the way they learn to separate the samples. In the case of binary classification problems, Support Vector Machines (SVM) (Cortes and Vapnik, 1995) try to find the vector of weights that defines the hyperplane that maximally separates the images in the Hilbert space of the training examples belonging to the two classes. The SVM is a binary classifier, but image classification tasks are typically multi-class classification problems. There are many approaches for combining binary classifiers to solve multi-class problems. Usually, the multi-class problem is broken down into multiple binary classification problems using common decomposing schemes such as one-versus-all and one-versus-one. In the experiments presented in this paper the one-versus-all scheme is adopted for SVM. Nevertheless, there are some classifiers that take the multi-class nature of the problem directly into account, such as Kernel Discriminant Analysis (KDA). The KDA method provides a projection of the data points to a one-dimensional subspace where the Bayes classification error is smallest. It is able to improve accuracy by avoiding the class masking problem (Hastie and Tibshirani, 2003). Further details about kernel methods can be found in (Shawe-Taylor and Cristianini, 2004).

5.2 Bag of Visual Words Model

In computer vision, the BOVW 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.

The BOVW 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 a new image. The entire process, that involves both training and testing stages, is illustrated in Figure 3.

The feature detection and representation step works as described next. Features are detected using a regular grid across the input image. At each interest point, a SIFT feature (Lowe, 1999) is computed. This approach is known as dense SIFT (Dalal and Triggs, 2005; Bosch et al, 2007). 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 (Leung and Malik, 2001), and visual words are stored in a randomized forest of k-d trees (Philbin et al, 2007) to reduce search cost. The frequency of each visual word in an image is then recorded in a histogram. The histograms of visual words enter the training step. Typically, a kernel method is used for training the model. In the experiments, the recently introduced PQ kernel is used to embed the features into a Hilbert space. The learning is done either with SVM or KDA, which are both described in Section 5.1.

6 Handwritten Digit Recognition Experiments

Isolated handwritten character recognition has been extensively studied in the literature (Suen et al, 1992; Srihari, 1992), and was one of the early successful applications of neural networks (LeCun et al, 1989). Comparative experiments on recognition of individual handwritten digits are reported in (LeCun et al, 1998). While recognizing individual digits is one of many problems involved in designing a practical recognition system, it is an excellent benchmark for comparing shape recognition methods.

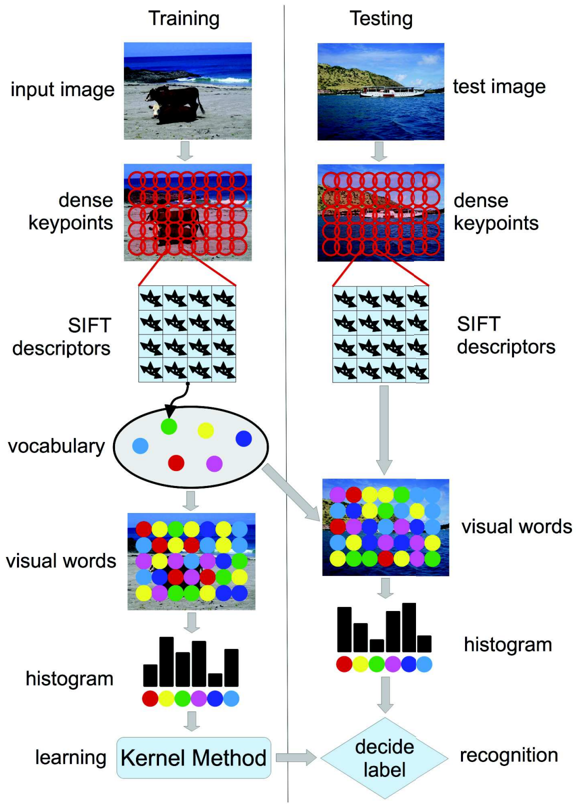


Fig. 3 The BOVW learning model for image classification. The feature vector consists of SIFT features computed on a regular grid across the image (dense SIFT) and vector quantized into visual words. The frequency of each visual word is then recorded in a histogram. The histograms enter the training stage. Learning is done by a kernel method.

6.1 Data Set Description

The data set used for testing the feature representation presented in this paper is the MNIST set, which is described in detail in (LeCun et al, 1998). The regular MNIST database contains $60,000$ train samples and $10,000$ test samples, size-normalized to 20×20 pixels, and centered by center of mass in 28×28 fields. A random sample of 15 images from this data set is presented in Figure 4. The data set is available at http://yann.lecun.com/exdb/mnist.

6.2 Implementation Details

The PAF representation is used in a learning context in order to evaluate its performance on handwritten digit recognition. Certainly, two learning methods based on the PAF representation are evaluated to provide a more clear overview of the performance improvements brought by PAF and to demonstrate that the improvements are not due to a specific learning method. The first classifier, that is

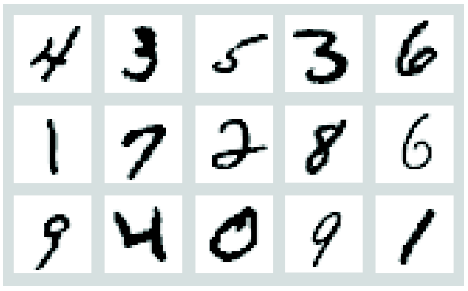


Fig. 4 A random sample of 15 handwritten digits from the MNIST data set.

intensively used through all the experiments, is the k-Nearest Neighbors (k-NN). The k-NN classifier was chosen because it directly reflects the characteristics of the PAF representation, since there is no actual training involved in the k-NN model. A state-of-the-art kernel method is also used in the experiments, namely the SVM based on the linear kernel.

It is important to note that the TRIPAF representation is not used in the handwritten digit recognition experiments, because it is not suitable for this task. Since TRIPAF is rotation invariant, it will certainly produce nearly identical representations for some samples of digits 6 and 9, or similarly for some samples of digits 1 and 7. This means that a classifier based on the TRIPAF representation will not be able to properly distinguish these digits, consequently producing a lower accuracy rate.

6.3 Organization of Experiments

Several classification experiments are conducted using the 3-NN and the SVM classifiers based on the PAF representation. These are systematically compared with two benchmark classifiers, specifically a 3-NN model based on the euclidean distance between pixels and a SVM trained on the raw pixel representation. The experiments are organized as follows. First of all, the two parameters of PAF and the regularization parameter of SVM are tuned in a set of preliminary experiments performed on the first 300 images from the MNIST training set. Another set of experiments are performed on the first 1000 images from the MNIST training set. These subsets of MNIST contain randomly distributed digits from 0 to 9 produced by different writers. The experiments on these subsets are performed using a 10-fold cross validation procedure. The procedure is repeated 10 times and the obtained accuracy rates are averaged for each representation and each classifier. This helps to reduce the variation introduced by the random selection of samples in each fold, and ensures a fair comparison between results. It is worth mentioning that the subset of 1000 images is used to assess the robustness of the evaluated methods when less training data is available. Usually, the accuracy degrades when there is less training data available, but a robust system should maintain its accuracy level as high as possible. Nevertheless, the classifiers are also evaluated on the entire MNIST data set. Each experiment is repeated using deslanted digits,

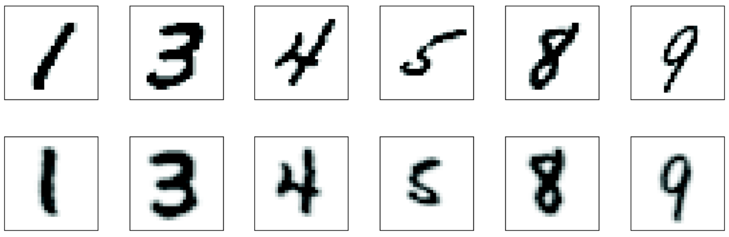


Fig. 5 A random sample of 6 handwritten digits from the MNIST data set before and after deslanting. The original images are shown on the top row and the slant corrected images are on the bottom row. A Gaussian blur was applied on the deslanted images to hide pixel distortions introduced by the deslanting technique.

which was previously reported to improve the recognition accuracy. The method used for deslanting the digits is described in Section 6.5.

6.4 Parameter Tuning

A set of preliminary tests on the first 300 images of the MNIST training set are performed to adjust the parameters of the PAF representation, such as the patch size and the pixel interval used to extract the patches. Patch sizes ranging from 1×1 to 8×8 pixels were considered. The best results in terms of accuracy were obtained with patches of 5×5 pixels, but patches of 4×4 or 6×6 pixels were also found to work well. In the rest of the experiments, the PAF representation is based on patches of 5×5 pixels.

After setting the desired patch size, the focus is to test different grid densities. Obviously, the best results in terms of accuracy are obtained when patches are extracted using an interval of 1 pixel. However, the goal of adjusting the grid density is to obtain a desired trade-off between accuracy and speed. Girds with intervals of 1 to 10 pixels were evaluated. The results indicate that the accuracy does not drop significantly when the pixel interval is less than the patch size. Consequently, a choice was made to extract patches at every 3 pixels to favor accuracy while significantly reducing the size of the PAF representation. Given that the MNIST images are 28×28 pixels wide, a grid with a density of 1 pixel generates 165.600 features, while a grid with the chosen density of 3 pixels generates only 2.016 features, which is roughly 80 times smaller.

The regularization parameter C of SVM was adjusted on the same subset of 300 images. The best results were obtained with $C = 10^2$ and, consequently, the rest of the results are reported using $C = 10^2$ for the SVM.

6.5 Deslanting Digits

A good way of improving the recognition performance is to process the images before extracting the features in order to reduce the amount of pattern variations within each class. As mentioned in (Teow and Loe, 2002), a common way of reducing the amount of variation is by deslanting the individual digit images. The

Table 1 Accuracy rates on the subset of 1000 images for the MNIST data set for the PAF representation versus the standard representation based on raw pixel data. The results are reported with two different classifiers using the 10-fold cross validation procedure. The results obtained on the original images are shown on the left-hand side and the results obtained on the deslanted images are shown on the right-hand side.

Features	Method	Original	Deslanted
Standard	3-NN	86.97%	91.60%
PAF	3-NN	**90.65%**	**93.42%**
Standard	SVM	86.21%	92.34%
PAF	SVM	**93.88%**	**95.38%**

deslanting method described in (Teow and Loe, 2002) was also adopted in this work and it is briefly described next. For each image, the least-squares regression line passing through the center of mass of the pixels is computed in the first step. Then, the image is skewed with respect to the center of mass, such that the regression line becomes vertical. Since the skewing technique may distort the pixels, a slight Gaussian blur is applied after skewing. A few sample digits before and after deslanting are presented in Figure 5.

6.6 Experiment on 1000 MNIST Images

In this experiment, the PAF representation is compared with the representation based on raw pixel data using two classifiers. First of all, a 3-NN classifier based on the PAF representation is compared with a baseline k-NN classifier. The 3-NN based on the euclidean distance measure (L_2-norm) between input images is the chosen baseline classifier. In (LeCun et al, 1998) an error rate of 5.00% on the regular test set with $k = 3$ for this classifier is reported. Other studies (Wilder, 1998) report an error rate of 3.09% on the same experiment. The experiment was recreated in this work, and an error rate of 3.09% was obtained. Second of all, a SVM classifier based on the PAF representation is compared with a baseline SVM classifier. All the tests are conducted on both original and deslanted images.

Table 1 shows the accuracy rates averaged on 10 runs of the 10-fold cross validation procedure. The results reported in Table 1 indicate that the PAF representation improves the accuracy over the standard representation. However, the PAF representation does not equally improve the accuracy rates for original versus deslanted images. On the original images, the PAF representation improves the accuracy of the 3-NN classifier by 3.68% and the accuracy of the SVM classifier by roughly 7.67%. In the same time, the PAF representation improves the accuracy of the 3-NN classifier by almost 2% and the accuracy of the SVM classifier by roughly 3% on the deslanted digit images. The best accuracy (95.38%) is obtained by the SVM based on the PAF map on deslanted images. First of all, the empirical results presented in Table 1 show that the PAF representation is better than the standard representation for the digit recognition task. Second of all, the PAF representation together with the deslanting technique further improve the results. In conclusion, there is strong evidence that the Patch Autocorrelation Features provide a better representation for the digit recognition task.

Table 2 Accuracy rates on the full MNIST data set for the PAF representation versus the standard representation based on raw pixel data. The results are reported on the official MNIST test set of 10,000 images. The results obtained on the original images are shown on the left-hand side and the results obtained on the deslanted images are shown on the right-hand side.

Features	Method	Original	Deslanted
Standard	3-NN	96.91%	98.11%
PAF	3-NN	**97.64%**	**98.42%**
Standard	SVM	92.24%	94.40%
PAF	SVM	**98.64%**	**98.93%**

6.7 Experiment on the Full MNIST Data Set

The results presented in the previous experiment look promising, but the PAF vector should be tested on the entire MNIST data set for a strong conclusion of its performance level. Consequently, the 3-NN and the SVM classifiers based on the PAF representation are compared on the full MNIST data set with the 3-NN and the SVM classifiers based on the feature representation given by raw pixels values. The results are reported in Table 2.

As other studies have reported (Wilder, 1998), an error rate of 3.09% is obtained by the baseline 3-NN model based on the euclidean distance. The 3-NN model based on the PAF representation gives an error rate of 2.36%, which represents an improvement lower than 1%. The two 3-NN models have lower error rates on deslanted images, proving that the deslanting technique is indeed helpful. The baseline 3-NN shows an improvement of 1.2% on the deslanted images. The PAF representation brings an improvement of 0.31% on the deslanted digits. Compared to the results reported in the previous experiments, the PAF representation does not have a great impact on the performance of the k-NN model. However, there are larger improvements to the SVM classifier. The PAF representation improves the accuracy of the SVM classifier by 6.4% on the original images, and by 4.53% on the deslanted images. Overall, the PAF representation seems to have a significant positive effect on the performance of the evaluated learning methods. The lowest error rate on the official MNIST test set (1.07%) is obtained by the SVM based on Patch Autocorrelation Features. This performance is similar to those reported by state-of-the-art models such as virtual SVM (DeCoste and Schölkopf, 2002), boosted stumps (Kégl and Busa-Fekete, 2009), or deep Boltzmann machines (Salakhutdinov and Hinton, 2009). In conclusion, the PAF representation can boost the performance of the 3-NN and the SVM up to state-of-the-art accuracy levels on the handwritten digit recognition task, with almost no extra time required. Indeed, it takes 20 milliseconds on average to compute the PAF representation for a single MNIST image on a machine with Intel Core i7 2.3 GHz processor and 8 GB of RAM using a single Core. Hence, the method can even be used in real-time applications.

7 Texture Classification Experiments

Texture classification is a difficult task, in which methods have to take into account several aspects such as translation, rotation, and scale variations, illumination

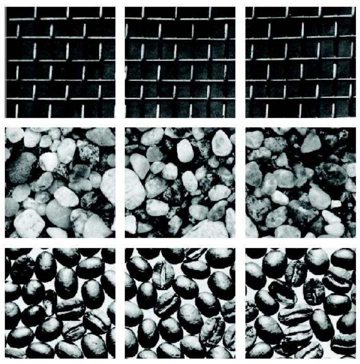


Fig. 6 Sample images from three classes of the Brodatz data set.

changes, viewpoint changes, and non-rigid distortions. Thus, texture classification is an adequate task to prove that a particular method is robust to such image variations. Indeed, the main goal of the texture classification experiments presented in this section is to demonstrate that TRIPAF is invariant to several image transformations, including translation and rotation changes, and that it can yield good results on this rather difficult image classification task.

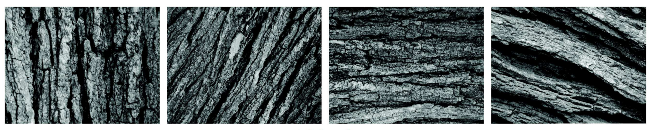
7.1 Data Sets Description

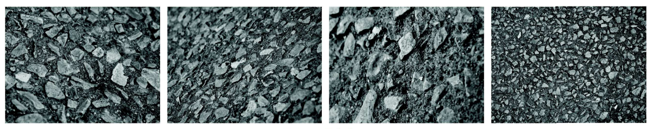
Texture classification experiments are presented on two benchmark data sets of texture images. The first experiment is conducted on the Brodatz data set (Brodatz, 1966). This data set is probably the best known benchmark used for texture classification, but also one of the most difficult, since it contains 111 classes with only 9 samples per class. The standard procedure in the literature is to obtain samples of 213×213 pixels by cutting them out from larger images of 640×640 pixels using a 3 by 3 grid. Figure 6 presents three sample images per class of three classes randomly selected from the Brodatz data set.

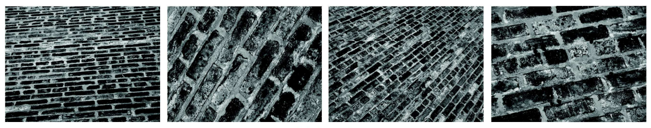
The second experiment is conducted on the UIUCTex data set of Lazebnik et al (2005). It contains 1000 texture images of 640×480 pixels representing different types of textures such as bark, wood, floor, water, and more. There are 25 classes of 40 texture images per class. Textures are viewed under significant 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 7 presents four sample images per class of four classes representing bark, brick, pebbles, and plaid.

7.2 Implementation Details

The TRIPAF representation is compared with a method that extracts texture-specific features from entire images, on the texture classification task. Among the GLCM features that show a good discriminatory power, only four of them are used


(a) bark


(b) pebbles


(c) brick

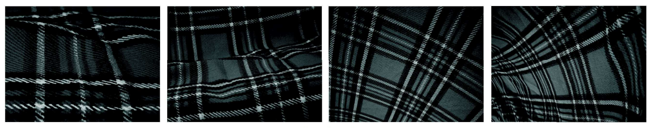

(d) plaid

Fig. 7 Sample images from four classes of the UIUCTex data set. Each image is showing a textured surface viewed under different poses.

by the classification systems, namely the contrast, the energy, the homogeneity, and the correlation. These GLCM features are averaged on 4 directions (vertical, horizontal and diagonals) using gaps of 1 and 2 pixels. The mean and the standard deviation are also added to the set of texture-specific features. Another feature is given by the box counting dimension. The Gabor features (the mean-squared energy and the mean amplitude) are computed on 3 scales and 6 different rotations, resulting in a total of 36 Gabor features (2 features $\times$ 3 scales $\times$ 6 directions). There are 43 texture-specific features put together. Alternatively, the two Gabor features are also averaged on 3 scales and 6 different rotations, generating only 4 Gabor features. This produces a reduced set of 9 texture-specific features, which is more robust to image rotations and scale variations. Results are reported using both representations, one of 43 features and the other of 9 features.

The parameter d that determines the number clusters in the TRIPAF approach was set to the average diagonal size of the images in each data set. Without any tuning, the parameter c was set to 0.5 in order to reduce the size of the TRIPAF map by half. As a consequence, all the TRIPAF representations have between 300 and 1500 components.

The TRIPAF approach, which is rotation and translation invariant, is combined with the BOVW model, which is scale invariant (due to the SIFT descriptors), in order to obtain a method that is invariant to all affine transformations. These methods are combined at the kernel level through *multiple kernel learning* (MKL) (Gonen and Alpaydin, 2011). When multiple kernels are combined, the features are actually embedded in a higher-dimensional space. As a consequence, the search space of linear patterns grows, which helps the classifier to select a better discriminant function. The most natural way of combining two kernels is to sum them up, and this is exactly the MKL approach used in this paper. Summing up kernels in the dual form is equivalent to feature vector concatenation in the primal form. The combined representation is alternatively used with the SVM or the KDA. The two classifiers are compared with other state-of-the-art approaches (Zhang et al, 2007; Nguyen et al, 2011; Quan et al, 2014), although Quan et al (2014) do not report results on the Brodatz data set.

7.3 Parameter Tuning

A set of preliminary tests on a subset of 40 classes from Brodatz are performed to adjust the parameters of the PAF representation, such as the patch size, and the pixel interval used to extract the patches. Patches of 16×16, 32×32 and 64×64 pixels were considered. Better results in terms of accuracy were obtained with either patches of 16×16 or 32×32 pixels, and the rest of the experiments are based on patches of such dimensions. The TRIPAF representations generated by patches of 16×16 or 32×32 pixels are also combined by summing up their kernels to obtain a more robust representation.

After setting up the patch sizes, the next goal is to adjust the grid density. The grid density was chosen such that the processing time of TRIPAF is less than 5 seconds per image on a machine with Intel Core i7 2.3 GHz processor and 8 GB of RAM using a single Core. Notably, more than 90% of the time is required to extract the texture-specific features from every patch. For the Brodatz data set, patches are extracted at an interval of 8 pixels, while for the UIUCTex data set, patches are extracted at every 16 pixels.

The regularization parameter C of SVM was adjusted on the same subset of of 40 classes from Brodatz. The best results were obtained with $C = 10^4$ and, consequently, the rest of the results are reported using $C = 10^4$ for the SVM. In a similar fashion, the regularization parameter of KDA was set to 10^{-5}.

7.4 Experiment on Brodatz Data Set

Typically, the results reported in previous studies (Lazebnik et al, 2005; Zhang et al, 2007; Nguyen et al, 2011) on the Brodatz data set are based on randomly selecting 3 training samples per class and using the rest for testing. Likewise, the results presented in this paper are based on the same setup with 3 random samples per class for training. Moreover, the random selection procedure is repeated for 20 times and the resulted accuracy rates are averaged. This helps to reduce the amount of accuracy variation introduced by using a different partition of the data set in each of the 20 trials. To give an idea of the amount of variation in each trial, the standard deviations for the computed average accuracy rates are also reported. The evaluation procedure described so far is identical to the one used in the state-of-the-art approaches (Zhang et al, 2007; Nguyen et al, 2011) that are included in the following comparative study.

Table 3 presents the accuracy rates of the SVM classifier based on several TRIPAF representations, in which the number of texture-specific features and

Table 3 Accuracy rates of the SVM classifier on the Brodatz data set for the TRIPAF representation versus the standard representation based on texture-specific features. The reported accuracy rates are averaged on 20 trials using 3 random samples per class for training and the other 6 for testing. The best accuracy rate for each set of texture-specific features is highlighted in bold.

Feature Map	Texture Features	Patches	Kernel	Accuracy
Standard	9	none	linear	$76.52\% \pm 1.6$
Standard	9	none	intersection	$77.11\% \pm 1.3$
TRIPAF	9	16×16	intersection	$87.78\% \pm 1.2$
TRIPAF	9	32×32	intersection	$91.40\% \pm 0.9$
TRIPAF	9	$16 \times 16 + 32 \times 32$	intersection	$\mathbf{91.92}\% \pm 0.6$
Standard	43	none	linear	$89.93\% \pm 1.1$
Standard	43	none	intersection	$90.42\% \pm 1.2$
TRIPAF	43	16×16	intersection	$92.11\% \pm 0.8$
TRIPAF	43	32×32	intersection	$92.28\% \pm 0.9$
TRIPAF	43	$16 \times 16 + 32 \times 32$	intersection	$\mathbf{92.85}\% \pm 0.8$

Table 4 Accuracy rates of the TRIPAF and BOVW combined representation on the Brodatz data set compared with state-of-the-art methods. The TRIPAF representation is based on 43 texture-specific features extracted from patches of 16×16 and 32×32 pixels. The BOVW model is based on the PQ kernel. The reported accuracy rates are averaged on 20 trials using 3 random samples per class for training and the other 6 for testing. The best accuracy rate is highlighted in bold.

Model	Accuracy
SVM based on BOVW (Ionescu et al, 2014b)	$92.94\% \pm 0.8$
SVM based on TRIPAF	$92.85\% \pm 0.8$
SVM based on TRIPAF + BOVW	$96.24\% \pm 0.6$
KDA based on TRIPAF + BOVW	$\mathbf{96.51}\% \pm 0.7$
Best model of Zhang et al (2007)	$95.90\% \pm 0.6$
Best model of Nguyen et al (2011)	$96.14\% \pm 0.4$

the size of the patches are varied. Several baseline SVM classifiers, based on the same texture-specific features used in the TRIPAF representation, are included in the evaluation in order to estimate the performance gain offered by TRIPAF. When the set of 9 texture-specific features is being used, the TRIPAF approach improves the baseline by more than 10% in terms of accuracy. On the other hand, the difference is roughly 2% in favor of TRIPAF when the set of 43 texture-specific features is being used. Both the standard and the TRIPAF representations work better when more texture-specific features are extracted, probably because the Brodatz data set does not contain significant rotation changes within each class of images. Nevertheless, the TRIPAF approach is always able to give better results than the baseline SVM. An interesting remark is that the results of TRIPAF are always better when patches of 16×16 pixels are used in conjunction with patches of 32×32 pixels, even if the accuracy improvement over using them individually is not considerable (below 1%). The best accuracy (92.85%) is obtained by the SVM based on the TRIPAF approach that extracts 43 features from patches of 16×16 and 32×32 pixels.

The empirical results presented in Table 3 clearly demonstrate the advantage of using the TRIPAF feature vectors. Intuitively, further combining TRIPAF with BOVW should yield even better results. While TRIPAF is rotation and translation

Table 5 Accuracy rates of the SVM classifier on the UIUCTex data set for the TRIPAF representation versus the standard representation based on texture-specific features. The reported accuracy rates are averaged on 20 trials using 20 random samples per class for training and the other 20 for testing. The best accuracy rate for each set of texture-specific features is highlighted in bold.

Feature Map	Texture Features	Patches	Kernel	Accuracy
Standard	9	none	linear	$78.23\% \pm 1.7$
Standard	9	none	intersection	$79.87\% \pm 1.3$
TRIPAF	9	16×16	intersection	$92.60\% \pm 1.1$
TRIPAF	9	32×32	intersection	$92.38\% \pm 1.1$
TRIPAF	9	$16 \times 16 + 32 \times 32$	intersection	$\mathbf{93.54\% \pm 1.1}$
Standard	43	none	linear	$82.19\% \pm 1.6$
Standard	43	none	intersection	$83.62\% \pm 1.4$
TRIPAF	43	16×16	intersection	$91.40\% \pm 1.2$
TRIPAF	43	32×32	intersection	$90.91\% \pm 1.2$
TRIPAF	43	$16 \times 16 + 32 \times 32$	intersection	$\mathbf{91.82\%} \pm 1.0$

invariant, BOVW is scale invariant, and therefore, these two representations complement each other perfectly. Table 4 compares the results of TRIPAF and BOVW combined through MKL with the results of two state-of-the-art methods (Zhang et al, 2007; Nguyen et al, 2011). The intersection kernel used in the case of TRIPAF is summed up with the PQ kernel used in the case of BOVW (Ionescu et al, 2014b). The individual results of TRIPAF and BOVW are also listed in Table 4. The two methods obtain fairly similar accuracy rates when used independently, but the accuracy rates are almost 3% lower than the state-of-the-art methods. However, the kernel combination of TRIPAF and BOVW yields results comparable to the state-of-the-art methods (Zhang et al, 2007; Nguyen et al, 2011). In fact, the best accuracy rate on the Brodatz data set (96.51%) is given by the KDA based on TRIPAF and BOVW, although the SVM based on the kernel combination is also slightly better than both state-of-the-art models. The kernel sum of TRIPAF and BOVW is much better than using the two representations individually, proving that the idea of combining them up is indeed crucial for obtaining state-of-the-art results.

7.5 Experiment on UIUCTex Data Set

As in the previous experiment, the standard evaluation procedure for the UIUCTex data set is used to assess the performance of approach proposed in this work. More precisely, the samples are split into a training set and a test set by randomly selecting 20 samples per class for training and the remaining 20 samples for testing. The random selection process is repeated for 20 times and the resulted accuracy rates are averaged over the 20 trials. Table 5 reports the average accuracy rates along with their standard deviations which give some indication about the amount of variation in each trial. In Table 5, the SVM classifiers based on various TRIPAF representations are compared with baseline SVM classifiers that extract the same texture-specific features used in the TRIPAF representation, but from entire images instead of patches. For each set of texture-specific features, the TRIPAF approach improves the baseline by roughly 10%. It can be observed that the improvement is higher for the set of 9 texture-specific features, going up by

Table 6 Accuracy rates of the TRIPAF and BOVW combined representation on the UIUCTex data set compared with state-of-the-art methods. The TRIPAF representation is based on 9 texture-specific features extracted from patches of 16 × 16 and 32 × 32 pixels. The BOVW model is based on the PQ kernel. The reported accuracy rates are averaged on 20 trials using 20 random samples per class for training and the other 20 for testing. The best accuracy rate is highlighted in bold.

Model	Accuracy
SVM based on BOVW (Ionescu et al, 2014b)	$91.74\% \pm 1.4$
SVM based on TRIPAF	$93.54\% \pm 1.1$
SVM based on TRIPAF + BOVW	$97.75\% \pm 0.7$
KDA based on TRIPAF + BOVW	$98.31\% \pm 0.5$
Best model of Zhang et al (2007)	$\mathbf{98.70}\% \pm 0.9$
Best model of Nguyen et al (2011)	$97.84\% \pm 0.3$
Pattern Lacunarity Spectrum (Quan et al, 2014)	96.57%

13% in terms of accuracy. Perhaps surprisingly, the results of TRIPAF are better when the lighter patch representation is being used. The set of 9 texture-specific features is more robust to rotation and scale variations, and this kind of changes are more prominent in the UIUCTex data set. This could probably explain why the lighter patch representation produces better results. The overall results presented in Table 5 are consistent with the results on the Brodatz data set shown in Table 3. Indeed, the TRIPAF approach is always able to give better results than the SVM baseline. As in the Brodatz experiment, the results of TRIPAF are better when patches of 16 × 16 pixels are used together with patches of 32 × 32 pixels. However, if the patches are used individually, it seems that using 16 × 16 patches is a better choice. The best accuracy (93.54%) is obtained by the SVM based on the TRIPAF approach that extracts 9 features from patches of 16 × 16 and 32 × 32 pixels. To conclude, the best TRIPAF representation along with the other TRIPAF representations bring significant improvements over the baseline SVM systems.

Next, the TRIPAF representation is combined with the BOVW representation of Ionescu et al (2014b) and compared with three state-of-the-art methods (Zhang et al, 2007; Nguyen et al, 2011; Quan et al, 2014) using the same evaluation procedure. The results of this comparative study are presented in Table 6. To better understand why is it important to combine TRIPAF and BOVW, the individual results of the two feature representations are also included in Table 6. Notably, the accuracy rate of the BOVW model (91.74%) is roughly 2% lower than the accuracy rate of the TRIPAF approach (93.54%), probably because there are more rotation and translation changes than scale variations in the UIUCTex data set. Certainly, it can be easily observed that the two approaches work much better when they are combined through MKL. The kernel combination of TRIPAF and BOVW is at least 4% better than each of the individual components. As in the previous experiment, KDA yields slightly better results than SVM, although they are both comparable with the state-of-the-art methods. More specifically, the SVM based on the kernel combination obtains similar performance to the best model of Nguyen et al (2011), while KDA is only 0.4% lower than the best model of Zhang et al (2007). Nevertheless, the difference between the accuracy rates of the last four approaches listed in Table 6 is within the range designated by the standard deviations. Interestingly, both SVM and KDA based on the kernel combination of

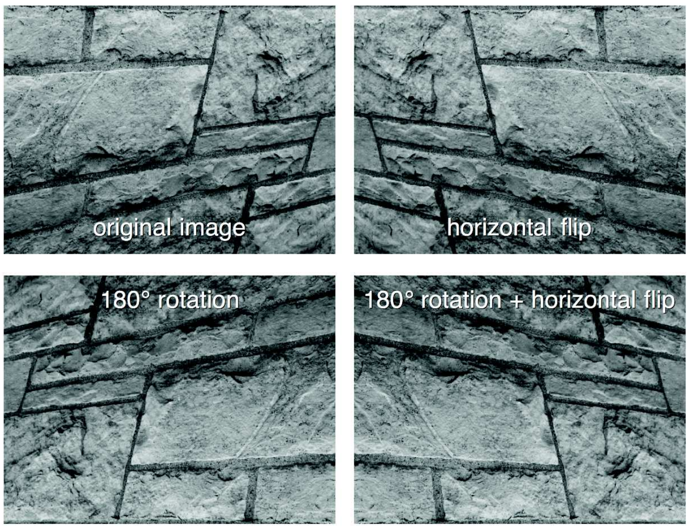


Fig. 8 A sample image from the of UIUCTex data set shown under four different rotations. The top left image is the original version. The others are obtain by applying 180 degrees rotation and horizontal flip on the original.

TRIPAF and BOVW are better than the Pattern Lacunarity Spectrum of Quan et al (2014). Using MKL for texture classification proves to be again an efficient way to improve performance.

7.6 Empirical Proof of Rotation Invariance

Since the UIUCTex data set contains significant image variations including rotations, one concern is to demonstrate that TRIPAF is rotation invariant. In this context, three different rotations are applied on each image of the UIUCTex data set, while also keeping the original images. First, a copy of UIUCTex data set is created by rotating the original images by 180 degrees. A second copy of the original images is obtained by flipping them horizontally, which corresponds to a mirroring effect. The third and last copy is obtained by first rotating the original images by 180 degrees and then by flipping them horizontally. Essentially, four versions (including the original) are obtained from each image, as illustrated in Figure 8. In the end, 4000 images are obtained by applying all these rotations.

Next, the intersection kernel matrix is computed from the TRIPAF feature vectors that correspond to all the 4000 images. The best TRIPAF representation in the UIUCTex classification experiment was used, more precisely, the one based on 9 texture-specific features extracted from patches of 16×16 and 32×32 pixels. As defined in literature (Shawe-Taylor and Cristianini, 2004), the kernel matrix for n samples is the matrix $K \in \boldsymbol{\mathcal{M}}_{n,n}$ of all possible pairwise similarities between the corresponding feature vectors. In this experiment, the intersection kernel as defined in Equation (7) is used to compute the similarity of each two samples. The resulted kernel matrix of 4000×4000 components is illustrated in Figure 9.

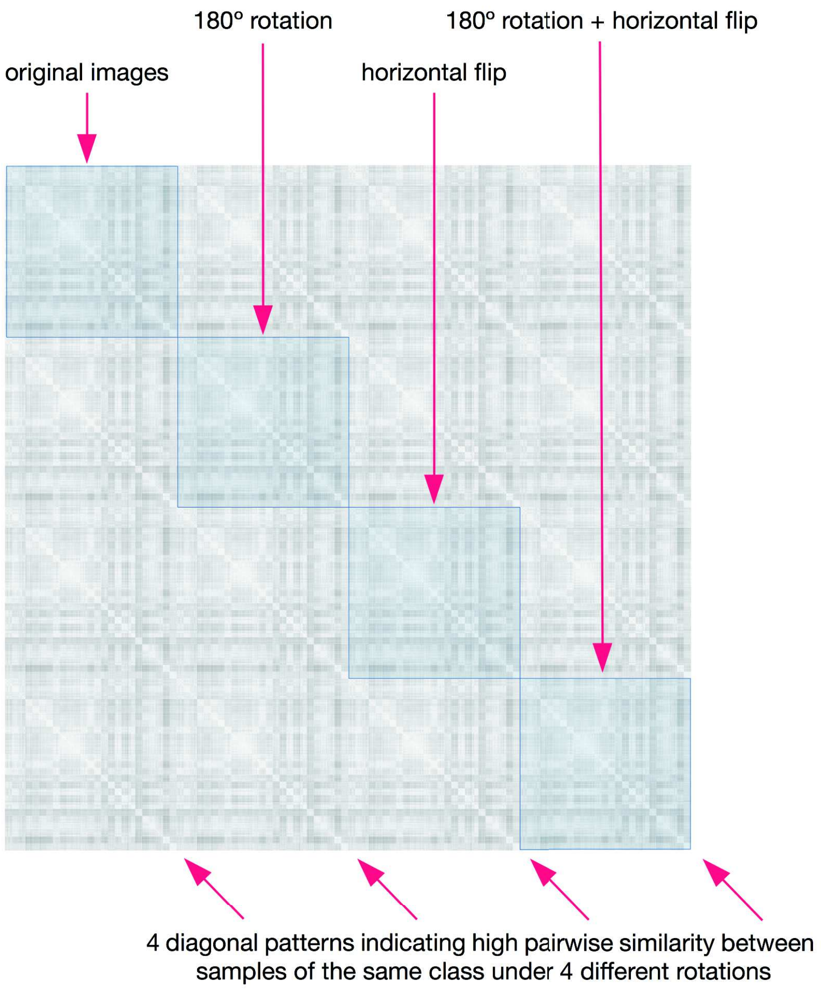


Fig. 9 The intersection kernel matrix based on TRIPAF features computed on the UIUCTex images given under four different rotations. Light shades of gray correspond to high pairwise similarities and dark shades correspond to low pairwise similarities. The diagonal patterns indicate that the TRIPAF representation is rotation invariant.

The normalized similarity stored in a matrix cell is a value between 0 and 1. In Figure 9, the higher similarity values (closer to 1) are represented by lighter shades of gray, while the lower similarity values (closer to 0) are represented by darker shades of gray, in a space of 256 shades of gray.

To produce the kernel matrix, the images are considered in a specific order that gives the means for some *interesting patterns*[2] to emerge in the matrix. First, images are grouped according to the four types of rotation changes. Certainly, the

[2] In this context, interesting refers to patterns that can be visually interpreted with the naked eye.

first 1000 images are the original ones from UIUCTex, the next 1000 images are the ones rotated by 180 degrees, and so on. Within each of these groups of 1000 unique images, the images are again grouped into smaller subgroups according to their classes. Consequently, the small light gray squares along the main diagonal (there are 100 of them, as there are 25 classes and 4 rotations) represent the within class similarity. The fact that these are lighter in color indicates that images within each class are more similar to each other and less similar to the images from other classes.

The four highlighted blocks along the main diagonal represent the similarity between samples with the same kind of rotation. For each type of rotation, there is a block of 1000×1000 pairwise similarities. These blocks appear to be identical, which means that the similarity between the images remains invariant when these are obtained by applying a certain kind of rotation on all the images. This pattern can be expressed mathematically as follows:

$$K_{i,j} \simeq K_{i+r\cdot 1000, j+r\cdot 1000}, \forall i,j \in \{1,...,1000\}, \forall r \in \{1,2,3\}, \tag{11}$$

where K is the intersection kernel matrix illustrated in Figure 9. Furthermore, the entire kernel matrix seems to be constructed by repeating the same block (of 1000×1000 pairwise similarities) for 16 times (4 times in each of the two directions). This pattern can be expressed mathematically as follows:

$$K_{i,j} \simeq K_{i+r\cdot 1000, j+q\cdot 1000}, \forall i,j \in \{1,...,1000\}, \forall r,q \in \{0,1,2,3\}. \tag{12}$$

Equation (12) indicates that for any two images (at indexes i and j) in the UIUCTex data set, the similarity between them is almost the same, no matter which of the four kinds of considered rotations is applied to each of the two images. This property of the intersection kernel matrix was put to the test. Remarkably, Equation (12) holds for all $i,j \in \{1,...,1000\}$ and all $r,q \in \{0,1,2,3\}$ with an approximation error lower than 10^{-6}. Given that $K_{i,j}$ can be any real value between 0 and 1, the error is indeed very small. Hence, it is safe to consider that the pairwise similarities are almost completely invariant to the four types of rotations, which translates to the fact that the TRIPAF representation is rotation invariant. The diagonal patterns formed by the little light gray squares (parallel to the main diagonal) that can be observed in Figure 9 show that the intraclass pairwise similarities are higher than interclass pairwise similarities (which are represented by darker shades of gray), even when a different rotation is applied to each image of a given class. Therefore, it can be concluded that the TRIPAF representation is invariant to significant rotation changes and even mirroring effects.

8 Conclusion

This work presented two feature representations for images inspired by the autocorrelation. The first representation, termed Patch Autocorrelation Features, extracts patches by applying a grid over the image, then records the similarities between all pairs of patches in a vector, also referred to as the PAF representation. Despite of being successfully used for handwritten digit recognition, the PAF approach is not invariant to affine transformations, thus having limited applications.

The second representation, termed Translation and Rotation Invariant Patch Autocorrelation Features, is an extension of the PAF representation that is designed to be invariant to image transformations, such as translations and rotations. In the TRIPAF approach, a set of texture-specific features are extracted from each image patch. Based on these features, a vector of similarities is computed between each pair of patches. The resulting similarity vectors are quantized according to the spatial offset between the patches of each corresponding pair. In the end, the mean and the standard deviation of each similarity value are computed for each cluster of similarity vectors and stored in the TRIPAF feature vector.

The PAF approach was evaluated in a series of handwritten digit recognition experiments using the popular MNIST data set. In these experiments, the PAF representation improved the accuracy rate of the learning methods (k-NN and SVM), a fact that indicates that the Patch Autocorrelation Features provide a robust and consistent approach of boosting the recognition performance with almost no extra time required. In a similar way, the TRIPAF approach was evaluated in a set of texture classification experiments that require the use of invariant techniques to obtain good performance. The TRIPAF representation consistently improved the accuracy rate over a baseline model based on extracting texture-specific features from entire images. Moreover, the TRIPAF approach was combined with a BOVW model (Ionescu et al, 2014b) through MKL. The kernel combination of TRIPAF and BOVW reached accuracy levels comparable and sometimes better than three state-of-the-art texture classification methods (Zhang et al, 2007; Nguyen et al, 2011; Quan et al, 2014).

By evaluating PAF and TRIPAF in several experiments on various data sets, statistical evidence was collected in order to eliminate any doubts with respect to the danger of overfitting. In every experiment, the data sets were always separated into different training and test sets, and regularization was employed as a mean to control overfitting on the training set. Hence, the uncertainty that PAF and TRIPAF would yield equally good image classification results on other data sets (not considered in the present study) is greatly reduced.

It is interesting to note that the TRIPAF representation is more compact and can be computed more efficiently than state-of-the-art representations such as BOVW (Bosch et al, 2007; Ionescu et al, 2014b) or Fisher Vectors (Perronnin et al, 2010). Indeed, both these state-of-the-art models need to cluster the descriptors extracted from all the images, and this step requires a considerable amount of time. Furthermore, BOVW needs thousands of visual words, while Fisher Vectors have a quadratic dependence on the number of visual words, and thus, can even go up to a million features. By contrast, the empirical results presented in this work indicate that TRIPAF works well with only a few hundred features.

In future work, the TRIPAF approach can be used for other image classification tasks such as object recognition or facial expression recognition. Depending on the task, a more suitable set of features to be extracted from patches could be discovered and used to improve performance.

Acknowledgements The authors thank the reviewers for their helpful comments. Andreea Lavinia Ionescu has been funded by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Ministry of European Funds through the Financial Agreement POSDRU/159/1.5/S/134398.

References

Agarwal S, Roth D (2002) Learning a Sparse Representation for Object Detection. Proceedings of ECCV pp 113–127

Barnes C, Goldman DB, Shechtman E, Finkelstein A (2011) The PatchMatch Randomized Matching Algorithm for Image Manipulation. Communications of the ACM 54(11):103–110

Belongie S, Malik J, Puzicha J (2002) Shape matching and object recognition using shape contexts. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(4):509–522

Bishop CM (1995) Neural Networks for Pattern Recognition. Oxford University Press, Inc., New York, NY, USA

Bosch A, Zisserman A, Munoz X (2007) Image Classification using Random Forests and Ferns. Proceedings of ICCV pp 1–8

Breiman L (2001) Random forests. Machine Learning 45(1):5–32, DOI 10.1023/A:1010933404324

Brochard J, Khoudeir M, Augereau B (2001) Invariant feature extraction for 3D texture analysis using the autocorrelation function. Pattern Recognition Letters 22(6-7):759–768, DOI http://dx.doi.org/10.1016/S0167-8655(01)00015-0

Brodatz P (1966) Textures: a photographic album for artists and designers. Dover pictorial archives, Dover Publications, New York, USA

Caruana R, Niculescu-Mizil A (2006) An empirical comparison of supervised learning algorithms. In: Proceedings of the 23rd international conference on Machine learning, New York, NY, USA, ICML '06, pp 161–168

Cho TS, Avidan S, Freeman WT (2010) The patch transform. IEEE Transactions on Pattern Analysis and Machine Intelligence 32(8):1489–1501

Ciresan DC, Meier U, Schmidhuber J (2012) Multi-column Deep Neural Networks for Image Classification. Proceedings of CVPR pp 3642–3649

Cortes C, Vapnik V (1995) Support-Vector Networks. Machine Learning 20(3):273–297

Csurka G, Dance CR, Fan L, Willamowski J, Bray C (2004) Visual categorization with bags of keypoints. In Workshop on Statistical Learning in Computer Vision, ECCV pp 1–22

Dalal N, Triggs B (2005) Histograms of Oriented Gradients for Human Detection. Proceedings of CVPR 1:886–893

Daugman JG (1985) Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. Journal of the Optical Society of America A 2(7):1160–1169

DeCoste D, Schölkopf B (2002) Training Invariant Support Vector Machines. Machine Learning 46(1–3):161–190, DOI 10.1023/A:1012454411458

Deselaers T, Keyser D, Ney H (2005) Discriminative Training for Object Recognition using Image Patches. Proceedings of CVPR pp 157–162

Dinu LP, Ionescu R, Popescu M (2012) Local Patch Dissimilarity for Images. Proceedings of ICONIP 7663:117–126

Efros AA, Freeman WT (2001) Image quilting for texture synthesis and transfer. Proceedings of SIGGRAPH '01 pp 341–346

Falconer K (2003) Fractal Geometry: Mathematical Foundations and Applications, 2nd edn. Wiley

Gonen M, Alpaydin E (2011) Multiple Kernel Learning Algorithms. Journal of Machine Learning Research 12:2211–2268

Guo G, Dyer CR (2007) Patch-based Image Correlation with Rapid Filtering. Proceedings of CVPR

Haouas F, Dhiaf ZB, Solaiman B (2016) Fusion of spatial autocorrelation and spectral data for remote sensing image classification. Proceedings of ATSIP pp 537–542

Haralick RM, Shanmugam K, Dinstein I (1973) Textural Features for Image Classification. IEEE Transactions on Systems, Man and Cybernetics 3(6):610–621

Hastie T, Tibshirani R (2003) The Elements of Statistical Learning, corrected edn. Springer

Horikawa Y (2004a) Comparison of support vector machines with autocorrelation kernels for invariant texture classification. Proceedings of ICPR 1:660–663, DOI 10.1109/ICPR.2004.1334253

Horikawa Y (2004b) Use of Autocorrelation Kernels in Kernel Canonical Correlation Analysis for Texture Classification. Proceedings of ICONIP 3316:1235–1240

Ionescu RT, Popescu M (2013a) Kernels for Visual Words Histograms. Proceedings of ICIAP 8156:81–90

Ionescu RT, Popescu M (2013b) Speeding Up Local Patch Dissimilarity. Proceedings of ICIAP 8156:1–10

Ionescu RT, Popescu M (2015) PQ kernel: a rank correlation kernel for visual word histograms. Pattern Recognition Letters 55:51–57, DOI http://dx.doi.org/10.1016/j.patrec.2014.06.003

Ionescu RT, Popescu M (2016) Knowledge Transfer between Computer Vision and Text Mining. Advances in Computer Vision and Pattern Recognition, Springer International Publishing

Ionescu RT, Popescu M, Grozea C (2013) Local Learning to Improve Bag of Visual Words Model for Facial Expression Recognition. Workshop on Challenges in Representation Learning, ICML

Ionescu RT, Popescu AL, Popescu D, Popescu M (2014a) Local Texton Dissimilarity with Applications on Biomass Classification. Proceedings of VISAPP

Ionescu RT, Popescu AL, Popescu M (2014b) Texture Classification with the PQ Kernel. Proceedings of WSCG

Ionescu RT, Popescu AL, Popescu D (2015a) Patch Autocorrelation Features for Optical Character Recognition. Proceedings of VISAPP

Ionescu RT, Popescu AL, Popescu D (2015b) Texture Classification with Patch Autocorrelation Features. Proceedings of ICONIP 9489:1–11

Ionescu RT, Popescu M, Cahill A (2016) String kernels for native language identification: Insights from behind the curtains. Computational Linguistics 42(3):491–525

Kameyama K, Phan TNB (2013) Image Feature Extraction and Similarity Evaluation Using Kernels for Higher-Order Local Autocorrelation. Proceedings of ICONIP 2013 pp 442–449

Kégl B, Busa-Fekete R (2009) Boosting Products of Base Classifiers. Proceedings of ICML pp 497–504, DOI 10.1145/1553374.1553439

Keysers D, Deselaers T, Gollan C, Ney H (2007) Deformation Models for Image Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 29(8):1422–1435, DOI 10.1109/TPAMI.2007.1153

Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet Classification with Deep Convolutional Neural Networks. Proceedings of NIPS pp 1106–1114

Kuse M, Wang YF, Kalasannavar V, Khan M, Rajpoot N (2011) Local isotropic phase symmetry measure for detection of beta cells and lymphocytes. Journal of Pathology Informatics 2(2):2

Laalaoui Y, Bouguila N (eds) (2015) Artificial Intelligence Applications in Information and Communication Technologies. Springer International Publishing

Lazebnik S, Schmid C, Ponce J (2005) A Sparse Texture Representation Using Local Affine Regions. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(8):1265–1278

Lazebnik S, Schmid C, Ponce J (2006) Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. Proceedings of CVPR 2:2169–2178

LeCun Y, Jackel LD, Boser B, Denker JS, Graf HP, Guyon I, Henderson D, Howard RE, Hubbard W (1989) Handwritten digit recognition: Applications of neural net chips and automatic learning. IEEE Communications pp 41–46

LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11):2278–2324

LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444

Leung T, Malik J (2001) Representing and Recognizing the Visual Appearance of Materials using Three-dimensional Textons. International Journal of Computer Vision 43(1):29–44

Liu L, Fieguth P, Kuang G, Zha H (2011) Sorted Random Projections for robust texture classification. Proceedings of ICCV pp 391–398, DOI 10.1109/ICCV.2011.6126267

Lowe DG (1999) Object Recognition from Local Scale-Invariant Features. Proceedings of ICCV 2:1150–1157

Manning CD, Raghavan P, Schütze H (2008) Introduction to Information Retrieval. Cambridge University Press, New York, NY, USA

Michaeli T, Irani M (2014) Blind deblurring using internal patch recurrence. Proceedings of ECCV pp 783–798

Nguyen HG, Fablet R, Boucher JM (2011) Visual textures as realizations of multivariate log-Gaussian Cox processes. Proceedings of CVPR pp 2945–2952

Paredes R, Prez-Cortes J, Juan A, Vidal E (2001) Local Representations and a Direct Voting Scheme for Face Recognition. Proceedings of Workshop on Pattern Recognition in Information Systems pp 71–79

Passino G, Izquierdo E (2007) Patch-based image classification through conditional random field model. Proceedings of the International Conference on Mobile Multimedia Communications pp 6:1–6:6

Perronnin F, Dance CR (2007) Fisher kernels on visual vocabularies for image categorization. Proceedings of CVPR

Perronnin F, Sánchez J, Mensink T (2010) Improving the fisher kernel for large-scale image classification. Proceedings of ECCV pp 143–156

Philbin J, Chum O, Isard M, Sivic J, Zisserman A (2007) Object retrieval with large vocabularies and fast spatial matching. Proceedings of CVPR pp 1–8

Popescu AL, Popescu D, Ionescu RT, Angelescu N, Cojocaru R (2013) Efficient Fractal Method for Texture Classification. Proceedings of ICSCS

Popovici V, Thiran J (2001) Higher order autocorrelations for pattern classification. Proceedings of ICIP 3:724–727, DOI 10.1109/ICIP.2001.958221

Quan Y, Xu Y, Sun Y, Luo Y (2014) Lacunarity analysis on image patterns for texture classification. Proceedings of CVPR pp 160–167

Salakhutdinov R, Hinton GE (2009) Deep boltzmann machines. Proceedings of AISTATS pp 448–455

Shawe-Taylor J, Cristianini N (2004) Kernel Methods for Pattern Analysis. Cambridge University Press

Simard P, LeCun Y, Denker JS, Victorri B (1996) Transformation Invariance in Pattern Recognition, Tangent Distance and Tangent Propagation. Neural Networks: Tricks of the Trade

Simonyan K, Zisserman A (2014) Very Deep Convolutional Networks for Large-Scale Image Recognition. CoRR abs/1409.1556

Socher R, Huval B, Bath B, Manning C, Ng A (2012) Convolutional-Recursive Deep Learning for 3D Object Classification. Proceedings of NIPS pp 665–673

Srihari SN (1992) High-performance reading machines. Proceedings of the IEEE (Special issue on Optical Character Recognition) 80(7):1120–1132

Suen CY, Nadal C, Legault R, Mai TA, Lam L (1992) Computer recognition of unconstrained handwritten numerals. Proceedings of the IEEE (Special issue on Optical Character Recognition) 80(7):1162–1180

Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going Deeper With Convolutions. Proceedings of CVPR

Szeliski R (2010) Computer Vision: Algorithms and Applications, 1st edn. Springer-Verlag New York, Inc., New York, NY, USA

Teow LN, Loe KF (2002) Robust vision-based features and classification schemes for off-line handwritten digit recognition. Pattern Recognition 35(11):2355–2364, DOI http://dx.doi.org/10.1016/S0031-3203(01)00228-X

Toyoda T, Hasegawa O (2007) Extension of higher order local autocorrelation features. Pattern Recognition 40(5):1466–1473, DOI http://dx.doi.org/10.1016/j.patcog.2006.10.006

Upton G, Cook I (2004) A Dictionary of Statistics. Oxford University Press, Oxford

Valipour M (2015a) Long-term runoff study using SARIMA and ARIMA models in the United States. Meteorological Applications 22(3):592–598

Valipour M (2015b) Optimization of neural networks for precipitation analysis in a humid region to detect drought and wet year alarms. Meteorological Applications 23(1):91–100

Valipour M, Banihabib ME, Behbahani SMR (2013) Comparison of the ARMA, ARIMA, and the autoregressive artificial neural network models in forecasting the monthly inflow of Dez dam reservoir. Journal of Hydrology 476:433–441

Wilder KJ (1998) Decision tree algorithms for handwritten digit recognition. Electronic Doctoral Dissertations for UMass Amherst

Yi S, Pavlovic V (2013) Spatio-temporal Context Modeling for BoW-Based Video Classification. Proceedings of ICCV Workshops pp 779–786

Zhang J, Marszalek M, Lazebnik S, Schmid C (2007) Local Features and Kernels for Classification of Texture and Object Categories: A Comprehensive Study. International Journal of Computer Vision 73(2):213–238