Pattern Recognition 45 (2012) 1396–1408

Contents lists available at SciVerse ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Learning pairwise image similarities for multi-classification using Kernel Regression Trees

Andrzej Ruta [a,*], Yongmin Li [b]

[a] AGH University of Science and Technology, Department of Computer Science, Al. Mickiewicza 30, 30-059 Krakow, Poland
[b] School of Information Systems, Computing and Mathematics, Brunel University, Uxbridge, Middlesex UB8 3PH, United Kingdom

ARTICLE INFO

Article history:
Received 24 April 2011
Received in revised form
5 September 2011
Accepted 7 September 2011
Available online 18 October 2011

Keywords:
Visual similarity
Learning from example pairs
Object recognition
Kernel Regression Trees

ABSTRACT

We are often faced with the problem of distinguishing between visually similar objects that share the same general appearance characteristics. As opposed to object categorization, this task is focused on capturing fine image differences in a pose-dependent fashion. Our research addresses this particular family of problems and is centered around the concept of learning from example pairs. Formally, we construct a parameterized visual similarity function optimally separating the pairs of images that depict the objects of the same class or identity from the pairs representing different object classes/identities. It combines various image distances that are quantified by comparing local descriptor responses at the corresponding locations in both paired images. To find the best combinations, we train ensembles of so-called *Kernel Regression Trees* which model the desired similarity function as a hierarchy of fuzzy decision stumps. The obtained function is then used within a *k-NN*-like framework to address complex multi-classification problems. Through the experiments with several image datasets we demonstrate the numerous advantages of the proposed approach: high classification accuracy, scalability, ease of interpretation and the independence of the feature representation.

1. Introduction

Visual object recognition is a difficult problem that has been addressed for many years. In the machine vision community there is a somewhat vague distinction between categorization and classification. The former attempts to determine the general category of an object or scene assumed that categories substantially differ from one another and that there is a significant appearance variability within each. If the above conditions hold, images can be conveniently represented by their salient feature signatures, e.g. quantized SIFT descriptors [1], which often provide decent discriminative potential. A whole range of methods capable of handling tasks of this kind have been proposed, e.g. those based on the part-based representation and generative modeling [2–5].

Vision-based classification, which is in the center of our interest, encompasses problems where the perceptual differences between objects may not be so obvious. In the same time a relatively low error rate of the classifier is anticipated (consider for instance an iris recognition system guarding access to key premises of a company). The generative approaches, so successful in content-based image retrieval, do not prove useful in this case.

As conventional discriminative techniques are usually only suitable for binary classification, many previous attempts to distinguishing similar objects focused on developing efficient multi-class extensions to known binary classifiers. For instance, Crammer and Singer [6] proposed a multi-class variant of SVM based on the extended notion of margin. Similarly, Hao and Luo [7] introduced an elegant generalization of the popular *AdaBoost* algorithm. Other solutions were based on *one-vs-one* or *one-vs-all* decomposition of the original multi-class problem [8–11].

1.1. Rationale behind the nearest neighbors family of methods

In certain application contexts class modeling may appear difficult, time-consuming or impractical. Scenarios where this holds include the case when the number of classes or distinct identities within a given object category is large compared to the volume of available training data, or when the notion of class/category is ambiguous. Consider, for instance, a recently popular LFW face database [12] with over 13,000 images depicting over 5700 individuals where, however, only 1680 persons appear in two or more photographs. On the other hand, the above mentioned ambiguity results from the fact that object naming is essentially a subjective matter, i.e. two visually similar objects can be labeled in a different way, often due to the semantic

* Corresponding author. Tel.: +48 12 6173491; fax: +48 12 6339406.
E-mail addresses: aruta@agh.edu.pl, mailme@aruta.pl (A. Ruta), Yongmin.Li@brunel.ac.uk (Y. Li).

0031-3203/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2011.09.028

inclusion of categories (e.g. building and house or vehicle and car). Conversely, the same object can be captured from different viewpoints, resulting in the images that have visually very little in common, despite consistent labeling used (e.g. a frontal and side view of a motorcycle). Also, sometimes it is simply irrelevant what exactly an observed object represents. Instead, what matters is which other object it resembles. In all such cases a common-sense strategy is to perform recognition via some sort of nearest prototype matching. Despite its simplicity, it has been proven to work surprisingly well, even for a large number of classes, e.g. in [13].

The other appealing property of the *Nearest Neighbors* family of methods is that they can accommodate different distance metrics or scale a fixed metric in appropriate directions to fine tune the classifier to a particular problem [14,15]. If elementary metrics fail to encode the domain-specific knowledge, a suitable metric can be learned automatically from the data. It has been done for instance by Nowak and Jurie [16] or Malisiewicz and Efros [17]. Besides, a hybrid approach is possible, i.e. the similarities to the closest prototypes may span a new feature space in which standard pattern recognition techniques are applied. The studies of Zhang et al. [18] or Paclík et al. [19] follow this direction. Feature distribution matching within an SVM framework proposed in [10] is another example.

1.2. Overview of the proposed approach

Inspired by several previous studies [16,17,20–24], we adopt the above mentioned strategy of handling complex classification problems via nearest prototype association. However, unlike in most recent approaches, we want to reliably discriminate between the essentially very similar object classes that: (1) are well-defined in terms of unambiguous naming and low intra-class appearance variations, and (2) belong to the same general category, e.g. faces. Therefore, we need to focus on often very subtle and localized appearance differences between them, which requires substantial alignment of the images being compared. Hence, the view independence aspect of object recognition is considered irrelevant for this study.

To reliably infer the object's identity from its nearest neighbors, we combine images into pairs labeled "same" or "different", depending on whether or not they depict the objects of the same class. Then, boosted ensembles of so-called *Kernel Regression Trees* (*SimKRTs*) are employed to learn the localized distances between the training pairs and combine these distances into robust discriminant functions. Their soft responses measure global image-to-image similarity which underlies a conventional *k-NN* rule. The latter is used to determine the most likely label of a novel image by comparing it to the prototypes of known identity. Adapting fuzzy regression tree framework to pairwise similarity metric learning is the major contribution of this paper.

The above approach makes our work follow the same line of thinking as that of Wolf et al. [24] or Frome et al. [22]. In both studies the image-to-image distance/similarity measure is estimated in a data-driven manner, i.e. such that the resulting function changes depending upon the input images. This is a departure from the body of metric learning work which are based on a single "global" distance measure—learned and then fixed.

Wolf et al. [24] used a so-called *One-Shot Similarity* score which for an input pair of images required an auxiliary set A of images having different labels than those being compared. The proposed similarity score was calculated by averaging the responses of a selected classifier, e.g. LDA or SVM, learned on either of the test images vs A and applied to the other image. Good results were reported for this method for a number of multi-class identification tasks. The main difference between our approach and the above cited work lies in the underlying distance learning framework and, primarily, in that we compare images directly (without recourse to any other images). It means that the parameters of the global similarity function are learned from a number of training pairs and then this function is simply evaluated on a novel pair.

This is what links the proposed method to that of Frome et al. [22]. Also, as in their work, the learning algorithm yields similarity functions that are consistent across images and hence applicable to the previously never seen objects. Here the main differences concern the learning framework (variant of a maximum-margin classifier and image triplets are used by the authors), the form of the target similarity function (our tree-based and non-linear vs their linear combination) and the types of problems addressed (Frome et al. focus on general object categorization).

A unique property of our algorithm is its independence of the image features and local distance metrics chosen, ease of interpretation, as well as its inherently parallel nature, which facilitates hardware acceleration. To provide justification of the above claims, we conduct a series of recognition experiments involving several image datasets, including traffic signs, car models and human faces.

The rest of this paper is composed as follows. In Section 2 the concept of learning similarity from image pairs is briefly outlined. Section 3 provides the theoretical background of *Kernel Regression Trees* and then discusses how they are used for visual similarity learning. Section 4 is devoted to the analysis of the experimental results we obtained with the proposed *SimKRT–k-NN* framework using various image datasets. Finally, conclusions of this work are drawn in Section 5.

2. Pairwise similarities in visual recognition

In this section we introduce the concept of learning object similarity from pairs of examples and discuss how it can be used in the machine vision domain to address complex multi-class classification problems. Learning from image pairs has previously been studied, e.g. by Jones and Viola [25], Athitsos et al. [21], Nowak and Jurie [16] or Wolf et al. [24].

2.1. Local distance and global similarity

Assume we have an image of an unknown object, I_j, belonging to the one of N classes, $C_1,\ldots,C_N$, each represented by a prototype image, I_{C_i}. Let also each image be represented by some number of feature descriptors, r_k, where each r_k can be any function of the image, unrestricted in terms of the type and the dimensionality of the value returned. To be able to classify an unknown image, we need to define two distance/similarity measures.

The local distance between two images, I_1, I_2, with respect to a descriptor r_k is generically defined as

$$d_k(I_1,I_2)=\phi(r_k(I_1),r_k(I_2)), \tag{1}$$

where function ϕ takes the values returned for both input images by the chosen image descriptor and yields a scalar output. The global similarity between the images is denoted by $S(I_1,I_2)$ and can be any scalar-valued function of I_1 and I_2. In this work we are interested in the global similarity functions based on the local distances, i.e.

$$S(I_1,I_2)=F(d_1(I_1,I_2),\ldots,d_n(I_1,I_2)). \tag{2}$$

One possible realization of F is a non-linear function modeled by a fuzzy regression tree described in Section 3. Below, it is briefly discussed how a multi-class object recognition problem can be addressed based on the learned pairwise similarity function and which image descriptors are suitable for quantification of the local distances incorporated in this function.

Table 1
Local distance metrics associated with different image descriptors that were used in the experiments involving the proposed pairwise similarity learning framework.

Feature type	Output value	Distance formula
Haar filter [26]	Scalar v	$\lvert v_2-v_1\rvert$
HOG [29]	Vector $\mathbf{v}\in\mathbb{R}^n$	$\sqrt{\sum_{i=1}^{n}(v_{2,i}-v_{1,i})^2}$
Region covariance [28]	Matrix $C_{n\times n}$	$\sqrt{\sum_{i=1}^{n}\ln^2\lambda_i(C_1,C_2)}$, where $\{\lambda_i(C_1,C_2)\}_{i=1,\ldots,n}$ are the generalized eigenvalues of $\lambda C_1\mathbf{x}_i-C_2\mathbf{x}_i=0$ (see [28] for details)

2.2. Multi-classification via image pairing

Assume that an input image I_j has to be classified into one of N categories, $C_1,\ldots,C_N$. Employing our global similarity measure S, this multi-class problem can be solved by pairing I_j with the prototype image of each class, I_{C_i}, $i=1,\ldots,N$, and picking the label maximizing the similarity between them:

$$L(I_j)=\arg\max_i S(I_j,I_{C_i}). \tag{3}$$

Alternatively, any variant of the *k-Nearest Neighbors* scheme can be used. In this approach the test image is paired with a number of prototype images of each class and the desired identity is obtained through majority voting among the k closest prototypes. If two or more classes receive an equal number of votes, ambiguity is resolved by picking the label shared by the prototypes with the highest average similarity to the tested image. The main difference between the classical *k-NN* and the method adopted in this work is that only a small portion of available images are used for comparison to the query image. Moreover, an equal number of representative images of each class are ensured to belong to this *comparison set*.[1]

The choice of prototype images for the comparison set may vary. If there exists a representative idealized template for a given object class, such as a pictogram of a traffic sign, this template alone may be used as a prototype image. For a much broader range of problems a better approach is to pick several prototypes from among the available real-life images, either randomly or in a more methodical way, e.g. via clustering of the training data and selection of the images that are closest to the found modes. As a result, the possibly multi-modal distribution of the classes' appearance is accounted for, regardless of whether it is caused by real intra-class variations or the imperfections of the image acquisition process (varying illumination, image resolution, etc.). In addition, the computational cost of the classifier is hugely reduced, compared to the regular *k-NN* involving all available images.

What finally calls for explanation is the choice of feature descriptors r_k that underlie each local pairwise distance computation. These descriptors are to be found in a training process. The general idea behind it is to consider for all input image pairs a large (possibly over-complete) space of descriptors and then select those that, when used in Eq. (1), best separate the pairs representing the same class from the pairs representing two different classes. As a result, the target similarity function F incorporates only a small number of the most informative descriptors and the *k-NN*-like classifier utilizing F requires fewer computations when faced with a novel image. In this respect training is simultaneously aimed at finding the parameters of the visual similarity function and at dimensionality reduction.

In Table 1 three example image descriptors considered in this study have been characterized, together with the corresponding distance metrics. It is important to note that with the recent algorithmic developments made in visual feature extraction based on integral images [26–28], all three descriptors can be evaluated in a very efficient way.

[1] This term is used consistently throughout the rest of this paper.

3. Pairwise similarity based Kernel Regression Trees

Formally, our target classifier is designed to recognize only two classes: "same" and "different", and is trained using pairs of images, i.e. $\mathbf{x}=(I_1,I_2)$. The pairs representing the same object class are labeled $y=1$, and the pairs representing two different object classes are labeled $y=0$. Recall that our goal is to construct a real-valued discriminant function $F(\mathbf{x})$ separating the two classes. This function can be realized by a fuzzy regression tree induced within the learning framework proposed by Olaru and Wehenkel [30]. The approach is presented in detail in the next section.

The motivation behind choosing fuzzy regression trees [30–34] is primarily their simplicity, accuracy, and natural ability to handle conditional and uncertain information effectively. Conditional (i.e. hierarchical) reasoning seems especially close to how humans distinguish between globally very similar visual patterns that differ only with respect to subtle, spatially isolated details. Introduction of fuzzy split rules brings additional strengths to the tree framework, as shown in [30]. First, it decreases bias near the local decision boundary at each node. Second, it ensures continuity of the tree's output. Finally, it prevents the tree from overfitting by not allowing too fast decrease in the size of the local growing samples when going down into the tree. Ensemble techniques on the other hand are known to generally improve prediction accuracy and stability of the underlying classifiers [35].

What distinguishes the proposed tree framework from those used in previous studies is that it is fed with pairs of images and used for similarity estimation. As a result, if used within a *k-NN* scheme, response of a single tree (instead of N, as in [30]) suffices to discriminate between N classes. In the following sections the term "example" is used with the meaning of image pair.

3.1. Tree semantics

In our fuzzy regression tree each node D_i is assigned a numerical label $L_{D_i}\in[0,1]$ expressing its local estimation of the output, that is the degree of similarity of an input pair of images passing through that node. Each non-terminal node is additionally assigned a fuzzy discriminant function, $f_{D,k,\boldsymbol{\Theta}}(\mathbf{x})$, used to determine whether the tested example should be directed to the left ($f_{D,k,\boldsymbol{\Theta}}(\mathbf{x})=1$) and/or right subtree ($f_{D,k,\boldsymbol{\Theta}}(\mathbf{x})=0$). It is characterized by k, the index of the selected image descriptor and a vector of parameters, $\boldsymbol{\Theta}$. Function $f_{D,k,\boldsymbol{\Theta}}$, as well as label L_{D_i} are determined at the training stage.

With our specific meaning of $\mathbf{x}$, f becomes a function of the distance between two images with respect to the descriptor r_k

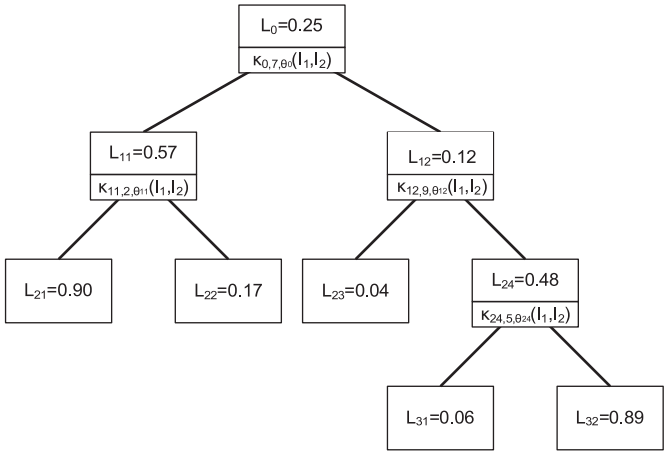

Fig. 1. An example fuzzy regression tree.

Table 2
Three possible kernel functions used to split the examples passing through each non-terminal node of a *Kernel Regression Tree*. The last two are fuzzy.

Kernel type	Formula
Step	$\kappa_{D,k}(I_1,I_2)=\begin{cases}1 & \text{if } d_k(I_1,I_2)<\alpha\\ 0 & \text{if } d_k(I_1,I_2)\geq\alpha\end{cases}$
Piecewise linear	$\kappa_{D,k}(I_1,I_2)=\begin{cases}1 & \text{if } d_k(I_1,I_2)<\alpha-\beta\\ \frac{\alpha+\beta-d_k(I_1,I_2)}{2\beta} & \text{if } d_k(I_1,I_2)\in[\alpha-\beta,\alpha+\beta]\\ 0 & \text{if } d_k(I_1,I_2)>\alpha+\beta\end{cases}$
Gaussian RBF	$\kappa_{D,k}(I_1,I_2)=e^{-\alpha d_k^2(I_1,I_2)}$

and, as such, acts as a kernel

$$f_{D,k,\Theta}(\mathbf{x})=f_D(d_k(I_1,I_2),\Theta)=\kappa_{D,k,\Theta}(I_1,I_2). \tag{4}$$

An example tree with four non-terminal and five terminal nodes is depicted in Fig. 1. For instance, the input pairs passing through the node labeled L_{12} are split by evaluating the kernel function based on the descriptor r_9.

In general, splitting is done in a soft way. It means that the examples are directed to both child nodes of a given node simultaneously, partitioning the local input space[2] into two overlapping subspaces where the degree of overlap depends on the type of kernel function and the parameter vector Θ. In the example tree from Fig. 1 an input image pair at node labeled L_{24} will be propagated to the nodes labeled L_{31} and L_{32} simultaneously whenever the response of the kernel $\kappa_{24,5,\Theta_{24}}$ is in between 0 and 1 exclusive. In order to retain the ease of interpretation of the hard-split trees,[3] only the monotonic kernel functions have been considered in this work. Two examples of such functions are characterized in Table 2.

The above definitions imply that the examples propagated to both successor nodes in parallel do not strictly belong to any of them. Instead, each example is assigned a degree of membership to each node D, $\mu_D(\mathbf{x}_j)\in[0,1]$. The degree of membership to the root node, $\mu_{Root}(\mathbf{x}_j)$, is by default set to unity, which reflects the fact that all examples fully belong to it. The degree of membership to the child nodes D_L and D_R of a given node D is defined recursively:

$$\mu_{D_L}(\mathbf{x}_j)=\mu_D(\mathbf{x}_j)\kappa_{D,k,\Theta}(\mathbf{x}_j),$$

$$\mu_{D_R}(\mathbf{x}_j)=\mu_D(\mathbf{x}_j)(1-\kappa_{D,k,\Theta}(\mathbf{x}_j)), \tag{5}$$

which means that the membership degree at a child node is at most as high as at the parent node and it is weighted by the response of the parent's kernel function. This gives an effect of "diffusion" of an example as it propagates through the tree's hierarchy. Finally, each example passed on input of the tree potentially follows multiple paths at once. Therefore, in order to calculate the tree's output for the input example, the local estimates at all terminal nodes have to be summed, taking into account the likelihood of the paths leading to each:

$$\hat{y}_j=\frac{\sum_{i\in\text{leaves}}\mu_{D_i}(\mathbf{x}_j)L_{D_i}}{\sum_{i\in\text{leaves}}\mu_{D_i}(\mathbf{x}_j)}. \tag{6}$$

To sum up, on the input pair's way down the tree multiple kernels are evaluated on various local image distances, the obtained responses are combined, and the tree outputs an estimation of the global similarity between the paired images (desired S from Eq. (2)). Therefore, we call our fuzzified regression tree a *Pairwise Similarity Based Kernel Regression Tree* (*SimKRT*). It should not be confused with that proposed by Torgo [36], also called *Kernel Regression Tree*. In this approach the prediction for a query was calculated as a weighted average of the target variable values of the most similar training examples residing at the terminal node that was reached by the query. A selected kernel function was used to quantify this similarity.

3.2. Training

Learning the *SimKRT* trees requires two independent sets of examples: growing set and pruning set. In a growing phase it is necessary to define: (1) a rule for automatic determination of the splitting descriptor, r_k, and the associated kernel parameters at each node, as well as the labels of the successors of this node and (2) a stopping criterion for the growing process. Below, both issues are discussed without providing all derivations.

Given a node D, our objective is to find the split feature and the kernel function's parameters that minimize the squared error function:

$$E_D=\sum_{\mathbf{x}\in D}w(\mathbf{x})\mu_D(\mathbf{x})(y(\mathbf{x})-y'(\mathbf{x}))^2, \tag{7}$$

where

$$y'(\mathbf{x})=\kappa_{D,k,\Theta}(\mathbf{x})L_{D_L}+(1-\kappa_{D,k,\Theta}(\mathbf{x}))L_{D_R}, \tag{8}$$

and $w(\mathbf{x})$ is a normalized weight of the training pair (by default all pairs' weights are equal).

The actual strategy for searching the minimum of the above error functional depends on the type of the kernel function used to split the examples passing through the considered node. Generally, if it depends on a single parameter, α, minimization of (8) is straightforward. To do this, all possible local distances d_k and the values of α parameter are considered. For a fixed image descriptor r_k used to calculate d_k and fixed α, we compute the derivatives of E_D with respect to L_{D_L} and L_{D_R}, and set them to zero. Solving the resulting linear system in L_{D_L} and L_{D_R}, we obtain the formulas for the optimal successor node labels:

$$L_{D_L}=\frac{CD-EB}{B^2-AC},\quad L_{D_R}=\frac{AE-BD}{B^2-AC}, \tag{9}$$

where

$$A=\sum_{\mathbf{x}\in D}w(\mathbf{x})\mu_D(\mathbf{x})\kappa_{D,k,\alpha}(\mathbf{x})^2,$$

[2] 1-dimensional space determined by the local distances between the input image pairs at a given node with respect to a single descriptor selected at this node.

[3] By a hard-split tree we understand a regression tree with step discriminant function at each node. Such trees introduce no fuzziness.

$$B = \sum_{\mathbf{x} \in D} w(\mathbf{x})\mu_D(\mathbf{x})\kappa_{D,k,\alpha}(\mathbf{x})\kappa'_{D,k,\alpha}(\mathbf{x}),$$

$$C = \sum_{\mathbf{x} \in D} w(\mathbf{x})\mu_D(\mathbf{x})\kappa'_{D,k,\alpha}(\mathbf{x})^2,$$

$$D = -\sum_{\mathbf{x} \in D} w(\mathbf{x})\mu_D(\mathbf{x})y(\mathbf{x})\kappa_{D,k,\alpha}(\mathbf{x}),$$

$$E = -\sum_{\mathbf{x} \in D} w(\mathbf{x})\mu_D(\mathbf{x})y(\mathbf{x})\kappa'_{D,k,\alpha}(\mathbf{x}), \tag{10}$$

and $\kappa'_{D,k,\alpha}(\mathbf{x})$ stands for $1-\kappa_{D,k,\alpha}(\mathbf{x})$. Introducing (9) in (8) gives a recipe for split error computation. Ultimately, the splitting image descriptor and the associated kernel parameter α minimizing this error are saved.

In the case of a piecewise linear kernel function, which is dependent on two parameters, the above error minimization strategy would have to consider all possible triples (r_k,α,β), which is computationally expensive. However, it can be shown that for fixed r_k the pursued global minimum of (7) as a function of four parameters, α, β, L_{D_L}, L_{D_R}, is generally very close to its minimum for any fixed β. Therefore, minimization of E_D can be performed sequentially. First, the optimal threshold α is found for $\beta=0$, as though the split was ideally step. Then, for the found value of α the best value of β is searched for in the same way as outlined above for the single-parameter kernels. In the case of more complex, multi-parameter kernel functions, numerical approximations to the minimization problem in (7) are inevitable.

The above tree growing process is recursive. We start with the root node by setting its label to the value equal to the fraction of positive examples in the growing set, i.e. the prior probability of the input example representing the "same" class. Also, all $\mu_{Root}(\mathbf{x})$ values are set to unity. Then, we search for the optimal splitting descriptor and kernel parameters of the root node, which produces two child nodes, D_{1L}, D_{1R}, labeled according to (9). Having computed the degree of membership of the examples to these two nodes using (5), we split them in turn. The process is continued until the stop condition is met. Among the possible stop rules, we chose the one based on the minimum acceptable reduction of the squared node error in the meaning of (7). In practice, this error reduction threshold is set such that the tree reaches up to 9–10 levels of depth.

As a standard practice in decision/regression tree learning, a separate set of examples, $\mathbf{x}_j, j=1,\ldots,M_P$, is used to prune the tree [37–39]. It is aimed at reducing the tree's complexity and prevents overfitting. The pruning algorithm adopted in this work is primarily aimed at finding the tree that performs the best on the pruning set.[4] It is carried out in the following way. The grown tree is fed with all pruning (validation) examples and the products $\mu_{D_i}(\mathbf{x}_j)L_{D_i}$ are stored for all nodes, as well as the tree classification rate defined as:

$$c_T = \sum_{j=1}^{M_P} w(\mathbf{x}_j)(1-|y(\mathbf{x}_j)-\hat{y}(\mathbf{x}_j)|). \tag{11}$$

Subsequently, we simulate collapsing a subtree rooted at each non-terminal node to a terminal node with the same label. Each obtained tree is evaluated using (11) and the stored membership degree-label products. The tree leading to the maximum increase in the classification rate over the pruning examples is saved and the process is continued until no further improvement can be made.

It should be noted that because *SimKRT* is essentially a binary classifier, it can be easily extended to a boosted ensemble proposed by Freund and Schapire [40]. It is done by treating a single tree as a weak classifier and using weights $w(\mathbf{x})$ from Eqs. (7), (10) and (11) for controlling the importance of the input pairs in the consecutive boosting rounds, according to the well-known exponential loss function:

$$w_j^{(t+1)} = w_j^{(t)} e^{-F(\mathbf{x})y}, \tag{12}$$

where $F(\mathbf{x})$ denotes the discriminant function obtained in the current round.

Other tree combination schemes, such as bagging [41] or random forest [42],[5] are also straightforward to implement. In Section 5 we demonstrate that using such tree ensembles improve overall classification accuracy.

3.3. Parallel implementation

Our preliminary experiments revealed that the introduction of soft node splitting significantly increased the predictive accuracy of the proposed regression trees. However, sequential implementation of the processing steps defined in Eqs. (5) and (6) leads to a heavy computational load resulting from the necessity of performing local image comparisons at multiple tree nodes and branches one after another. This problem is especially severe for the classifiers where trees have many levels of depth and the split kernel functions yield non-zero value in the entire $[0,\infty]$ interval, e.g. Gaussian RBF. Such trees may appear prohibitively expensive to evaluate and hence unsuitable for many real-time recognition tasks.

To help overcome the above performance problem, we propose a parallel implementation of the classifier. The architecture of the system is illustrated in Fig. 2. The skeleton of the classifier's code was written in C++ and deployed on a *Nios II* embedded processor implemented entirely in the programmable logic and memory blocks of the Altera's *Stratix II* FPGA chip. Shared SDRAM memory acts as a storage for a copy of the original input image to be analyzed, the object prototypes and various auxiliary data structures utilized by the software-side program code. Parallel tree evaluation was offloaded to an FPGA co-processor featuring separate hardware process per tree node and appropriate synchronization. It enabled efficient implementation of the recursive formulas defined in Eqs. (5) and (6).

In Fig. 2 the process assigned to a given node[6] first receives the corresponding portion of both input images from the dispatcher process. Then, it evaluates the underlying image descriptor, calculates the local distance, determines the response of the kernel function, and upon completion waits for a signal to be sent by the parent node process. This signal contains the parent's membership degree and kernel response for the input image pair. When received, the process assigned to the current node calculates its own membership degree, posts an update signal to both successor nodes, if present, and then suspends its execution. This enables correct recursive computation of the membership degree of an input example to the terminal nodes according to Eq. (5). Once a given component of the sum in the numerator of Eq. (6) has been calculated by the corresponding terminal node process, a signal is sent to the parent node process. A cascade of such asynchronous signals reactivates the suspended processes which allows the ultimate tree's response to be computed on the way back from the recursion. The operation of a single non-terminal node process is shown in Algorithm 1.

[4] Other criteria are possible. See for instance [38,39].

[5] It should be noted that in the random forest construction procedure the pruning step is omitted.

[6] We mean here a process responsible for all the computations related exclusively to that node.

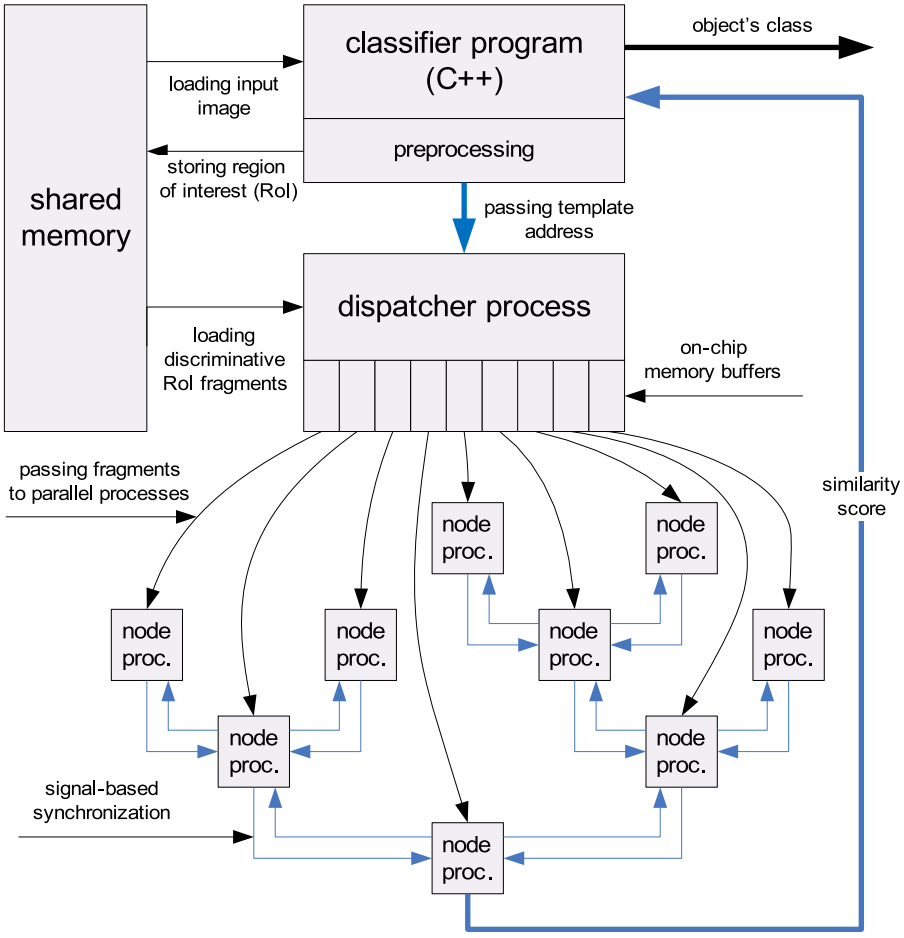

Fig. 2. The architecture of the parallel object classifier implemented in FPGA.

The root node process finally sends a signal with the response to the software-side classifier's program. This in turn triggers estimation of the similarity of an input image to the next prototype. Once all prototypes have been compared to, the most similar ones are determined according to a *k-NN* rule. Note that in the above distributed tree evaluation scheme the most intensive computations, i.e. computation of the image descriptor and local distance followed by kernel evaluation (line 2 in Algorithm 1), are done in parallel.

Algorithm 1. A single cycle of operation of a non-terminal node process participating in a distributed computation of the *SimKRT* tree's response for an input image pair according to Eqs. (5) and (6). $\hat{y}_{D_L}$ and $\hat{y}_{D_R}$ denote the output of subtrees rooted at the child nodes of node D and $\hat{y}_D$ is the output of a subtree rooted at D.

1: Receive the appropriate portion of the input images $\mathbf{x}$ via a stream from the dispatcher
2: Calculate $\kappa_{D,k(D),\Theta(D)}(\mathbf{x})$
3: Wait for the parent node process' signal with $\kappa_{P,k(P),\Theta(P)}(\mathbf{x})$ and $\mu_P(\mathbf{x})$
4: Calculate $\mu_D(\mathbf{x})$ using Eq. (5)
5: Post a signal with $\kappa_{D,k(D),\Theta(D)}(\mathbf{x})$ and $\mu_D(\mathbf{x})$ to the child node processes
6: Wait for child node processes' signals with $\hat{y}_{D_L}$ and $\hat{y}_{D_R}$
7: Calculate $\hat{y}_D = \hat{y}_{D_L} + \hat{y}_{D_R}$
8: Post a signal with $\hat{y}_D$ to the parent node process

4. Experiments

In order to validate the proposed approach, we have conducted several experiments. The aim of each experiment was to learn a visual similarity function from pairs of images representing various objects. Each function was then put into a *k-NN* framework to enable recognition of the previously unseen images. Ultimately, we wanted to justify our claim that: (1) learning how individual object instances differ from one another facilitates construction of efficient, general-purpose object classifiers and (2) *SimKRT* framework is an inexpensive means of building such classifiers. In Section 4.1 we give a brief description of the image datasets used in our experimental evaluation. In Section 4.2 the results obtained in each experiment are discussed.

4.1. Datasets

Six image datasets were used in our experiments. Three of them: *Yale faces A*, *AT&T faces* and *Labeled Faces in the Wild* are public and available online.[7] Here only a basic characteristic of these datasets is provided.

The *Yale faces A* dataset is comprised of 165 images representing 15 different individuals, 11 images per individual. For our purposes

[7] *Yale faces A* dataset is available at the author's web site: http://home.agh.edu.pl/~aruta/research/pr/datasets/Yale_Faces_A.zip, *AT&T faces* dataset is available at: http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html, *Labeled Faces in the Wild* database's home page is http://vis-www.cs.umass.edu/lfw/.

Fig. 3. Example images from the test datasets: Japanese traffic signs (highway code templates – first row, dataset images – second row), car fronts (third and fourth row), car rears (fifth and sixth row), *Yale faces A* (seventh and eighth row), *AT&T faces* (ninth and tenth row) and *Labeled Faces in the Wild* (bottom row). For the first two face datasets different views of the same person are shown to illustrate the within-class variation of appearance.

they were scaled to the size of 150×114 pixels. Images representing the same person differ in terms of the illumination (intensity and angle of incidence) and facial expression of the subjects. The *AT&T faces* dataset consists of 400 90×110 pixels images of 40 individuals (10 images per subject). The images representing the same subject feature only minor facial expression and scene illumination variations, but the same face is shown from slightly varying view angles. In both datasets certain persons are depicted with and without glasses. *Labeled Faces in the Wild* [12] is a large benchmark designed for studying the problem of unconstrained face recognition. It contains over 13,000 images of faces collected from the web using the Viola and Jones' detector [26]. The subset chosen for investigation contains images of only those people who appear in at least 10 pictures. They have been downscaled to 120×120 pixels.

The fourth dataset represents 17 types of Japanese traffic signs.[8] It contains images extracted from a real-life video captured in various street scenes from a moving vehicle. This dataset is split into 8473 training images and 9036 test images coming from disjoint sequences, all scaled to 60×60 pixels. It is severely unbalanced, i.e. the numbers of images representing each class vary significantly: from less than twenty to more than one thousand. The quality of the images ranges from very clean (good illumination and contrast), to very noisy (poor illumination, light reflections, motion blur).

The two remaining datasets were built by us. They depict fronts and rears of 12 models of full-size European and Japanese cars. Each model is represented by 30 front and 30 rear images, giving a total of 360 front and 360 rear images, all scaled to the size of 200×100 pixels. The images are well aligned and only small camera's pan and tilt variations were allowed when the pictures were taken. The cars representing the same model differ in terms of the body color and general scene illumination as they were captured at different times of day and at different locations across Europe, both outdoors and inside the showrooms.

Example images from all six datasets are shown in Fig. 3.

4.2. Experimental results

This section presents an experimental evaluation of the proposed visual similarity learning algorithm using the previously

[8] This dataset was provided by Shintaro Watanabe from the Advanced Technology R&D Center, Mitsubishi Electric Corporation, Amagasaki, Japan, and used in the experiments described in [43].

described image datasets. The learned similarity function was employed by a *k-NN*-like classifier and the success recognition rate was measured for performance assessment.

4.2.1. Real-time traffic sign recognition

In the first experiment we measured the correct classification rate of a *k-NN* classifier tested on the road sign dataset using separately three different low-level image descriptors: Haar filters [26], HOG-s [29], and region covariances [28]. In the first case the input feature space was populated by five types of filters capturing certain horizontal, vertical, and diagonal structures of the underlying images, separately for each color channel. The size of each rectangular part of the filters satisfied the condition $w,h=\{4px,8px\}$ and the filters were shifted by half of that size along each dimension. In the second case we used 6-bin histograms computed over regions of scale $w,h=\{10px,20px\}$ shifted by 10 pixels along each dimension. In the latter case, a pool of all 4-feature covariance matrices were constructed in the same regions as HOG-s. The matrices combined x and y positional coordinates and the first-order image derivatives along horizontal and vertical axis. The local distance metrics used in (1) for the above mentioned descriptors are listed in Table 1.

An ensemble of *Kernel Regression Trees* were trained via boosting combined with cross-validation on the training set. First 40% of the images of each sign were chosen to build a growing set, and all other images were put in a pool used to construct a pruning set. Appropriate sampling was done on such partitioned data, respecting the proportions of the images representing the classes in the original data. Its goal was to limit the number of image pairs passed on input of each tree to approximately 2000 and to keep the percentage of "same" pairs at approximately 0.15 level. Having grown and pruned the first ensemble, another 40% of images in each class, half-overlapping the last growing portion, were used for the new growing set construction, against the other 60% used for the new pruning set construction. Using such a training scheme, five tree ensembles were generated, each containing five trees, and their responses were averaged when evaluating the *k-NN* classifier on the test set.

Evaluation of the classifier was done using the following strategy. A large pool of test examples were constructed by picking each image from each class C_i, $i=1,\ldots,N$ and pairing it with a randomly selected image from each other class C_j, $j\neq i$, as well as with a randomly selected image from the same class, but ensuring no same two images were selected. This way the comparison set contained N prototypes. A test example was classified correctly if among the resulting N pairs, one "same" and $N-1$ "different", the "same" pair was assigned the highest similarity score. To avoid bias resulting from random selection of the second image in each test pair, ten test runs per classifier were performed and the recognition rates were averaged.

Correct classification rates obtained for the 17-class Japanese road signs problem are shown in Table 3. These rates are additionally compared to the results obtained using three alternative techniques: (1) PCA-TM, template matching in a PCA-reduced feature space built from the concatenated, regularly spaced HOGs, (2) class-specific template matching approach (CSTM) discussed in [44] and (3) a k-NN classifier based on the similarity function learned using an *AdaBoost*-based algorithm proposed by Jones and Viola [25]. Fig. 4a illustrates the influence of boosting on the classifier's accuracy. Additionally, in Fig. 4b the confusion matrix corresponding to the most discriminative similarity function learned has been shown.

Results of the above experiment show that the proposed similarity learning scheme generates efficient road sign classifiers that, when combined appropriately, are comparable to *AdaBoost* and outperform the alternative algorithms. Taking into account the low quality of many test images, nearly 80% recognition rate observed is a promising result. Further accuracy gain can be achieved by mixing color-aware and color-unaware features within the same trees and by extending both the comparison set and the parameter k of the classifier. However, within a sequential architecture it would cost a significant drop in the computational performance and hence an inability to test the classifier on video input [43]. The herein described implementation can run at approximately 30 fps on a standard PC equipped with Pentium IV processor for $k=1$, but slows down linearly with k for $k>1$. Note (Fig. 4b) that most confusions occurred between semantically similar classes, e.g. between speed limits or between "no stopping" and "no parking" signs. In a real-life vision-based driver assistance system on board of a moving vehicle such partial classifications may still be of use as they enforce the same type of driver's reaction.

4.2.2. Car model recognition

Another experiment was conducted to validate our pairwise visual similarity learning approach on the car image datasets described in Section 4.1. The effect of combining multiple trees on the overall performance of the resulting classifier was investigated. We were particularly interested in how much gain in the success recognition rate can be achieved using different kernels and different tree combination methods (bagging [41], boosting [40] and random forest [42]), and at what computational cost.

In this experiment several tree ensembles were trained separately from the images of car fronts and rears, using HOGs as the underlying image representation, in order to predict the model of new, unseen cars. Both datasets were divided into training and test parts. Ten images per model were used for tree training and the other 20 images per model were used for evaluation. Each ensemble was trained via cross-validation in the same way as described in Section 4.2.1 and the number of trees built using the same portion of data was treated as a variable. The testing scheme was a generalization of the one used in the traffic sign recognition experiment. Specifically, each test image of class C_i, $i=1,\ldots,N$ was paired with k ($k\geq 1$) randomly chosen (without repetitions) images from each class C_j, $j=1,\ldots,N$, giving a total ok kN pairs in the comparison set. The classification was done via majority voting among the k images most similar to C_i.

Results of the experiment are shown in Table 4. The "var" abbreviation is used to denote the variable kernel that is chosen automatically and independently at each tree node from among

Table 3
Classification rates (in percent) obtained for a 17-class Japanese traffic signs problem using different methods: PCA-TM, CSTM [44], modified *AdaBoost* of Jones and Viola [25], and our *SimKRT–k-NN* framework. In parentheses variances of the classification rates are provided.

Feature type	PCA-TM	CSTM	*AdaBoost*	5 × 5 bagged *SimKRT*	5 × 5 boosted *SimKRT*
Haar filter	X	X	62.4 (0.2)	60.0 (0.3)	57.8 (0.2)
HOG	22.3 (0.2)	74.4 (0.3)	74.7 (0.2)	74.4 (0.2)	76.8 (0.3)
Covariance	X	X	54.4 (0.2)	47.1 (0.4)	54.1 (0.3)

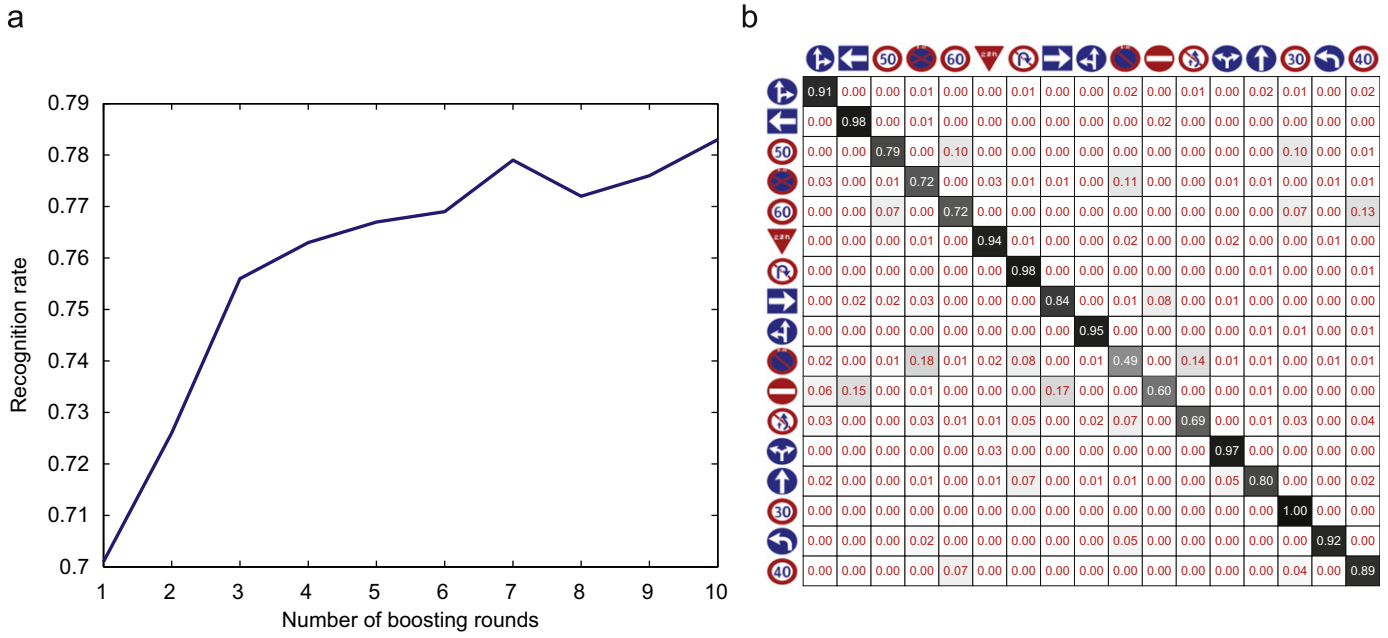

Fig. 4. Experimental evaluation of the classifier based on a boosted *SimKRT* ensemble with piecewise linear splitting kernels and HOG descriptors: (a) the influence of the number of boosting rounds on the classifier's accuracy, (b) a confusion matrix obtained for an ensemble of 5×10 trees. This figure is best viewed in color.

Table 4
Classification rates (in percent) obtained for the car front and car rear datasets using combined *Kernel Regression Trees*. Results associated with the two best-performing combinations for each dataset are marked. Note that boosting is meaningless and equivalent to bagging for a single round of training (table entries marked with "X").

Setting	5×1 trees				5×5 trees				5×10 trees			
	Front		Rear		Front		Rear		Front		Rear	
	1-NN	3-NN	1-NN	3-NN	1-NN	3-NN	1-NN	3-NN	1-NN	3-NN	1-NN	3-NN
Bagg./step	57.9	71.3	50.7	68.7	63.6	77.7	73.2	86.5	69.3	81.4	77.2	90.7
Bagg./p-l	56.9	72.4	59.3	74.0	65.7	77.6	82.3	91.2	68.5	79.1	83.3	93.3
Bagg./RBF	64.9	75.9	82.1	90.4	69.1	80.3	83.6	90.6	67.7	78.8	84.0	91.0
Boost./step	X	X	X	X	75.2	82.8	79.4	91.3	78.9	89.1	85.5	94.5
Boost./p-l	X	X	X	X	73.2	85.0	81.8	92.9	78.6	87.9	85.6	94.9
Boost./RBF	X	X	X	X	77.5	85.3	84.9	92.1	82.4	91.3	88.0	94.9
Boost./var	X	X	X	X	80.6	88.4	84.9	93.1	81.5	91.1	88.5	96.4
Forest/step	50.5	65.9	53.1	74.8	69.1	81.9	76.9	89.8	71.7	85.1	81.3	92.1
Forest/p-l	63.1	77.7	59.2	78.8	72.9	85.8	80.2	92.2	75.3	88.5	84.3	94.2
Forest/RBF	78.8	89.1	85.9	92.8	81.1	90.0	87.0	94.3	80.3	90.8	88.1	95.1
Forest/var	63.8	76.8	51.5	69.8	74.8	85.3	79.4	92.1	76.5	87.5	82.1	92.7

Table 5
Average numbers of visited tree nodes per example depending on the tree combination scheme and the split kernel function used.

Dataset	Bagg. step	Bagg. p-l	Bagg. RBF	Boost. step	Boost. p-l	Boost. RBF	Forest step	Forest p-l	Forest RBF
Front	2.48	2.73	54.05	3.87	5.25	37.85	12.95	16.53	89.00
Rear	2.35	3.20	64.20	4.22	5.14	51.55	11.98	17.88	65.85

all considered kernel types. In addition, in Table 5 the average numbers of visited tree nodes per example are listed, depending on the tree combination scheme and the split kernel type used.

Clearly, among the tree ensembling methods boosting and random forest appear to be superior to bagging. The latter consists in taking a simple average of the performances of multiple trees trained from different, randomly sampled portions of data (so-called *bootstrap samples*). In practice, it significantly reduces variance of the prediction accuracy of a single tree, but is generally not guaranteed to improve its average accuracy. Therefore, based on the results from Table 4, we conclude that using a single *SimKRT* tree for similarity estimation does not provide enough discriminative power for datasets with very low between-class variance, such as those considered in this study.

Regarding the fuzziness, it guarantees consistently better recognition, compared to the results obtained using the hard-split trees, as shown in Table 4. However, it holds on condition that the shape of the kernel functions is chosen appropriately. In the particular case of car datasets the trees incorporating Gaussian RBF kernels outperform the hard-split trees significantly, but the ones incorporating the piecewise-linear kernels do not (see face recognition results in Section 4.2.3 for comparison). We attribute it to the sub-optimality of the piecewise-linear kernel function, of which the step kernel is a special case

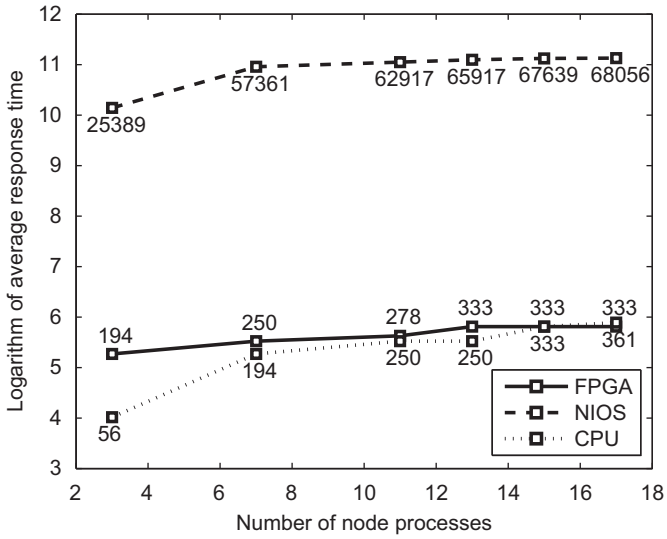

Fig. 5. Comparison of the average response time of a 17-node *Kernel Regression Tree* using: a parallel FPGA-based accelerator controlled by a *Nios II* embedded processor program, both working at 100 MHz, and the sequential implementations on: (1) a PC equipped with 2-core, 32-bit CPU (Intel T9600, 2.8 GHz), 4 GB RAM, 1 GB RAM GPU, and (2) the same embedded processor, but without hardware acceleration. The graphs are shown in logarithmic scale as functions of n, the number of tree nodes participating in the distributed computation. The data points have been annotated with the original execution times measured in microseconds.

(with parameter $\beta=0$). Although soft splitting results in much deeper trees (see Table 5) and thus more expensive processing to be done to make a prediction, this cost is worth consideration in all applications that are not mission-critical or where hardware acceleration is possible. Interestingly, the highest success recognition rate was observed for variable-kernel trees only in boosted ensembles, but not in random forests. After closer inspection, trees in those ensembles were found to contain a well-balanced mixture of both fuzzy discriminant types.

To show that compromising the fuzzy tree's accuracy by adopting computationally less expensive solutions can be avoided, we measured the average response time of an example 17-node *SimKRT* tree implemented sequentially and compared it to a massively parallel implementation in FPGA, as described in Section 3.3. Fig. 5 shows the obtained results. They clearly confirm the advantages of both FPGAs and the proposed algorithm. When implemented using hardware accelerator, the classifier runs dramatically faster than the sequential version executed solely by an embedded processor within the same hardware platform. In the same time it is comparatively fast to the sequential algorithm deployed on a modern PC with nearly 30 times faster system clock and generally much more powerful hardware architecture. Therefore, taking into account better resource utilization, lower power consumption and reduced cost of FPGAs, it seems that the implementation of parallelized *SimKRT*-based classifiers in this way is justified, e.g. when building autonomous smart cameras for access control.

Finally, it should be noted that the *SimKRT* tree model is easily interpretable as it can be used for descriptive visualization of the automatic decision-making process. This is particularly interesting when confronted with how humans spot and rank the importance of local object differences. As an example, we illustrate in Fig. 6 the importance of the first seven top-scored image descriptors found by the trees trained on the car datasets. By "top-scored" we mean those descriptors that parameterized the splitting kernel functions at the first three levels of the trees. It can be seen that for instance in order to estimate similarity between two rear car views, the tree first puts focus on the regions where the rear lights are typically located. On the other hand, comparison between two car fronts proceeds by primarily looking into the grille and headlight areas. The presence and the shape of fog lights appears to be a secondary clue.

4.2.3. Face recognition

In separate experiments the capabilities of our recognition system were demonstrated on three face datasets: *Yale faces A*, *AT&T faces* and *Labeled Faces in the Wild*. In the first case the face similarity was learned in the same way as from car images (see Section 4.2.2), but five images per class were used for training and six images per class were used for testing. In the latter the *3-NN* scheme was applied. Table 6 illustrates the obtained results for the three best-performing *SimKRT* setups. They are compared to the best correct recognition rates obtained for the same dataset partition by applying linear projections, *Eigenfaces* and *Fisherfaces*, to the data, followed by *k-NN* classification. This method was originally proposed for the *Yale faces A* dataset in [45].[9]

Using the *AT&T faces* dataset, our goal was to demonstrate that the learned similarity function can also be used to recognize previously never seen objects, as in [16]. Specifically, we learned similarity from the pairs of same/different images representing 20 individuals. Then, we tested the 3-NN classifier based on this similarity on the same/different pairs generated from the images of 20 completely different individuals. To avoid bias in the results, upon completion of the first test run the training and test subsets were swapped. The training/test pair generation scheme described earlier in this section was adopted. The averaged recognition rates of the best obtained classifiers are shown in Table 7. For comparison, we learned *Eigenfaces* and *Fisherfaces* projections from the training set and applied them to the test set, retaining the same 3-NN classifier and the comparison set randomization procedure.

SimKRT framework was finally tested against a more challenging benchmark, *Labeled Faces in the Wild*. Compared to the above two datasets, additional difficulties included much higher number of subjects, pose differences and large background variations. To achieve a better degree of face alignment, we adopted a fiducial-points based technique similar to the one reported by Taigman et al. [46]. Specifically, we employed the face detector developed by the *Visual Geometry Group* from Oxford University, available at: http://www.robots.ox.ac.uk/~vgg/research/nface/. This detector outputs coordinates of the fiducial points, such as eyes, nose and mouth. We generated such coordinates from all training and test images and applied PCA to determine a 1-dimensional space in which faces seen from different viewpoints became reasonably well separable. It was followed by appropriate binning of the values in this 1-dimensional space. As a result, the original dataset was partitioned into five smaller subsets, each corresponding to a separate appearance mode, i.e. containing faces seen from roughly the same viewpoint.

The training procedure adopted, although operating on smaller sets of images than the studies included in the LFW benchmark,[10] satisfies the conditions of the so-called "image-unrestricted protocol". It means that the training data provides both the equivalence constraints for each image pair and the subject identity for each image. A separate face similarity function was learned from each pose-dependent subset using a boosted ensemble of five *SimKRT* trees. Five-fold cross-validation scheme was

[9] The classification rates reported in [45] depart from those obtained in our experiments due to the differences in the image preprocessing and different sizes of training sets. Namely, Belhumeur et al. cropped faces to eliminate background and trained their model on all images except for the one being classified.

[10] Benchmark ROC curves can be viewed at: http://vis-www.cs.umass.edu/lfw/results.html.

Fig. 6. Visualization of the first three levels of a *SimKRT* classifier trained on the car front dataset (left) and the car rear dataset (right). Selected HOG regions used to build split rules at each node are overlapped on two random car images.

Table 6
Classification rates obtained for the *Yale faces A* dataset using the three best-performing Kernel Regression Tree ensembles.

Setting	5×1 trees (%)	5×3 trees (%)	5×10 trees (%)	Eigenfaces (%)	Fisherfaces (%)
Boost./p-l	82.6	89.3	90.2		
Boost./RBF	71.1	80.4	82.0	90.0	73.3
Forest/p-l	88.2	94.0	83.8		

Table 7
Classification rates obtained for the *AT&T faces* dataset using the two best-performing Kernel Regression Tree ensembles. In this experiment the previously never seen faces were recognized.

Setting	5×1 trees (%)	5×3 trees (%)	5×10 trees (%)	Eigenfaces (%)	Fisher faces (%)
Boost./p-l	71.2	83.5	88.1		
Forest/p-l	77.7	85.2	89.4	87.4	74.5

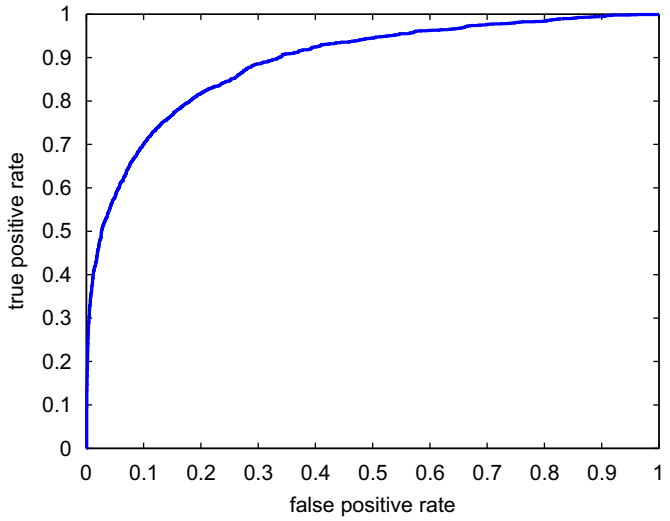

Fig. 7. Averaged ROC curve obtained from the face similarity functions learned from different pose-dependent subsets of the LFW database using a boosted *SimKRT* ensemble of 5×40 trees.

adopted class-wise, i.e. such that no images of the test individuals were seen at the model training stage, as in the experiment involving *AT&T faces* dataset. Local image distances were measured with respect to region covariance descriptors [28] of scale in between 10 and 40 pixels along each direction. This time, binary discrimination capability of the classifier ("same" vs "different") was evaluated. Although direct comparison of the results to the state of the art is not valid due to the dataset size differences, the obtained ROC curve (Fig. 7) is close to the best ones so far published [23,46].

In the above experiments it has been demonstrated that the proposed algorithm is suitable for learning visual similarity between relatively well-aligned human faces. This similarity measure can further be employed for efficient discrimination between multiple individuals. In the case of *Yale faces A* dataset the *3-NN* classifier incorporating the learned similarity function reached up to 94% success recognition rate. More importantly, through the experiment involving the *AT&T faces* and LFW datasets we showed that the proposed approach has large generalization capabilities as the similarity function need not be constructed from the images of the same individuals that are further to be recognized. It is only essential that both training and test object instances share the same general appearance characteristics which can be captured from only a limited number of training instances. Finally, in all experiments the proposed method performed comparably or better than the selected benchmark approaches.

5. Conclusions

The classic approach to solving multi-classification problems is either by modeling the joint probability of image features and class labels followed by an appropriate Bayesian inference (generative paradigm) or by modeling the decision boundary (discriminative paradigm). While the first strategy is adequate for rough image categorization, it cannot capture subtle differences between (very) similar objects falling into the same broad category. On the other hand, typical discriminative approaches are not well scalable and require many training instances. As a result, for problems involving multiple classes and little data a costly decomposition into a large number of binary problems is often inevitable.

The main intuition behind our solution to the above problem is that humans often discriminate between a number of similar objects by primarily identifying the key differences between them. This cannot be done reliably in a pose-invariant fashion and using the generative modeling techniques which inherently assume loose constraints on the objects' shape and appearance. Therefore, we claim that to deal with this type of multi-classification tasks, it is more adequate to learn which distinct data objects resemble one

another and how. To do that, we have formulated the notion of global image similarity as a trainable function of multiple localized image distances. Our goal was set to infer this function automatically from the input pairs of images such that the pairs depicting the objects of the same class become clearly distinguishable from the pairs representing two different object classes.

The proposed visual similarity learning algorithm is based on a novel formulation of a fuzzy regression tree that is grown and/or pruned from image pairs. We call it *Pairwise Similarity Based Kernel Regression Tree* (*SimKRT* for short). In our tree framework local image distances and the corresponding image descriptors are combined into a hierarchy of decision stumps where on each level of the hierarchy the soft degree of assignment of the input pair to the "same" class is estimated. In addition, each decision stump is made fuzzy by allowing the local input space to be split into overlapping subspaces using the appropriate kernel functions. Trained trees are finally combined into robust ensembles to further increase the stability and the discriminative power of the obtained similarity function.

Our approach has been evaluated on several challenging image datasets. Namely, each learned similarity function was used with a *k-Nearest Neighbors* classifier to rate the resemblance of a number of test image pairs where the second image in each pair was treated as a prototype of known class. We studied the influence of various parameters on our system, including the feature representation of the images, local distance metrics, split kernel function parameters, classifier ensemble types and sizes or training pair subsampling schemes. For the optimal parameter combinations found, high success recognition rates were observed, proving the usefulness of our method.

In contrast to many previous approaches, the proposed method shows numerous desirable properties. First and foremost, it gives flexibility in choosing both the type of image features and the methods of comparing them. Moreover, *SimKRT* trees can be efficiently implemented in parallel hardware, yielding very fast classifiers at no performance cost. Our method is also suitable for handling the recently addressed problem of recognizing never seen objects. This application is particularly interesting in the presence of many similar yet distinct instances of the same category, e.g. human faces, most of which for obvious reasons cannot be reflected in the training data. Finally, the concept of combining local distances into a hierarchy is easy to comprehend, allowing one to quickly visualize and possibly tune the decision making-process according to the available expert knowledge.

Acknowledgement

EUROPEAN SOCIAL FUND

The research presented in this paper has been partially supported by the European Union within the European Social Fund program no. UDA-POKL.04.01.01-00-367/08-00.

References

[1] D.G. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision 60 (2) (2004) 91–110.
[2] L. Fei-Fei, P. Perona, A Bayesian hierarchical model for learning natural scene categories, in: Proceedings of the 2005 IEEE International Conference on Computer Vision and Pattern Recognition, 2005, pp. 524–531.
[3] J. Sivic, B. Russell, A. Efros, W. Freeman, Discovering object categories in image collections. in: Proceedings of the 2005 International Conference on Computer Vision, vol. 1, 2005, pp. 370–377.
[4] E. Sudderth, A. Torralba, W. Freeman, A. Willsky, Describing visual scenes using transformed objects and parts, International Journal of Computer Vision 77 (1–3) (2005) 291–330.
[5] S. Fidler, M. Boben, A. Leonardis, A coarse-to-fine taxonomy of constellations for fast multi-class object detection, in: Proceedings of the 11th European Conference on Computer Vision, Part V, 2010, pp. 687–700.
[6] K. Crammer, Y. Singer, On the algorithmic implementation of multiclass kernel-based vector machines, Journal of Machine Learning Research 2 (2002) 265–292.
[7] W. Hao, J. Luo, Generalized multiclass adaboost and its applications to multimedia classification, in: Proceedings of the 2006 Computer Vision and Pattern Recognition Workshop, 2006, p. 113.
[8] J. Platt, N. Cristianini, J. Shawe-Taylor, Large margin dags for multiclass classification, in: Advances in Neural Information Processing Systems, MIT Press, 1999, pp. 547–553.
[9] G.-D. Guo, H.-J. Zhang, S. Li, Pairwise face recognition. in: Proceedings of the 2001 International Conference on Computer Vision, vol. 2, 2001, pp. 282–287.
[10] J. Zhang, M. Marszałek, S. Lazebnik, C. Schmid, Local features and kernels for classification of texture and object categories: a comprehensive study, International Journal of Computer Vision 73 (2) (2001) 213–238.
[11] R. Rifkin, A. Klautau, In defense of one-vs-all classification, Journal of Machine Learning Research 5 (2004) 101–141.
[12] G.B. Huang, M. Ramesh, T. Berg, E. Learned-Miller, Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments, Technical Report 7-49, University of Massachusetts, Amherst, October 2007.
[13] B. Russell, A. Torralba, C. Liu, R. Fergus, W. Freeman, Object recognition by scene alignment. in: Advances in Neural Information Processing Systems, vol. 20, MIT Press, 2008, pp. 1241–1248.
[14] T. Hastie, R. Tibshirani, Discriminant adaptive nearest neighbor classification, IEEE Transactions on Pattern Analysis and Machine Intelligence 18 (6) (1997) 607–616.
[15] C. Domeniconi, D. Gunopulos, Adaptive nearest neighbor classification using support vector machines. in: Advances in Neural Information Processing Systems, vol. 14, MIT Press, 2001, pp. 665–672.
[16] E. Nowak, F. Jurie, Learning visual similarity measures for comparing never seen objects, in: Proceedings of the 2007 IEEE International Conference on Computer Vision and Pattern Recognition, 2007, pp. 1–8.
[17] T. Malisiewicz, A. Efros, Recognition by association via learning per-exemplar distances, in: Proceedings of the 2008 IEEE International Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8.
[18] H. Zhang, A. Berg, M. Maire, J. Malik, SVM-KNN: discriminative nearest neighbor classification for visual category recognition, in: Proceedings of the 2006 IEEE International Conference on Computer Vision and Pattern Recognition, 2006, pp. 2126–2136.
[19] P. Paclík, J. Novovicová, R. Duin, Building road-sign classifiers using a trainable similarity measure, IEEE Transactions on Intelligent Transportation Systems 7 (3) (2006) 309–321.
[20] P. Phillips, Support vector machines applied to face recognition. in: Advances in Neural Information Processing Systems, vol. 11, MIT Press, 1999, pp. 803–809.
[21] V. Athitsos, J. Alon, S. Sclaroff, Efficient nearest neighbor classification using a cascade of approximate similarity measures, in: Proceedings of the 2005 IEEE International Conference on Computer Vision and Pattern Recognition, 2005, pp. 486–493.
[22] A. Frome, Y. Singer, F. Sha, J. Malik, Learning globally-consistent local distance functions for shape-based image retrieval and classification, in: Proceedings of the 2007 IEEE International Conference on Computer Vision, 2007, pp. 1–8.
[23] M. Guillaumin, J. Verbeek, C. Schmid, Is that you? Metric learning approaches for face identification, in: Proceedings of the 2009 IEEE International Conference on Computer Vision, 2009, pp. 498–505.
[24] L. Wolf, T. Hassner, Y. Taigman, The one-shot similarity kernel, in: Proceedings of the 2009 IEEE International Conference on Computer Vision, 2009, pp. 1–8.
[25] M. Jones, P. Viola, Face Recognition Using Boosted Local Features, Technical Report TR2003-25, Mitsubishi Electric Research Laboratories, 2003.
[26] P. Viola, M. Jones, Robust real-time face detection, International Journal of Computer Vision 57 (2) (2004) 137–154.
[27] F. Porikli, Integral histogram: a fast way to extract histograms in cartesian spaces. in: Proceedings of the 2005 IEEE International Conference on Computer Vision and Pattern Recognition, vol. 1, 2005, pp. 829–836.
[28] O. Tuzel, F. Porikli, P. Meer, Region covariance: a fast descriptor for detection and classification, in: Proceedings of the 2006 European Conference on Computer Vision, 2006, pp. 589–600.
[29] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection. in: Proceedings of the 2005 IEEE International Conference on Computer Vision and Pattern Recognition, vol. 1, 2005, pp. 886–893.
[30] C. Olaru, L. Wehenkel, A complete fuzzy decision tree technique, Fuzzy Sets and Systems 138 (2) (2003) 221–254.
[31] R. Chang, T. Pavlidis, Fuzzy decision tree algorithms, IEEE Transactions on Systems, Man and Cybernetics 7 (1) (1977) 28–35.
[32] Y. Yuan, M. Shaw, Induction of fuzzy decision trees, Fuzzy Sets and Systems 69 (2) (1995) 125–139.

[33] A. Suarez, F. Lutsko, Globally optimal fuzzy decision trees for classification and regression, IEEE Transactions on Pattern Analysis and Machine Intelligence 21 (12) (1999) 1297–1311.
[34] F. Gasir, Z. Bandar, K. Crockett, Elgasir: an algorithm for creating fuzzy regression trees, in: Proceedings of the 2009 IEEE International Conference on Fuzzy Systems, 2009, pp. 332–337.
[35] T. Dietterich, Ensemble methods in machine learning. in: Proceedings of the 1st International Workshop on Multiple Classifier Systems, vol. 1857, 2000, pp. 1–15.
[36] L. Torgo, Kernel regression trees, in: Proceedings of the 9th European Conference on Machine Learning, 1997, pp. 118–127.
[37] L. Breiman, J. Friedman, R. Olshen, C. Stone, Classification and Regression Trees, Chapman & Hall, New York, NY, 1984.
[38] M. Bohanec, I. Bratko, Trading accuracy for simplicity in decision trees, Machine Learning 15 (3) (1994) 223–250.
[39] L. Torgo, Error estimators for pruning regression trees, in: Proceedings of the 10th European Conference on Machine Learning, 1998, pp. 125–130.
[40] Y. Freund, R. Schapire, A short introduction to boosting, Journal of Japanese Society for Artificial Intelligence 14 (5) (1999) 771–780.
[41] L. Breiman, Bagging predictors, Machine Learning 24 (2) (1996) 123–140.
[42] T. Ho, The random subspace method for constructing decision forests, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (8) (1998) 832–844.
[43] A. Ruta, F. Porikli, S. Watanabe, Y. Li, A new approach for in-vehicle camera traffic sign detection and recognition, in: Proceedings of the 2009 IAPR Conference on Machine Vision Applications, 2009, pp. 509–513.
[44] A. Ruta, Y. Li, X. Liu, Towards real-time traffic sign recognition by class-specific discriminative features. in: Proceedings of the 2007 British Machine Vision Conference, vol. 1, 2007, pp. 399–408.
[45] P. Belhumeur, J. Hespanha, D. Kriegman, Eigenfaces vs. fisherfaces: recognition using class specific linear projection, IEEE Transactions on Pattern Analysis and Machine Intelligence 19 (7) (1997) 711–720.
[46] Y. Taigman, L. Wolf, T. Hassner, Multiple one-shots for utilizing class label information, in: Proceedings of the 2009 British Machine Vision Conference, 2009, pp. 1–8.

Andrzej Ruta is an Assistant Professor at the Department of Computer Science, AGH University of Science and Technology, Krakow, Poland. He received his MSc from the same department in 2005 and the PhD degree from Brunel University, UK, where he worked in the area of computer vision and pattern recognition. His current research interests include computer vision, machine learning, pattern recognition, nature-inspired computing and time-series prediction.

Yongmin Li is a Senior Lecturer at the Department of Information Systems and Computing, Brunel University, Uxbridge, UK. He received his PhD from Queen Mary, University of London, MEng and BEng from Tsinghua University, China. His research interests cover the areas of computer vision, image processing, video analysis, medical imaging, bio-imaging, machine learning, pattern recognition, automatic control and non-linear filtering. Before joining Brunel University, he worked as a research scientist in the British Telecom Laboratories. Dr. Li is a senior member of the IEEE.