Pattern Recognition 103 (2020) 107291

Contents lists available at ScienceDirect

Pattern Recognition

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

Novel dimensionality reduction approach for unsupervised learning on small datasets

Petr Hurtik*, Vojtech Molek, Irina Perfilieva

Institute for Research and Applications of Fuzzy Modeling, Centre of Excellence IT4Innovations, University of Ostrava, 30. dubna 22, Ostrava, Czech Republic

ARTICLE INFO

Article history:
Received 29 November 2018
Revised 1 February 2020
Accepted 19 February 2020
Available online 22 February 2020

Keywords:
Unsupervised learning
Dimensionality reduction
PCA
F-transform
Image classification
Autoencoder

ABSTRACT

We focus on an image classification task in which only several unlabeled images per class are available for learning and low computational complexity is required. We recall the state-of-the-art methods that are used to solve the task: autoencoder-based approaches and manifold-decomposition-based approaches. Next, we introduce our proposed method, which is based on a combination of the F-transform and (kernel) principal component analysis. F-transform significantly reduces the computation time of PCA and increases the robustness of PCA to translation, while PCA proposes more descriptive features. This combination performs 3D reduction: the F-transform reduces dimensionality over a single 2D image, while PCA reduces dimensionality through the whole set of processed images. Based on the benchmark results, our method may outperform deep-learning-based methods in limited settings. For completeness, we also address other image resampling algorithms that can be used instead of the F-transform, and we find that the F-transform is the most suitable.

1. Introduction

General image classification problems are well-solved by user-friendly neural network (NN) architectures and pre-trained models, which can be used as starting points for solving other, similar problems. However, the disadvantages of these methods are (at least) twofold: high computation time and the demand for many – typically labeled – training samples. Such obstacles might be insurmountable, especially in the industry – it might be almost impossible to create reasonably large dataset, e.g., because products with a specified error may appear with a low frequency. Moreover, a device with sufficient computational power for deep learning methods can be difficult to integrate into available industrial systems with limited space. Partial solutions can be found in the literature, e.g., both the training time and lack of data can be mitigated by pre-trained models using the transfer learning technique. The computation time can be decreased by reducing the number of neural network parameters, by using specialized small architectures [1] or by increasing the computational power. In the case of lack of labeled data, only a few unsupervised approaches are available [2].

In this study, we consider an image classification problem, in which data is scarce: only several images per class are available, and the problem domain is so narrow that no suitable pre-trained model exists. Furthermore, labels for the training dataset are not available in the training phase. Finally, there is an emphasis on low computational complexity. Based on these assumptions, such a problem is highly challenging for deep learning techniques.

This paper presents a short overview of methods that are suitable for the defined problem and compares their performances with that of our proposed method based on our preliminary investigation [3]. The proposed method consists of a combination of PCA [4] and F-transform of a higher degree [5]. The contribution and the novelty of the proposed method are in the proper combination of two methods for dimensionality reduction, where the connection is beneficial for each of them – the F-transform significantly reduces the computation time of PCA and increases the robustness of PCA to translation, while PCA proposes more descriptive features. This combination realizes 3D reduction: the F-transform realizes dimensionality reduction over a single 2D image, while PCA realizes dimensionality reduction through the whole set of obtained reduced images. Based on the benchmark results "classical" methods may outperform deep-learning-based methods in limited settings.

The highlights and contributions of the paper are as follows:

- We investigated the state-of-the-art methods for unsupervised learning and selected PCA, together with its kernel version, as the baselines.
- We identified the main shortcomings of (k)PCA: high memory requirements and high computational complexity.

* Corresponding author.
E-mail address: petr.hurtik@osu.cz (P. Hurtik).

https://doi.org/10.1016/j.patcog.2020.107291
0031-3203/

- We proposed F-transform preprocessing for overcoming the main (k)PCA hortcoming.
- We experimentally proved the F-transform improves the performance of (k)PCA significantly and that it is the only suitable preprocessing method among various frequently used methods.

2. Current state

Considering the combination of a supervised deep neural network and a small training dataset, the most important effect to avoid is overfitting. In [6], the authors used a data augmentation approach for rotation-invariant object detection to make the model less prone to overfitting. The authors adopted convolutional layers from AlexNet and added a fully connected layer for extracting rotation-invariant features and an SVM classifier. Another interesting strategy for avoiding overfitting on a small training dataset was presented in [7], where the authors proposed a weakly supervised learning (WSL) framework for object detection and classification in aerial images. They benefit from the size of the images; hence they used the sliding window method to divide the images into smaller patches. The WSL framework utilized three stages of features extraction and transformation from low-level SIFT descriptors to middle-level locality-constrained linear coded features to high-level features that were created by stacked Boltzmann machines in an unsupervised manner. The final features are used with SVM.

Another strategy for addressing the problem of collecting labeled training data is to evade it by using an unsupervised approach. G. Hinton presented such an unsupervised approach in [8], where he introduced autoencoders. An autoencoder (AE) is a type of neural network architecture that consists of two essential parts: an encoder and a decoder. The encoder transforms the input vector into a code vector (*latent representation*) with lower dimensionality. The decoder accepts the code vector as input and reconstructs it. An autoencoder minimizes the difference between a reconstructed vector and an input vector; hence, an input vector is treated as a "label". The encoder and the decoder are connected through a code vector and can be trained in an end-to-end manner. Since the latent representation is smaller than the input vector, the autoencoder is essentially conducting compression/dimensionality reduction. An autoencoder can also be utilized for unsupervised pretraining, where the latent representation is treated as a feature vector and fed into the classifier. The classifier is trained on labeled data while the encoder is left out of the training process. Because the classifier needs labels, this process results in semi-supervised learning. Through the years, autoencoders have been evolving with NNs, especially convolutional autoencoders (CAEs) [9], which utilize convolutional filters to code/decode the latent representation. These autoencoders excel in processing images, where they significantly outperform [10] autoencoders with multilayer perceptron (MLP) architecture. In current research, variational autoencoders (VAEs) [11] are being developed, and learn the parameters of a probability distribution that models input data.

Examples of "classical" methods for unsupervised learning that are not based on neural networks, include (k)PCA [4], Isomaps [12], Locally Linear Embedding (LLE) [13], Hessian Local Linear Embedding (HLLE) [14], and Local Tangent Space Alignment (LTSA) [15]. All these methods realize non-linear manifold dimensionality reduction and can be divided into two groups according to their locality; global (Isomaps) and local ((k)PCA, LLE, HLLA, and LTSA). The difference is that a global method preserves the geometry of a manifold in a low-dimensional space while a local method cannot guarantee this property. In this paper, we use the term manifold to refer to a general high-dimensional space without defining its type or conditions. The main strategies of these mentioned algorithms are as follows. PCA relies on a linear transformation from a high-dimensional space into a low dimensional one, where the dimensions are uncorrelated. According to [16], the Isomap algorithm creates a graph of points on a manifold, searches it for only the shortest paths between neighboring points and, based on a matrix of distances, solves an eigenvalue problem. LLE assumes that each of manifold data points lies on a locally linear patch of the manifold, where a patch reconstructs each data point from its neighborhood. Then, a projection that is based on minimizing reconstruction errors according to a cost function of the patches is utilized. The HLLE method is based on LLE, but it uses a Hessian estimator. Finally, LTSA extracts local information by searching for kNN data samples, computes the largest eigenvectors, and constructs an alignment matrix, which is followed by alignment of the local coordinates. Classification that is based on any of these methods can be realized by adding a classifier at the output of the algorithm, such as NN or SVM.

Other methods are available for projecting original data into a reduced space, such as t-distributed Stochastic Neighbor Embedding [17], Multidimensional Scaling [18], and Spectral Clustering [19]. However, as they are non-parametric, they cannot establish a transformation from training data for application to test data.

3. Proposed method

The proposed method consists of two essential parts: the F-transform and PCA. First, we describe the theoretical foundations of them in Sections 3.1 and 3.2, respectively. Then, we describe how the methods are combined in Section 3.3.

3.1. Preliminaries: PCA and its kernel version (kPCA)

The original PCA relies on a linear transformation from a high-dimensional space into a low-dimensional space, where the dimensions are uncorrelated. Although PCA is sometimes regarded as unsupervised learning, the terminology is not precise; PCA uses data without the labels, but the projection is created analytically without any iterative learning; thus, rather than *unsupervised learning*, it is an *unsupervised algorithm/method*.

The theory behind PCAis as follows. Let $\mathbf{X}$ be an $r \times c$ matrix, where we consider r observations (images) with c measured variables (pixel intensities). Then, the transformation can be expressed using the following formula:

$$\mathbf{Y} = \mathbf{XP}, \tag{1}$$

where $\mathbf{P}$ is a $c \times m$ matrix that consists of the $m \leq c$ most important eigenvectors. $\mathbf{Y}$ is called a projected feature matrix, and it represents de-correlated (and dimensionality reduced) transformed input data. For realizing the transformation, we must define two matrices: $\mathbf{X}$ and $\mathbf{P}$. Let us start with $\mathbf{P}$, which can be defined as the solution to the following equation:

$$\mathbf{P\Delta P}^T = \mathbf{C}, \tag{2}$$

where $\mathbf{\Delta}$ is the $c \times c$ diagonal matrix with the eigenvalues of $\mathbf{C}$ and $\mathbf{C}$ is the $c \times c$ covariance matrix of $\mathbf{X}$ with values that are defined as

$$C_{jk} = \frac{1}{r-1}\sum_{i=1}^{r}(X_{ij} - \bar{X}_j)(X_{ik} - \bar{X}_k), \tag{3}$$

where $\bar{X}_j$ and $\bar{X}_k$ are the mean values of the jth and kth columns, respectively, of matrix $\mathbf{X}$. If $\mathbf{X}$ is a column-centered matrix, namely, $\bar{X}_j = \bar{X}_k = 0$, Formula (3) is simplified to the following form:

$$C_{jk} = \frac{1}{r-1}\sum_{i=1}^{r}X_{ij}X_{ik}, \tag{4}$$

which increases the computation speed.

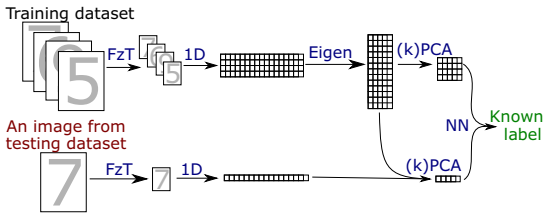

Fig. 1. Scheme of the herein proposed approach. A database of images is reduced by FzT, transformed to 1D, and arranged to a matrix. Using (k)PCA, most important eigenvectors are selected. Then, an image with an unknown label is again reduced by FzT, mapped to 1D, transformed by (k)PCA and by the usage of Nearest Neighbor, a class (label) is determined.

For the definition of $\mathbf{X}$, we consider r two-dimensional grayscale images that are specified by $f : \mathbb{N}^2 \to \mathbb{R}$. These two-dimensional functions must be transformed into one-dimensional space, namely, rows of matrix $\mathbf{X}$, e.g., by

$$\mathbf{X}_{i,yW+x} = f(x, y), \tag{5}$$

where we denote by W and H the image width and height, and $i = 1, \ldots, r$, $x = 1, \ldots, W$, and $y = 1, \ldots, H$. Hence, $c = WH$.

The main limitation of PCA is the requirement on the data distribution: it is applicable mainly to linearly separable data. In the case of non-linear data distribution, the use of non-linear generalization, which is called kernel PCA (kPCA) [20], is feasible. The main strategy behind kPCA is to use the *kernel trick* while considering a positive-definite kernel such as kernel that is based on RBF, or the cosine function.

More formally, we consider Φ to be a general mapping function into a feature space and a modification of Eq. (4) into the form

$$C_{jk} = \frac{1}{r-1} \sum_{i=1}^{r} \Phi(X_{ij}) \Phi(X_{ik}), \tag{6}$$

where, by using the kernel trick, we can avoid working in the higher dimensional feature space, namely, by defining an $r \times r$ matrix K as a product between the two mapping functions that are used in (6). Then, the remaining steps of PCA are modified analogously. The non-linearity property of kPCA is due to the possibly nonlinear map Φ. For a practical illustration, see Fig. 1 in [21].

Considering the Nearest Neighbor or Support Vector Machine classifier, PCA can be used as a part of a classification method. In our work, we are going to use the simpler classifier, namely the Nearest Neighbor classifier, where the classification relies on searching for

$$\arg\min_{i=1,\ldots,r} \| \mathbf{Y}_i - \mathbf{OP} \|, \tag{7}$$

where $\mathbf{O}$ with dimension $1 \times c$ is an unknown object (image) to be classified, and $\mathbf{Y}_i$ represents the i^{th} row of $\mathbf{Y}$ and assigns a known label of this row to $\mathbf{O}$. This approach can be easily modified by searching for k-Nearest Neighbors or by selecting various norms or cost functions - for additional details, see [22].

3.2. Preliminaries: F-transform

Through the years, the suitability of F-transform has been demonstrated for various image processing tasks, such as image compression, fusion, inpainting, classification, and recognition.

Below, we briefly introduce the theoretical foundation of the F-transform (FzT) [23]. The F-transform is an integral transform, that uses a fuzzy partition of a universe of discourse as a free parameter. For simplicity, we provide an explanation in which functions of one variable are used as originals.

The F-transform has two phases: direct and inverse. The direct F-transform (FzT) is applied to functions from $L^2(\mathbb{R})$ and maps them linearly onto sequences (originally finite) of numeric/functional components. Each component is a weighted orthogonal projection of a specified function on a linear subspace of $L^2(\mathbb{R})$. The inverse F-transform (iFzT) is applied to a sequence of components and transforms themn linearly into a function from $L^2(\mathbb{R})$. In general, it is different from the original one. In [23], it has been shown that the iFzT can approximate a continuous function with arbitrary precision.

3.2.1. Uniform fuzzy partition

We say that function $A : \mathbb{R} \to [0, 1]$ is a *generating function* of a uniform fuzzy partition, if it is non-negative, continuous, even, bell-shaped and moreover, it vanishes outside $[-1, 1]$ and fulfills $\int_{-1}^{1} A(t)\, dt = 1$. Generating function A produces infinitely many *rescaled* functions $A_H : \mathbb{R} \to [0, 1]$ with the scale factor $H > 0$, so that

$$A_H(t) = A\left(\frac{t}{H}\right).$$

An (h, H)-*uniform fuzzy partition* of $\mathbb{R}$ is a collection of translations $\{A_H(t - k \cdot h),\ k \in \mathbb{Z}\}$, where $0 < h < 2H$ is a step-value. The points $k \cdot h$, $k \in \mathbb{Z}$, are called *nodes* and particular partition elements $A_H(t - k \cdot h)$ are denoted by $A_k(t)$ and called *basic functions*[1]. If $h = H$, then the corresponding uniform fuzzy partition is called an h-uniform.

Let us remark that the name "fuzzy" in connection with the collection of basic functions $\{A_H(t - k \cdot h),\ k \in \mathbb{Z}\}$ appeared in [23] where the theory of F-transforms was proposed as a functional model of a particular set of fuzzy IF-THEN rules. Every considered basic function is a membership function of a fuzzy set whose support is the real line.

3.2.2. F^m-transform

Let us fix $[a, b] \subset \mathbb{R}$, generating function A, $n \geq 2$, and the h-uniform fuzzy partition $\{A_1, \ldots, A_n\}$, where $A_k(x) = A_h(x - x_k)$, $x_k = a + h(k-1)$, $k = 1, \ldots, n$, and $h = \frac{b-a}{n-1}$. Denote $L^2(A_k)$, $k = 1, \ldots, n$, a set of square-integrable functions on $[x_{k-1}, x_{k+1}]$ and consider a (weighted) inner product

$$\langle f, g \rangle_k = \int_{x_{k-1}}^{x_{k+1}} f(x) g(x) A_k(x) dx,$$

that makes space $L^2(A_k)$ a Hilbert one. Functions $f, g \in L^2(A_k)$ are *orthogonal* in $L^2(A_k)$, if $\langle f, g \rangle_k = 0$.

Let $m \geq 0$ and $L_m^2(A_k)$ a linear subspace of $L^2(A_k)$ spanned by restrictions of orthogonal polynomials P_k^0, P_k^1, $P_k^2 \ldots, P_k^m$ to the interval $[x_{k-1}, x_{k+1}]$. The orthogonal projection of an arbitrary function $f \in L^2(A_k)$ on $L_m^2(A_k)$ is equal to

$$F_k^m(f) = c_{k,0} P_k^0 + c_{k,1} P_k^1 + \cdots + c_{k,m} P_k^m, \tag{8}$$

where for all $i = 0, 1, \ldots, m$,

$$c_{k,i} = \frac{\langle f, P_k^i \rangle_k}{\langle P_k^i, P_k^i \rangle_k} = \frac{\int_{x_{k-1}}^{x_{k+1}} f(x) P_k^i(x) A_k(x) dx}{\int_{x_{k-1}}^{x_{k+1}} P_k^i(x) P_k^i(x) A_k(x) dx}. \tag{9}$$

The n-tuple $(F_1^m, \ldots, F_n^m)$ is an F^m-transform of f with respect to $\{A_1, \ldots, A_n\}$, or formally,

$$F^m[f] = (F_1^m(f), \ldots, F_n^m(f)).$$

$F_k^m(f)$ is called the k^{th} F^m-transform component of f, [5].

In particular case $m = 1$, the restrictions of orthogonal polynomials P_k^0 and P_k^1 to $[x_{k-1}, x_{k+1}]$ are: $1_k(x)$ and $x - x_k$, respectively,

[1] The condition $0 < h < 2H$ guarantees that every point from $\mathbb{R}$ is covered by a certain partition element A_k.

where by $1_k(x)$ we denote a constant function with the value 1 on the domain $[x_{k-1}, x_{k+1}]$. Then, the F^0-transform of f (or simply, FzT) of f with respect to the partition $\{A_1, \ldots, A_n\}$ is given by the n-tuple $(c_{1,0}1_1(x), \ldots, c_{n,0}1_n(x))$ of constant functions (0-degree polynomials) where for $k = 1, \ldots, n$,

$$c_{k,0} = \frac{\langle f, 1\rangle_k}{\langle 1, 1\rangle_k} = \frac{1}{h}\int_{x_{k-1}}^{x_{k+1}} f(x)A_k(x)dx. \tag{10}$$

We used to identify the F^0-transform of f with the n-tuple $(c_{1,0}, \ldots, c_{n,0})$.

The F^1-transform (or simply, Fz^{1}T) of f with respect to $\{A_1, \ldots, A_n\}$ is given by the n-tuple $(c_{1,0} + c_{1,1}(x - x_1), \ldots, c_{n,0} + c_{n,1}(x - x_n))$ of linear functions (1-degree polynomials). The latter are fully represented by their vectorial coefficients $((c_{1,0}, c_{1,1}), \ldots, (c_{n,0}, c_{n,1}))$, where $c_{1,0}, \ldots, c_{n,0}$ are the same as in (10) and $c_{1,1}, \ldots, c_{n,1}$ have the following representation:

$$c_{k,1} = \frac{\langle f, x - x_k\rangle_k}{\langle (x - x_k), (x - x_k)\rangle_k} = \frac{\int_{x_{k-1}}^{x_{k+1}} f(x)(x - x_k)A_k(x)dx}{\int_{x_{k-1}}^{x_{k+1}} (x - x_k)^2 A_k(x)dx}. \tag{11}$$

The following Proposition gives a semantical meaning of the F^1-transform components.

Proposition 1. *Let $A_1, \ldots, A_n$ be an h-uniform fuzzy partition of $[a, b]$. Let functions f and $A_k, k = 1, \ldots, n$, be four times continuously differentiable on $[a, b]$, and the F^1-transform of f with respect to $\{A_1, \ldots, A_n\}$ is identified with the vectorial representation $((c_{1,0}, c_{1,1}), \ldots, (c_{n,0}, c_{n,1}))$. Then for every $k = 1, \ldots, n$, the following estimation holds true:*

$$c_{k,0} = f(x_k) + O(h^2),$$
$$c_{k,1} = f'(x_k) + O(h^2).$$

Let us remark that according to (10) and (11), the F^0- and the F^1-transform components of f with respect to $\{A_1, \ldots, A_n\}$ are among values of normalized convolutions with the following kernels:

$$A_h(x - y), \quad (x - y)A_h^1(x - y),$$

where A_h is a rescaled generating function of the fuzzy partition.

3.2.3. F^0- and F^1-transform of images

In image processing, gray-scale images are often identified with their intensity functions, defined on a discrete domain. In [24], we introduced the F-transform of a discrete function, and below, we repeat formal expressions of F^0- and F^1-transform components related to it. These expressions are discrete versions of (10) and (11). After obvious reformulation, all above given properties of the F-transform components are valid for discrete functions.

We assume that a gray-scale image $f : D \to \mathbb{R}$ has domain $D = \{1, \ldots, W\} \times \{1, \ldots, H\}$ where W, H are integers, and $[1, W]$ and $[1, H]$ are corresponding real line intervals. Let $\{A_1, \ldots, A_m\}$ and $\{B_1, \ldots, B_n\}$ establish a fuzzy partition of $[1, W]$ and $[1, H]$ correspondingly, where $m < W$, $n < H$, $h = W/m = H/n$. Let moreover,

$$(\forall k = 1, \ldots, m)(\forall \ell = 1, \ldots, n)\ (\exists (i, j) \in D)\, A_k(i)B_\ell(j) > 0.$$

The direct F^0-transform of f w.r.t. the chosen partition is a $m \times n$ matrix $\mathbf{F} = (F_{k\ell})$ of components where

$$F_{k\ell} = \frac{\sum_{i=1}^{W}\sum_{j=1}^{H} f(i, j) \cdot A_k(i)B_\ell(j)}{\sum_{i=1}^{W}\sum_{j=1}^{H} A_k(i)B_\ell(j)}. \tag{12}$$

The direct F^1-transform of f w.r.t. the chosen partition is a $m \times n$ matrix $\mathbf{F}^1 = (F_{k\ell}^1)$ of vectorial components where

$$F_{k\ell}^1 = \left\langle c_{k\ell}^{00}, c_{k\ell}^{10}, c_{k\ell}^{01}\right\rangle, \tag{13}$$
$$c_{k\ell}^{00} = F_{k\ell},$$
$$c_{k\ell}^{10} = \frac{\sum_{i=1}^{W}\sum_{j=1}^{H} (i - i_k) f(i, j) \cdot A_k(i)B_\ell(j)}{\sum_{i=1}^{W}\sum_{j=1}^{H} (i - i_k)^2 A_k(i)B_\ell(j)},$$
$$c_{k\ell}^{01} = \frac{\sum_{i=1}^{W}\sum_{j=1}^{H} (j - j_\ell) f(i, j) \cdot A_k(i)B_\ell(j)}{\sum_{i=1}^{W}\sum_{j=1}^{H} (j - j_\ell)^2 A_k(i)B_\ell(j)},$$

and i_k (j_ℓ) are elements in D such that $A_k(i_k) = 1$ ($B_\ell(j_\ell) = 1$).

Similar to Proposition 1, we can show that $c_{k\ell}^{00}, c_{k\ell}^{10}, c_{k\ell}^{01}$ are approximate values of f, $\frac{\partial f}{\partial x}$, $\frac{\partial f}{\partial y}$, correspondingly, at node (i_k, j_ℓ), see [25,26]. This fact justifies the use of $\mathbf{F}$ (or $\mathbf{F}^1$) as a low dimensional representation of a corresponding image.

3.3. F-transform + PCA classification algorithm

The objectives of our work are to identify challenges that are encountered with (k)PCA and to propose strategies for overcoming them. First, solving (2) has the complexity of $\mathcal{O}(rc^2)$ and solving (4) has the complexity of $\mathcal{O}(c^{2.376})$; the reduction of the dimensionality c of the input data before PCA is executed may lead to a substantial savings of computation time. Moreover, as we discussed in the previous section, PCA performs well for data with linearly separable classes. Non-linearly separable classes can be handled by kernel PCA, which has higher memory requirements; hence, dimensionality reduction prior to kPCA seems to be relevant also in this case. To address the issue, we propose computing F-transform components and using them as the input to the (k)PCA method, as illustrated in Fig. 1.

Formally, let us consider r input images g_i, $i = 1, \ldots, r$, that are projected using FzT into the space of F-transform components $\mathbf{F}_i$ and are rearranged into 1D image functions f_i, which serve as input to the PCA algorithm. While the input images have a resolution of $W \times H$ pixels, the new reduced images have resolutions of $W/h \times H/h$ pixels in the case of FzT and $3W/h \times 3H/h$ pixels in the case of Fz^{1}T. Processing such reduced images with (k)PCA will lead to significant computation time improvement because establishing a covariance matrix has quadratic complexity according to the input data dimension. The direct F-transform is fast (it has linear complexity) and preserves important information [27] in an image.

Via the PCA orthogonal projection and the selection of eigenvalues, we can reduce the dimensionality and maximize the variance in the output dimensions. This process is valuable because by omitting minor eigenvalues, we improve computation speed and suppress possible noise or other distortions in input images. However, when the distance is not sufficiently large (in the orthogonal space) between our classes, we can suppress also important details, and our algorithm may fail. A natural property of the F-transform is noise filtering ability; therefore, distortions have already been suppressed in our reduced images.

Further, F-transform computes each component based on a fuzzy partition, which is determined by parameter h, namely, each component value captures its $(2h + 1)^2$ pixel neighborhood, which is valuable for a dataset in which images are partially shifted along the x and y axes. The process of capturing a local neighborhood by a single value is similar to the pooling technique that is used in Neural Networks, which is used to increase the robustness against shifts and rotations.

Based on the above, we propose utilizing the two methods to realize F-transform-based dimensionality reduction on the input images for the computation of a matrix of components and to perform the PCA orthogonal projection onto the matrix. The way, in which the FzT and PCA algorithms are connected is described in Algorithm 1 for the training phase and in Algorithm 2 for the classification (testing) phase. The algorithm is simple to modify for kPCA with the use of (6); hence, we will not provide a full de-

Algorithm 1 FzT+PCA training / pre-processing phase.

Inputs:

1: 2D-images f_i, $i = 1, \ldots, r$, with resolution of c pixels
2: m determining number of eigenvalues
3: h parameter controlling fuzzy partition

Outputs:

1: Transformed matrix $\mathbf{Y}$ with size $r \times m$
2: Eigenvector matrix $\mathbf{P}$ with size $c \times m$

Steps:

1: **for** $i = 1, \ldots, r$ **do**
2: Compute $\mathbf{F}_i$ of f_i using (12) w.r.t. h
3: Arrange $\mathbf{F}_i$ to $\mathbf{X}$ as its i^{th} row
4: Compute $\mathbf{C}$ of $\mathbf{X}$ using (3)
5: Compute $\mathbf{P}'$ of $\mathbf{C}$ using (2)
6: Extract $\mathbf{P}$ from $\mathbf{P}'$ according to m
7: Compute $\mathbf{Y} = \mathbf{XP}$

Algorithm 2 FzT+PCA classification phase.

Inputs:

1: f with resolution of c pixels to be classified
2: h parameter controlling fuzzy partition
Note: it must be equal to h used in the training phase
3: $\mathbf{Y}$ taken from the training phase
4: $\mathbf{P}$ taken from the training phase
5: $\mathbf{L}$ vector with known object labels

Output:

1: Known label

Steps:

1: Compute $\mathbf{F}'$ of f using (12) w.r.t. h
2: Arrange $\mathbf{F}'$ to $\mathbf{X}'$
3: Compute $\mathbf{Y}' = \mathbf{X}'\mathbf{P}$
4: **for** $i = 1, \ldots, r$ **do** $D_i = \| \mathbf{Y}_i - \mathbf{Y}' \|$
5: Return L_ℓ, $\ell = \arg\min_{i=1..r} D_i$

scription here. The same formulas are used as for the computation of the Fz^{1}T components, which are presented as (23) and (24) in [26].

For completeness, we built a Windows implementation of our proposed method and made all the source codes available online[2] in the form of a console application. The application has been written using C++, the Qt framework, and the Eigen library[3], which is used for eigenvector computation. The application includes both the training and classification phases as they are described here, and it enables a user to evaluate the algorithm success rate and the computation speed on his own dataset and hardware.

4. Benchmark

To evaluate the performance of the proposed algorithm, we split the benchmark into seven parts. First, we describe the used dataset in Section 4.1 and continue with the description of the settings of the tested algorithms in Section 4.2. The main experiment is conducted in Sections 4.3 and 4.4 for the expression of the dependency on the used components of the accuracy and for numerical comparison, in terms of the accuracies and computation times, respectively. The additional datasets are tested in Section 4.5. Finally, the results are discussed in Section 4.7 and several dead ends that we encountered are discussed in Section 4.8.

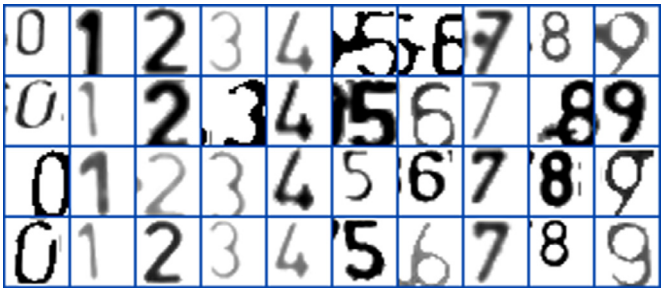

Fig. 2. Examples of images from the dataset used in the benchmark.

4.1. Dataset and the experiment configuration

We created a dataset of 1000 training and 63,455 testing grayscale images with a resolution of 26 × 28px. The dataset task is the classification of digits 0–9 numbers; thus, for the training phase, only 100 samples per class are available. In contrast to the well-known MNIST dataset, the numbers vary in size and intensity, are damaged, and are not centered. Images of several numbers are shown in Fig. 2. The dataset is also available online[4].

The dataset is used in our benchmark for all except the NN-based algorithms in two stages. First, an unsupervised phase is conducted, namely, a new, reduced representation is developed for the 1000 training images. Then, the testing images are transformed into the same representation, and the nearest neighbor is searched. Finally, the labels are revealed, and whether the label of a test image is the same as the label of its nearest neighbor is determined. The success rate is computed as a percentage ratio between the equivalent labels and the number of test images. For the Autoencoder-based algorithms, the labels are already available in the training phase. All experiments were conducted on a MacBook Pro notebook that was equipped with an i5-7267U @3.1 GHz processor and 16GB RAM.

4.2. Algorithms and their settings

For the benchmark, we test 15 algorithms, namely: PCA, kPCA, FzT+PCA, FzT+kPCA, Fz^{1}T+PCA, Fz^{1}T+kPCA, Isomaps, LLE, HLLA, LTA, AE, CAE, VAE, Siamese network, and ARC. For the FzT-based algorithms, we use the configuration of raised cosine basic function and $h = 4$, which leads to 20 components for FzT-based algorithms and 60 components for Fz^{1}T algorithms. The manifold-based algorithms accept as input the number of neighboring points for a projection. We tested several settings and chose the following ones, which yielded the best results: Isomaps: 30, LLE: 30, HLLE: 500, and LTSA: 60. To avoid possible inaccuracies in the implementations of the manifold-based algorithms, we used their tested implementations through the Scikit-learn library[5]. As a classifier, we used a simple nearest neighbor classifier in all the cases that are discussed above. For the autoencoders, we used the architectures that are listed in Table 1; hence, the autoencoder has input/output and nine hidden layers, the convolutional autoencoder has five hidden layers (plus six down/upsampling layers), and the variational autoencoder has two hidden layers. The siamese network has three fully connected layers with 128 neurons per layer. Furthermore, each layer is followed by a dropout layer to avoid overfitting. To train the network, the contrastive loss is utilized. Attentive recurrent comparators (ARCs) utilize glimpses to creates representations of training samples. The whole network uses a (dis)similarity measure for learning and to correctly classify new images. The authors

[2] graphicwg.irafm.osu.cz/storage/ft1-pca-app.zip.
[3] eigen.tuxfamily.org/index.php.
[4] http://graphicwg.irafm.osu.cz/storage/on-a-fast-classification-dataset.zip.
[5] https://scikit-learn.org/stable/modules/manifold.html.

Table 1
Architectures of autoencoders used in the benchmark. VAE coded representation is special composition that learns Gaussian distribution parameters used for sampling.

Autoencoder

layers	input	encoder-1	encoder-2	encoder-3	encoder-4	coded rep.	decoder-1	decoder-2	decoder-3	decoder-4	output
neurons	728	512	256	128	64	32	64	128	256	512	728

Convolutional Autoencoder

layers	input	encoder-1		encoder-2		coded rep.		decoder-1		decoder-2		output	
		conv	max pool	conv	max pool	conv	max pool	upsampl.	conv	upsampl.	conv	upsampl.	conv
kernel size	–	3x3	2x1	3x3	1x13	3x3	2x1	2x1	3x3	1x13	3x3	2x1	3x3
channels	1	16	16	8	8	4	4	4	4	4	8	8	1
neurons	28x26	–	–	–	–	–	–	–	–	–	–	–	–

Variational Autoencoder

layers	input	encoder-1	coded rep. sampling		decoder-1	output
	–	–	mean	std. dev.	–	–
neurons	728	256	30		256	728

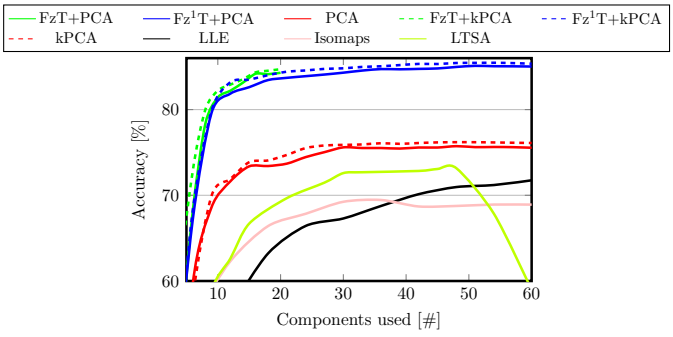

Fig. 3. The dependency of classification success rate on the number of used components for the nine evaluated algorithms.

used ARC in the problem of one-shot learning on database Omniglot and realized state-of-the-art performance [28].

All autoencoders were trained with a batch size of 128 in 300 epochs and with a decreasing learning rate on a plateau. After an autoencoder is trained, in a second model, the encoder is fine-tuned. This model has the following architecture: dense(728) → dropout(0.2) → dense(10), which is the classification layer; and the model is trained with a batch size of 128 in 50 epochs and with a decreasing learning rate on a plateau. We use the sigmoid function in the last autoencoder layer and the softmax function in the last classifier layer; in all other layers, we use ReLU. The best combination of optimizers was identified as Adadelta for the autoencoder and Adam for the classifier. The autoencoder minimizes the binary cross-entropy, and the classifier minimizes the categorical cross-entropy. The source codes of all three networks are available online[6].

4.3. Dependence of the success rate on number of used components

The first experiment that we conducted examines the dependence on the number of used components of the algorithm success rate. This experiment should also determine whether if it is better to realize single-step dimensionality reduction using only (k)PCA or two-step dimensionality reduction using FzT+(k)PCA.

According to the graph in Fig. 3, FzT+(k)PCA significantly outperforms (k)PCA – with approximately 10% higher accuracy for 20 eigenvalues. For completeness, we add into the graph the three discussed state-of-the-art methods for searching for ideal transformation from a high-dimensional manifold to a low-dimensional projection. During the experiment, we recognized complication with the HLLE algorithm and its memory complexity. We were not able to test the algorithm because even 16 GB of RAM was not sufficient; therefore, the algorithm is not included in the graph.

From the results of the experiment, it is concluded that FzT/Fz^{1}T + (k)PCA significantly outperforms the other tested algorithms.

[6] graphicwg.irafm.osu.cz/storage/sources-fast-class-small.ipynb.

Table 2
The comparison of the accuracy and computation times for the tested algorithms, in descending order according to the accuracy. Repository for Siamese: keras.io/examples/mnist_siamese/ Repository for ARC: github.com/Raverss/arc-pytorch.

Algorithm	Acc [%]	Train [s]	Test [s]
Fz^{1}T + kPCA	85.46	0.16	14.33
Fz^{1}T + PCA	85.02	0.02	11.94
CAE [9]	84.97	373.86 + 35.69	17.48
FzT + kPCA	84.70	0.12	5.76
Siamese	84.48	33.49	2.56
FzT + PCA	84.29	0.01	4.42
AE	80.40	120.93 + 12.31	6.78
VAE [11]	79.23	41.78 + 6.61	4.99
kPCA	76.19	0.20	10.49
PCA	75.73	0.10	7.43
LTSA [15]	73.16	2.10	116.81
ARC	72.22	>10,000	>10,000
LLE [13]	71.74	1.87	121.22
Isomaps [12]	69.45	2.45	106.10

4.4. Numerical comparison

The numerical comparison aims at measuring the algorithms' accuracies, together with their training and classification times. We used results from the previous experiment and selected the number of components, with which the best accuracy was realized. The detailed numerical results are presented in Table 2. We executed the NN-based approaches several times and selected the best result. For the same reason as in the previous experiment, the HLLE method is not evaluated.

According to the results in the table, we obtained a similar result to that in [8], namely, AE outperforms PCA significantly. Furthermore, CAE outperforms VAE, which is also natural because CAE considers the 2D structure of the data. Surprisingly, LTSA, LLE, and Isomap were outperformed by basic PCA. Among numbers of components from 1 to 60 (for which we evaluated the accuracy), LTSA

obtained the best result with 48 components and subsequently began to decrease rapidly; Isomaps obtained the best result with 36 components and subsequently decreased slowly; LLE tended to increase even at the end of the interval. Therefore, we examined the whole interval (0–728 components) with the best obtained result of 73.89% accuracy with 100 components, which is still slightly worse than the result of PCA.

The accuracy of autoencoders can be regarded as the state-of-the-art accuracy: if there are no FzT/Fz^{1}T+(k)PCA combinations, CAE would perform the best. Unfortunately, its training time is high, however, it can be massively improved via the use of more powerful hardware (GPU). We present two values for the autoencoder training times. The first value is for the end-to-end autoencoder training, and the second value is for the classifier training. Finally, we conclude that Fz^{1}T+kPCA outperforms all algorithms, while its training phase is more than 3400 $\times$ faster than the training phase of CAE. Such low computational complexity renders the proposed method a promising candidate for execution on embedded devices, which can also be valuable in the industry.

4.5. Additional datasets

For comparison with the previous results, we selected two additional, well-known, datasets: the AT&T eigenface dataset[7] and fashion-mnist.[8] The former dataset includes forty people, each of whom is captured in ten images of resolution 92 $\times$ 112px. We used two images of each person for training and the remaining images for testing. These conditions are challenging for neural networks when we do not utilize massive augmentation or transfer learning. That impacts the results: AE realized an accuracy of 13.1%, VAE 3.1%, and CAE 75.6%. The best result from the proposed combinations was realized by FzT+PCA, with an accuracy of 77.5%.

For the latter dataset, namely, Fashion Mnist, we selected 1000 images for training and 60,000 for testing. The resolution of the images is 28 $\times$ 28 px, and the content corresponds to ten classes of fashion items, such as boots, t-shirts, clothes, etc. In this dataset, the result is the opposite - the convolutional autoencoder realized an accuracy of 76.5%, while the best proposed method – Fz^{1}T+kPCA – realized a lower accuracy of 73.97%.

That behavior demonstrates the limitations of the proposed approach. The proposed approach realizes recognition over the intensity domain, while neural networks realize transformations of a domain and capture shapes. Therefore, the proposed method performs well on simple datasets, where it depends on the intensity. The robustness of the proposed method can be improved via the use of a nonlinear classifier such as SVM.

4.6. Comparison with generalized robust regression

We would like to mention the exciting work of Zhihui Lai [29,30] dealing with jointly sparse learning. Lai proposes to avoid the problems of global descriptors and sensitivity to outliers (due to L_2 norm) by his Generalized Robust Regression (GRR) – an iterative optimizing algorithm on the basis of $L_{2,1}$-norm. The paper results clearly shows good results with high accuracies in various tasks. We repeated YALE faces classification experiment from [29], using the author's source codes[9]. By selecting three faces per person for the training set, GRR reached the classification accuracy of 85.8%. For the comparison, we performed Fz^{1}T+PCA that achieved significantly lower accuracy of 75.0%. In terms of a computational complexity, GRR has $\mathcal{O}(Td^3)$, where T denotes the number of iterations and d the dimension of input data; hence, the author applies PCA before GRR in order to decrease the input data dimensionality. As we stressed before, PCA computational complexity is high too. Therefore, the future work lies in combination of FT+PCA with GRR: our proposed method ensures high computational speed while GRR provide robust learning.

[7] www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.
[8] github.com/zalandoresearch/fashion-mnist.
[9] http://www.scholat.com/laizhihui.

Table 3
The impact of various known resampling methods on the tested dimensionality reduction + classification algorithm.

	Acc [%]				
Preprocessing	PCA	kPCA	LLE	LTSA	Isomaps
FzT	84.29	84.70	80.16	80.92	80.11
Subsampling	50.74	50.87	49.73	47.81	49.27
Bilinear	52.18	52.23	49.23	47.15	49.12
Bicubic	50.13	49.96	47.63	45.98	47.74
Lanczos	49.88	49.76	47.42	47.93	47.25

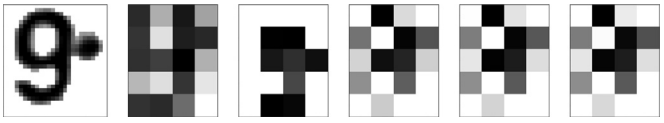

Fig. 4. Downsampled image with number '9' using various resampling methods. From left: original, FzT, Subsampling, Bilinear, Bicubic, Lanczos.

Table 4
The accuracy of preprocessing realized by FzT/Fz^{1}T in the combination with autoencoders.

Preprocessing	AE [%]	CAE [%]	VAE [%]
FzT	81.62	56.90	33.00
Fz^{1}T	83.25	79.92	39.44

4.7. Discussion

One may argue that the F-transform is one of many generic reduction methods and can be replaced by an arbitrary method. We refer the reader to [27], where it has been shown that F-transform can reduce an image while preserving the most important details among many rescaling methods. Moreover, these methods are comparable only with FzT and not with Fz^{1}T, which also provides information about the gradient magnitude. For completeness, we also conducted an experiment in which Bilinear, Bicubic, and Lanczos resampling were utilized; the results are presented in Table 3. The values in the table were computed as follows. A resampling method is selected, and it is used to resample all dataset images to the resolution of 5 $\times$ 4px, namely, the same number of values as for the FzT method. Then, the methods are applied, and the accuracy analyzed.

The results that are presented in Table 3 are much poorer than those of the FzT+(k)PCA combination. This is because the considered standard resampling methods cannot downsample the original image to such low resolution "properly", according to Fig. 4. We conducted the downsampling three times using three software programs to ensure that it is correct.

Based on the results that are presented in Section 4.4, we also tested combinations of FzT with autoencoders; the results are presented in Table 4. We observe that in the case of AE, the results have been improved. However, the CAE and VAE versions show much poorer performances which is because these two architecture are more sensitive to data size and the reduced representation is not sufficiently large.

The next point we want to discuss is the design of the autoencoders. Typically, when an expert designs an NN architec-

ture (where no pre-trained model is available) and encounters the problem of an insufficient amount of data, he/she can create artificial samples, e.g., by augmenting the available data. For the completeness, we also augmented the training images using Keras ImageDataGenerator[10]. Because we are dealing with images, only testing of CAE is meaningful. We tested many combinations of augmentation and ascertained that the best result is obtained when only shifting (in the interval of 0–0.1 of the width and height of the images) is used. This setting perfectly captures the dataset attribute of unaligned images and will not lead to the improvement of the results. With this augmentation, we realized a the maximum accuracy of 87.57%, however, data augmentation increased the autoencoder training time by 55%.

Regarding the size of the file where the configurations from the training phase were saved, which are necessary for the inference, the best algorithm, namely, Fz[1]T+kPCA, produces a file of size 236 kB. In comparison, CAE produces an output with twice the size, namely, 554 kB and AE produces an incomparably large file of size 5.9 MB.

4.8. Dead ends

The proposed algorithm realized the highest classification accuracy among all the tested state-of-the-art algorithms; however, it has drawbacks that should be addressed in future work. The main drawback originates from the PCA method, namely, from establishing a covariance matrix via Eq. (3). The data should be aligned to yield a high covariance for inner-class data and a low covariance for outer-class data. That is also why the Eigenface database [31] is so overused for PCA - because the data are aligned. In our experiment, the data are sometimes shifted (see Fig 2); this is the main reason why our proposed method did not realize even higher accuracy. To further weaken the alignment requirement, we addressed several aspects of pre-processing.

The first is a modified Discrete Cosine Transform (DCT) of type II, which we applied over the whole image instead of 8×8 blocks. After DCT II transforms an image from a spatial domain into a frequency domain, we use a zig-zag scheme to obtain a vector of sorted frequencies, such that the lower index of the vector is, the lower the frequency it describes. Unfortunately, even though the frequencies in the output vector are aligned, mainly higher frequencies are sensitive to data shifting. Therefore, this approach did not yield any improvement.

5. Conclusions

In the study, we investigated the state-of-the-art methods for unsupervised learning and selected PCA, together with its kernel version, as the baseline. For the baseline, we identified the main disadvantages: high memory requirements and high computational complexity. To overcome these disadvantages, we proposed F-transform preprocessing, which reduces the input data dimensionality via transformation into a new representation, namely, a matrix of components. The reduced representation serves as a new input of (k)PCA. We experimentally proved the following: the F-transform improves the performance of (k)PCA significantly. It is the only suitable preprocessing method among the frequently used. The proposed combination outperforms state-of-the-art approaches. The proposed method reached the highest accuracy with real-time training and fast prediction time, followed by Convolutional Autoencoder, for which the training time was more than 3400 $\times$ longer. Finally, we discuss dead ends that we encountered when using additional preprocessing transformations intended to overcome the main disadvantages of the proposed method, such as sensitivity to larger shifts and rotations of dataset images. Overcoming this disadvantage is left for future work.

[10] https://keras.io/preprocessing/image/.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This research was supported by the project “LQ1602 IT4Innovations excellence in science”.

References

[1] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, T. Raiko, Semi-supervised learning with ladder networks, in: Advances in Neural Information Processing Systems, 2015, pp. 3546–3554.
[2] Q.V. Le, Building high-level features using large scale unsupervised learning, in: Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, IEEE, 2013, pp. 8595–8598.
[3] P. Hurtik, I. Perfilieva, Fast training and real-time classification algorithm based on principal component analysis and f-transform, in: 2018 Joint 10th International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th International Symposium on Advanced Intelligent Systems (ISIS), IEEE, 2018, pp. 275–280.
[4] I.T. Jolliffe, Principal component analysis and factor analysis, in: Principal Component Analysis, Springer, 1986, pp. 115–128.
[5] I. Perfilieva, M. Daňková, B. Bede, Towards a higher degree F-transform, Fuzzy Sets Syst. 180 (1) (2011) 3–19.
[6] G. Cheng, P. Zhou, J. Han, Learning rotation-invariant convolutional neural networks for object detection in VHR optical remote sensing images, IEEE Trans. Geosci. Remote Sens. 54 (12) (2016) 7405–7415.
[7] J. Han, D. Zhang, G. Cheng, L. Guo, J. Ren, Object detection in optical remote sensing images based on weakly supervised learning and high-level feature learning, IEEE Trans. Geosci. Remote Sens. 53 (6) (2014) 3325–3337.
[8] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.
[9] J. Masci, U. Meier, D. Cireşan, J. Schmidhuber, Stacked convolutional auto-encoders for hierarchical feature extraction, in: International Conference on Artificial Neural Networks, Springer, 2011, pp. 52–59.
[10] A. Makhzani, B.J. Frey, Winner-take-all autoencoders, in: Advances in Neural Information Processing Systems, 2015, pp. 2791–2799.
[11] Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, L. Carin, Variational autoencoder for deep learning of images, labels and captions, in: Advances in Neural Information Processing Systems, 2016, pp. 2352–2360.
[12] V.D. Silva, J.B. Tenenbaum, Global versus local methods in nonlinear dimensionality reduction, in: Advances in Neural Information Processing Systems, 2003, pp. 721–728.
[13] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.
[14] D.L. Donoho, C. Grimes, Hessian eigenmaps: locally linear embedding techniques for high-dimensional data, Proc. Natl. Acad. Sci. 100 (10) (2003) 5591–5596.
[15] Z. Zhang, H. Zha, Principal manifolds and nonlinear dimensionality reduction via tangent space alignment, SIAM J. Sci. Comput. 26 (1) (2004) 313–338.
[16] J.B. Tenenbaum, V. De Silva, J.C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (5500) (2000) 2319–2323.
[17] L. Maaten, G. Hinton, Visualizing data using t-sne, J. Mach. Learn. Res. 9 (Nov) (2008) 2579–2605.
[18] J.B. Kruskal, Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis, Psychometrika 29 (1) (1964) 1–27.
[19] A.Y. Ng, M.I. Jordan, Y. Weiss, On spectral clustering: Analysis and an algorithm, in: Advances in Neural Information Processing Systems, 2002, pp. 849–856.
[20] B. Schölkopf, A. Smola, K.-R. Müller, Kernel principal component analysis, in: International Conference on Artificial Neural Networks, Springer, 1997, pp. 583–588.
[21] B. Schölkopf, A. Smola, K.-R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Comput. 10 (5) (1998) 1299–1319.
[22] K.Q. Weinberger, L.K. Saul, Distance metric learning for large margin nearest neighbor classification, J. Mach. Learn. Res. 10 (Feb) (2009) 207–244.
[23] I. Perfilieva, Fuzzy transforms: theory and applications, Fuzzy Sets Syst. 157 (8) (2006) 993–1023.
[24] I. Perfilieva, P. Vlašánek, F-transform and discrete convolution, in: Proceedings of the 2015 Conference of the International Fuzzy Systems Association and the European Society for Fuzzy Logic and Technology, 89, 2015, pp. 1054–1059.
[25] P. Hodáková, I. Perfilieva, F1 -transform of functions of two variables., in: 8th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT-13), Atlantis Press, 2013, pp. 547–553.

[26] I. Perfilieva, P. Hodáková, P. Hurtik, Differentiation by the f-transform and application to edge detection, Fuzzy Sets Syst. 288 (2016) 96–114.
[27] I. Perfilieva, P. Hurtik, F. Di Martino, S. Sessa, Image reduction method based on the f-transform, Soft Comput. 21 (7) (2017) 1847–1861.
[28] P. Shyam, S. Gupta, A. Dukkipati, Attentive recurrent comparators, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org, 2017, pp. 3173–3181.
[29] Z. Lai, D. Mo, J. Wen, L. Shen, W.K. Wong, Generalized robust regression for jointly sparse subspace learning, IEEE Trans. Circuits Syst. Video Technol. 29 (3) (2018) 756–772.
[30] D. Mo, Z. Lai, Robust jointly sparse regression with generalized orthogonal learning for image feature selection, Pattern Recognit. 93 (2019) 164–178.
[31] L. Sirovich, M. Kirby, Low-dimensional procedure for the characterization of human faces, J. Opt. Soc. Am. A 4 (3) (1987) 519–524.

Petr Hurtik has received Ph.D. (2017) in applied mathematics from the University of Ostrava. From 2007 to 2012 he worked as a coder in several IT companies and since 2012, he has been a member of IRAFM where he has worked in prof. Perfilieva's image processing team.

Vojtech Molek has received Mgr. (2015) in applied computer science from the University of Ostrava. He has started to cooperate with IRAFM in 2014 and later has become part of the prof. Perfilieva's image processing team. His main research focuses on Deep learning.

Irina Perfilieva has received Ph.D. (1980) in Applied Mathematics from the Lomonosov State University in Moscow. At present, she is a full professor of Applied Mathematics in University of Ostrava. At the same time, she is a head of Theoretical Research Department in the University of Ostrava, IRAFM.