sensors

MDPI

Article

Classification Algorithm for Person Identification and Gesture Recognition Based on Hand Gestures with Small Training Sets

Krzysztof Rzecki

AGH University of Science and Technology, 30 Mickiewicz Ave., 30-059 Kraków, Poland; krz@agh.edu.pl

Received: 19 October 2020; Accepted: 15 December 2020; Published: 18 December 2020

Abstract: Classification algorithms require training data initially labelled by classes to build a model and then to be able to classify the new data. The amount and diversity of training data affect the classification quality and usually the larger the training set, the better the accuracy of classification. In many applications only small amounts of training data are available. This article presents a new time series classification algorithm for problems with small training sets. The algorithm was tested on hand gesture recordings in tasks of person identification and gesture recognition. The algorithm provides significantly better classification accuracy than other machine learning algorithms. For 22 different hand gestures performed by 10 people and the training set size equal to 5 gesture execution records per class, the error rate for the newly proposed algorithm is from 37% to 75% lower than for the other compared algorithms. When the training set consists of only one sample per class the new algorithm reaches from 45% to 95% lower error rate. Conducted experiments indicate that the algorithm outperforms state-of-the-art methods in terms of classification accuracy in the problem of person identification and gesture recognition.

Keywords: biometrics; classification; gesture recognition; one-shot learning; person identification; small training sets

1. Introduction

Classification algorithms, an important tool in Computational Intelligence Methods and Statistical learning [1–3], are widely used in many areas, for example biometrics, economic trend analysis, human-computer interfaces, medical diagnostics, etc. These methods include random forest [3], *k*-nearest neighbour (*k*NN) [4], probabilistic neural network (PNN) [5], multi-layer perceptron (MLP) [6], support vector machine (SVM) [7], Gaussian processes [8], adaptive neuro-fuzzy inference system (ANFIS) [9], decision trees [3], radial basis function-based neural network (RBF NN) [10], generalized regression neural network (GRNN) [5] as well as siamese neural network (SNN) [11].

The common way to construct a person identification or gesture recognition system based on hand gestures is to collect, with the respect to overfitting, as a large database, as required to reach satisfactory values of the classification coefficients accompanying the receiver operating characteristic (ROC) curve. During the research, the system development or the deploying in real application, when users are volunteers, collecting gestures for predefined gesture recognition tasks is limited only by technical or algorithmic capabilities. In the case of person identification or personalized gesture recognition systems for use in real life it is not so obvious. First of all, the real user may not be so patient, or may be an elder person, or be a disabled person which limits the possibility to record a large number of repetitions of a single gesture, and then the biometric acceptance factor [12] might be lowered. Secondly, in the latest literature [13–17] the researchers point out the necessity to develop customizable gesture recognition systems, where a user can define her/his own gestures. In those systems the small training sets will allow for much quicker introduction of new gestures to be recognized and offer a potential for better user experience in end-user gesture customization. Gesture customization is also important for the most of motor impaired people having heavy movement constraints who for this reason may not be able to perform certain gestures defined by the manufacturer of the system [18,19]. In those systems the small training sets prepared individually for each user will allow the system to be used at all. The small training sets in gesture recognition research approach was already investigated, e.g., for spiking neural networks algorithm [20] and hand gesture recognition with a depth sensor concept [21].

The main contribution of the paper is a novel time series classification algorithm for person identification and gesture recognition where classification model is built using training sets containing a very limited number of gesture repetitions. The algorithm is based on k-means and kNN algorithms and comparisons based on the vector space model (VSM) [22].

The algorithm was tested and compared to other ones in the exemplary area of human-computer interaction based on hand gestures. A typical human-computer communication using hand glove gestures [23] can be split into two stages: person identification or verification to get access to a computer system and then the gesture recognition to issue commands for this system. Data acquisition for these tasks can be performed using a specialized hand glove [23], which records gestures as time series of data from sensors mounted on it, like accelerometers, gyroscopes and fingers flexion measurement.

The raw data from gesture recordings devices may not be directly suitable for classification or have noisy features causing low classification accuracy. There are some works where authors develop preprocessing methods to improve classification algorithms, for example by using specialized feature extraction algorithms [24] or by applying functional statistical methods [25–27]. The emotional state of the person performing a gesture is another source of variability that should be accounted for [28]. There is also some work on using other modalities for gesture recognition, for example vision-based systems [29,30], touchscreen-based methods [31], impedance tomography [32,33], micro-Doppler signatures [34] or controllers like Kinect [35] or LeapMotion [36,37].

Section 2 describes the mathematical model of the data used to present in Section 3 the new algorithm. Section 4 shows the design of the experiments conducted for this study. The results are in Section 5 and they are discussed in Section 6. Finally, Section 7 contains conclusions.

2. Mathematical Model

A data sample can be described as a multivariate time series of a number of variables. Values of the variables at the same moment in time constitute an observations. I assume that a data set of such samples is given. The samples are attributed to classes distinguishable by unknown characteristics of the samples. The following description of the algorithm assumes for clarity that samples have an equal number of observation but it can be easily extended to samples having a varying number of observations.

The mathematical model for data representation is described by the following symbols:

- V—number of variables
- v—variable index $(1, 2, \ldots, V)$
- D—number of observations in a sample
- i—observation index in a sample $(1, 2, \ldots, D)$
- J—number of all samples in the data set
- j—sample index in data set $(1, 2, \ldots, J)$
- $x^v_{j,i}$—the value of the ith observation of vth variable and jth sample.
- C—number of classes
- $c(j)$—label of the class the jth sample belongs to $(1, 2, \ldots, C)$

2.1. Data Definition

An observation is a list of values of all variables at the same moment in time and is defined as:

$$\Theta_{j,i} = (x_{j,i}^v)_{v=1}^V = (x_{j,i}^1, x_{j,i}^2, \dots, x_{j,i}^V). \tag{1}$$

A single sample consists of multiple observations as in Equation (2) read at regular time intervals and is represented by a matrix:

$$\Pi_j = \begin{bmatrix} x_{j,1}^1 & x_{j,1}^2 & \dots & x_{j,1}^V \\ x_{j,2}^1 & x_{j,2}^2 & \dots & x_{j,2}^V \\ \vdots & \vdots & \ddots & \vdots \\ x_{j,D}^1 & x_{j,D}^2 & \dots & x_{j,D}^V \end{bmatrix}. \tag{2}$$

Each row of the matrix Π_j corresponds to observation while each column represents a different variable. This notation is used to describe the algorithm presented in this work.

2.2. Data Sets

Using the data sets of samples described in Section 2.1 we can define a classification problem which consists of assigning a new sample to one of the predefined classes.

The data set for this task is given by:

$$T = \{(\Pi_j, c(j)) \colon j \in \{1, 2, \dots, J\}\} \tag{3}$$

The set of indices of the training samples which are used to build classification function is denoted by *TR* in the description of the new algorithm.

Then, the classification of an unknown sample is performed by the learned classification function:

$$f \colon F \to \{1, 2, \dots, C\}, \tag{4}$$

where F is the feature space, that is the set of real matrices with D rows and V columns.

3. New Algorithm

The description of the new classification algorithm is divided into training and predicting phases. We assume that all samples contain an equal number of observations with equal time intervals between observations, thus the continuous time domain is discretized at a fixed number of regularly spaced points of time.

Due to their size, pseudocodes describing both phases of the new algorithm have been included as Supplementary Materials.

3.1. Training

Training is a step consisting of building a classification model using training data set *TR*.

Step 1. Calculation of signal value differences

Observations described by Equation (1) are extended by V new variables. Values of new variables are calculated as difference of actual (at time index i) and previous (at time index $i-1$) variable values, except the first observation of each sample which is extended by zeros. As a result, we get observations containing $2V$ variables:

$$\Theta'_{j,1} = (x_{j,1}^1, x_{j,1}^2, \dots x_{j,1}^V, \underbrace{0, 0, \dots, 0}_{V \text{ times}}) \tag{5}$$

and

$$\Theta'_{j,i} = (x^1_{j,i}, x^2_{j,i}, \ldots, x^V_{j,i}, \\ x^1_{j,i} - x^1_{j,i-1}, x^2_{j,i} - x^2_{j,i-1}, \ldots, x^V_{j,i} - x^V_{j,i-1}) \tag{6}$$

for $i \in \{2, 3, \ldots, D\}$ and $j \in TR$. Finally we get a matrix:

$$\Pi'_j = \begin{bmatrix} x^1_{j,1} & x^2_{j,1} & \cdots & x^S_{j,1} & 0 & 0 & \cdots & 0 \\ x^1_{j,2} & x^2_{j,2} & \cdots & x^S_{j,2} & x^1_{j,2} - x^1_{j,1} & x^2_{j,2} - x^2_{j,1} & \cdots & x^S_{j,2} - x^S_{j,1} \\ \vdots & \vdots & \ddots & \vdots & x^1_{j,3} - x^1_{j,2} & x^2_{j,3} - x^2_{j,2} & \cdots & x^S_{j,3} - x^S_{j,2} \\ & & & & \vdots & \vdots & \ddots & \vdots \\ x^1_{j,D} & x^2_{j,D} & \cdots & x^S_{j,D} & x^1_{j,D} - x^1_{j,D-1} & x^2_{j,D} - x^2_{j,D-1} & \cdots & x^S_{j,D} - x^S_{j,D-1} \end{bmatrix}. \tag{7}$$

Step 2. Merging of data for clustering

Training samples are merged one-by-one into one long single matrix. This matrix has $2V$ columns. The number of rows is equal to the product of the size of the training set and the number of observations in a sample.

Step 3. k-means clustering

The vector quantization using k-means clustering with the given parameter k (denoted k_1 hereafter) is performed over sequences $\Theta'_{j,i}$ for observation index $i \in \{1, 2, \ldots, D\}$ and sample number $j \in TR$. This clustering partitions observations collected in the previous step into k clusters. As a result we have a sequence Ω of symbols representing partitions (clusters),

$$\Omega = (\omega_l)_{l=1}^{k_1}, \tag{8}$$

where ω_l is the symbol representing the lth cluster. Each symbol ω_l for $l \in \{1, 2, \ldots, k_1\}$ has a corresponding set of coordinates Θ'_{ω_l}. The coordinates are from the same space as input data, the $\Theta'_{i,j}$ sequences defined by Equations (5) and (6).

Symbols from Equation (8) are assigned to observations $\Theta'_{j,i}$. For each $i \in \{1, 2, \ldots, D\}$ and $j \in TR$ the symbol $\omega_{\Theta'_{j,i}}$ represents the cluster the observation $\Theta'_{j,i}$ belongs to. Each observation $\Theta'_{j,i}$ is assigned to the nearest cluster calculated using a certain distance function d_{clust}.

The set of the features Θ' and the corresponding classes Ω are a training input to kNN classifier used in prediction phase Step 2.

Step 4. Replacing observations by symbols

Each observation $\Theta'_{j,i}$ in each training sample (indicated by j and i) is replaced by a symbol corresponding to the cluster the observation belongs to, $\omega_{\Theta'_{j,i}}$. As a results, training samples are represented by sequences of symbols:

$$\Pi''_j = (\omega_{\Theta'_{j,i}})_{i=1}^{D} \tag{9}$$

for $j \in TR$.

Step 5. Calculating the frequency table

For each training sample indexed by $j \in TR$ the frequency sequence, known as the Vector Space Model [22], is computed using the corresponding sequence of symbols Π''_j:

$$B_j = (\beta_{j,l})_{l=1}^{k_1} \tag{10}$$

where $\beta_{j,l}$ is the number of times the symbol ω_l appears in the sequence Π''_j.

Step 6. Calculating the class centroid

Within each of class $c \in \{1, 2, \ldots, C\}$, the mean frequency value of each symbol is calculated, as

$$\Gamma_c = \left(\frac{1}{\#S_c} \sum_{j \in S_c} \beta_{j,l} \right)_{l=1}^{k_1}, \tag{11}$$

where S_c is the set of indices of samples from the training set assigned to the class c and $\#S_c$ is the number of elements in set S_c. These class centroids represent the model of the classification algorithm.

3.2. Prediction

A new sample represented by a matrix Π_{new} is assigned to one of the classes using the model built by the described algorithm. Following steps need to be performed during classification:

Step 1. Difference of signal values

Each observation of the new sample is extended as described in step 1 of the training procedure. The resulting matrix is denoted Π'_{new} and is computed analogically to Π'_j in Equation (7).

Step 2. Replacing observations by symbols

Extended observations of a new sample collected in the matrix Π'_{new} are replaced by symbols from the sequence Ω defined by Equation (8) developed during the training phase. This step is performed using the kNN (k-Nearest Neighbor) algorithm. The value of the k parameter is denoted k_2 and the distance function is denoted d_{kNN} hereafter. The training data for this algorithm consist of observations $\Theta'_{j,i}$ for all $j \in TR$ and $i \in \{1, 2, \ldots, D\}$ with corresponding symbols $\omega_{\Theta'_{j,i}}$ used as classes for the purpose of training the kNN model.

The matrix Π'_{new} is transformed to a sequence of symbols Π''_{new} corresponding to sequences calculated in step 4 of the training phase. They are, however, obtained by classifying each row of Π'_{new} using the learned kNN model.

Step 3. Calculate the frequency table

Calculate the symbol frequency table in VSM (Vector Space Model) of the new sample as in step 5 of the training phase:

$$B_{new} = (\beta_{new,l})_{l=1}^{k_1} \tag{12}$$

where $\beta_{new,l}$ is the number of times the symbol ω_l appears in the sequence Π''_{new}.

Step 4. Indicate the class label of a new survey

The distances between the symbol frequency table B_{new} of the new sample and centroids Γ_c of each class $c \in \{1, 2, \ldots, C\}$ from Equation (11) are calculated using a distance function d_{VSM}. The index c of the nearest centroid indicates the class for the new sample.

3.3. Parameter Optimization

The algorithm has a set of parameters that have to be adjusted to optimize its accuracy. For the training phase the number of partitions k_1 and the distance function d_{clust} of the k-means algorithm need to be determined. For the prediction phase, the number of neighbours k_2 and distance function of the kNN algorithm were optimized, as well as the distance function d_{VSM}.

Standard distance functions were considered for d_{clust}, d_{kNN} and d_{VSM}, including the city block distance, Chebyshev distance, correlation distance, cosine distance, Euclidean distance, Hamming distance, Jaccard distance, Mahalanobis distance, Minkowski distance, squared Euclidean distance and the Spearman distance.

Tested and optimal parameters are presented in more detail Section 5.

3.4. Time Complexity

The time complexity of the training depends linearly on the product of: the number of variables in an observation, the number of observations in a sample, the number of samples in each class and the number of classes. Scaling up of training depends only on the number of classes, because we consider small (and constant) training sets, as in motivation of this work and rest of the parameters are constant.

The classification time complexity of a single gesture is similar to training, but the number of samples in each of classes is omitted. Scaling up of classification is also linearly dependent only on the number of classes.

4. Experiments

The new algorithm presented in this article was tested, evaluated and compared to other methods using a database of gesture execution records [23] available as Supplementary Materials. To build this database the DG5 VHand glove was used. The database was used in two different problems: person identification using a known gesture and gesture recognition assuming the person performing it is identified. Two experiments with evaluation based on data set resampling were performed to compare the new method to well-known algorithms listed in Section 1. The different methods were compared quantitatively by determining the correlation between the training set size and the error rate of classification.

4.1. Gestures Data Set

The glove used to acquire the data has 10 sensors: five finger flexion sensors, one for each of the fingers (thumb, index, middle, ring, little), three accelerometers to measure hand movements in each of x-, y-, and z-axis and two gyroscopes to determine hand orientation (roll and pitch). The sensors are numbered from 1 to 10 in the given order. A single database record, called survey, corresponds to one gesture execution performed by a single person. Surveys are represented by matrices structured as in the example in Table 1. Their rows correspond to sensor readings at a particular moment. The first column denotes timestamp while the other ones correspond to readings from ten glove sensors pulled at that time. Glove readings of a sample survey are visualized in Figure 1.

Table 1. Gesture execution example (survey).

	←	Variables				→
	Timestamp	**Sensor 1**	**Sensor 2**	**Sensor 3**	**…**	**Sensor 10**
Observations	0	14.8438	17.1875	14.2725	…	0.0343
	47	14.8438	17.1875	14.2529	…	0.0467
	63	14.8438	17.1875	14.2432	…	0.0513
	…	…	…	…	…	…
	1869	47.7539	67.0801	20.3076	…	−0.0044
	1900	47.6465	66.8164	18.8184	…	0.0010

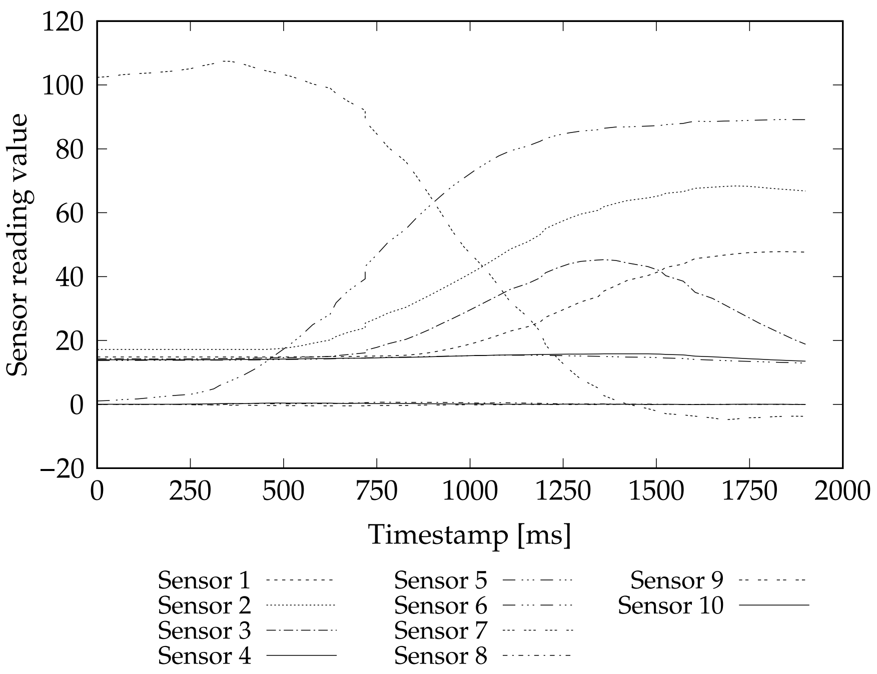

Figure 1. Visualization of exemplary gesture execution signals values.

Each sensor corresponds to a variable, a single reading of all sensors at the same time is an observation and the time series of observations from a particular gesture execution is a sample.

The database consists of surveys of 22 different hand gestures executed 10 times by 10 people, $J = 2200$ records in total. The details of this database are discussed in [23]. A single survey (single gesture execution) contains from 12 to 149 observations and lasts from 360 ms to 4625 ms. Readings are recorded at a sampling rate varying between about 20 Hz to 40 Hz.

4.2. Experiment Design

In both experiments, the comparisons of the algorithms were performed separately for each number $n \in \{1, 2, ..., S-1\}$ of surveys taken from each class to the training set. For a given n, S separate samplings from the data set were performed. The samples for the training set $TR_{n,w}$ were selected using a circular sliding window scheme within each class:

$$TR_{n,w} = \{j_{c,r} \colon c \in \{1, 2, \ldots, C\}, \\ r \in \{w, w+1, \ldots, w+n-1\}\}, \tag{13}$$

where $w \in \{1, 2, \ldots, S\}$ is the resampling number and $j_{c,r}$ is the sample index of the rth sample in class c, $r \in \{1, 2, \ldots, S\}$. In Equation (13) it is also assumed that $j_{c,r+S} = j_{c,r}$ for each class $c \in \{1, 2, \ldots, C\}$ and sample number $r \in \{1, 2, \ldots, S\}$.

The test set consisted of the other $S - n$ samples from each class. The dependency of the error rate on the size n of the training set was measured.

4.3. Data Preprocessing

Surveys acquired directly from glove are of different length because of differences among gestures shapes and irregular speed of their execution. Additionally, the readings from the glove hardware are not performed at regular time intervals. The preprocessing step is performed to resample the surveys to ensure that each survey is represented by a matrix Π_j of the same size given by Equation (2) and that time intervals between consecutive readings are constant. This step is performed using linear interpolation method. As a result, every survey record contains exactly the same number of observations, and thus observations can be consistently numbered by an index $i \in (1, 2, \ldots, D)$.

For other compared methods it is required additionally to transform the matrices Π_j to a single-column vector. This is done by concatenating columns of each matrix Π_j as in the previous work [23].

4.4. Methods Evaluation

The new algorithm was compared to the well-known ones listed in Section 1. Each method has various parameters to be adjusted to configure the given algorithm for the best classification results. The optimal parameters were looked for to minimize the mean error rate as algorithm evaluation criteria. The values of these parameters were determined using grids of parameter values and the exhaustive search. The Winner Takes All (WTA) rule was used to indicate the correct class in this multi-class classification problem and all other classes were indicated as incorrect.

Implementation of the test environment was based on Matlab software, scikit-learn Python Library [38] based on SciPy, NumPy and NeuPy, and LIBSVM.

5. Results

The model presented in Section 3 was tested using the database described in Section 4.1 in two experiments introduced in Section 4. Both experiments are the classification problems where the single class is a set of samples described in Section 2.1. In the first experiment the new algorithm was compared to other in task of the person identification using one given gesture. There were 22 sub-experiments for each gesture separately. Each of the sub-experiments had 10 classes ($C = 10$) corresponding to 10 people who performed a given gesture. In the second experiment the task of gesture recognition using a gesture performed by the identified person was considered. There were 10 sub-experiments for each person separately. In each-sub experiment there were 22 classes ($C = 22$), one for each gesture type. In both experiments the number of samples per class S is equal to 10, and the number of surveys in each of sub experiments is equal to, respectively, 100 and 220.

Proposed algorithm may be a part of a complete system that includes hardware, data acquisition and recording module, classification algorithm, decision module, etc. If a person is doing nothing, the part of the system that processes the signal from hardware should tag this signal as empty and the system should not pass it to the classification algorithm.

5.1. Parameter Selection

The PNN algorithm depends on one parameter: the spread. The method was tested with spread in a range from 0.01 to 1.00. No particular value in this range resulted in the highest accuracy in all cases.

The *k*NN method depends on the number of neighbours and the distance function. The number of neighbours was tested in the range from 1 to 4, with 1 neighbour resulting in the most accurate classification. The city block distance was found to be the best on average in terms of classification accuracy but for some particular gestures or persons different distance functions were better.

For the SVM based classification, the core parameters are SVM type and kernel (with its parameters). In the experiments there were tested SVM types: C-SVC, ν-SVC, one-class SVM, ϵ-SVR and ν-SVR and SVM kernel types: linear $k_{lin}(x, y) = x^T y$, polynomial $k_{poly}(x, y) = (x^T y + \gamma)^d$, Radial Basis Function-based $k_{RBF} = \exp(-\gamma \|x - y\|^2)$ and sigmoid $k_{sigm}(x, y) = \tanh(\gamma x^T y + C)$. The C-SVC variant with the polynomial kernel was the most accurate among variants of the SVM classifier. The degree d of the polynomial kernel was tested in the range from 1 to 5 and γ parameter tested in the range from 0.5 to 1.0.

In the standard multi-layer perceptron neural networks the optimized parameters are the number of neurons and the network optimization algorithm used for training. The number of neurons in the hidden layer was tested in the range from 10 to 100 with the step equal to 10, but the most accurate classification was obtained mostly in the range from 10 to 40 neurons. The hyperbolic tangent activation function was used in the hidden layer and the softmax activation function was used in the output layer. Different optimization algorithms were tested but scaled conjugate gradient and Fletcher-Powell conjugate gradient produced neural networks with the highest accuracy.

The number of trees affected the accuracy of the random forest classifier (denoted TBG) to the greatest degree. It was found that in the studied problems the forest should have at least 50 trees.

The siamese neural network was based on a convolutional neural network and was adapted into one-dimensional data. The architecture consisted of two convolutional layers with ReLU activation function and maximum pooling. Using more than two such layers did not give any relevant improvement. Three parameters were optimized in both convolutional layers: the size of filters in the range from 1 to 10, the number of filters in the range from 2^5 to 2^8, and the filter stride from 1 to the filter size. The best results were achieved when the first layer consisted of 64 filters of size 6 and stride 1 and in the second layer there were 128 filters of size 7 and stride 1 as well. There was also the fully connected layer optimized with the size of features vector in the range from 2^7 to 2^{12}, with 2^{11} resulting in the most accurate classification. As the SNN algorithm needs at least two training samples for each class to learn when samples are similar, the results start also from the number of training samples equal to two.

Finally, the new algorithm (denoted QUA, as it is based mainly on vector *qua*ntization) was tested for parameters listed in Section 3.3 and the most accurate classification was reached for the number of clusters k_1 in the range from 140 to 200 (tested with step 20), the city block distance function d_{clust}, the number of neighbours in the *k*NN algorithm k_2 equal to 1, the *k*NN distance function d_{kNN} equal to the standardized Euclidean distance and the city block distance used in VSM as d_{VSM}.

5.2. Efficiency Results

The mean error rate as algorithm evaluation criteria was used:

$$ERR = 1 - ACC \tag{14}$$

where ACC is the number of correctly classified instances divided by the number of all classified instances.

The comparison of the new algorithm and other classification methods optimized with respect to the mean error rate there is presented as follows. In Table 2 and correspondingly Figure 2 there are results of the person identification experiment, while in Table 3 and correspondingly Figure 3 there are results of the gesture recognition experiment. Both tables and figures present dependence of the average error rate of different classifiers on the number of training samples per class.

The two confusion matrices were calculated to evaluate the behavior of the proposed algorithm for the most interesting case of using only one training sample from each class. The matrices are attached as Supplementary Materials and the results are as follows. In the case of person identification the most distinguishable gestures executions were performed by person number 3, and the least by persons number 4 and 8. The lowest number of misclassified executions of gestures were assigned to person number 10, where the most misclassifications were assigned to person number 2. In the case of gesture recognition the most distinguishable gestures have numbers 11 and 19, and the least distinguishable were 7 and 21. Misclassfied gestures were most commonly recognized as gesture 15 and least often as gesture number 11.

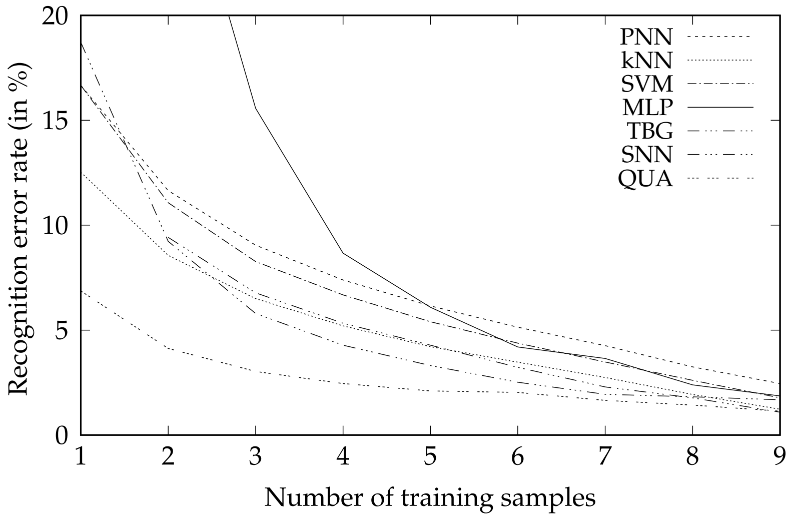

Figure 2. Dependence of the average error rate of different classifiers in the problem of person recognition on the number of training samples per class *n*.

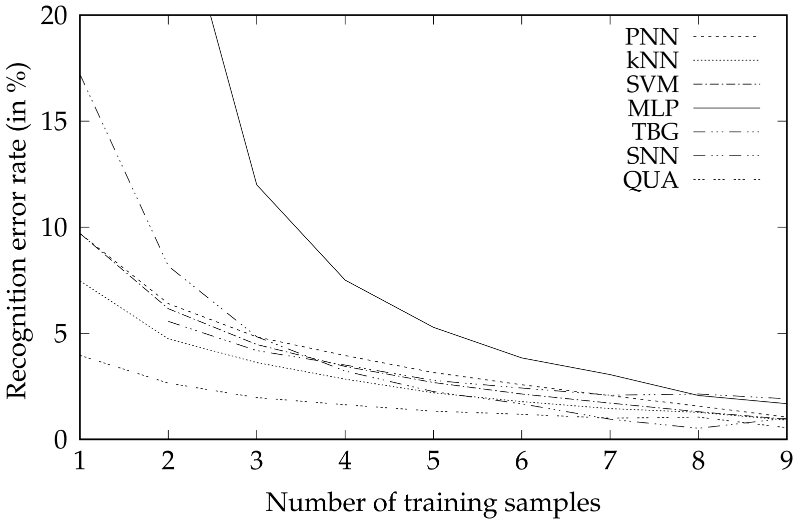

Figure 3. Dependence of the average error rate of different classifiers in the problem of gesture recognition on the number of training samples per class *n*.

Table 2. Person recognition error rate. The first column, labelled *n*, denotes the number of training samples per class. The best result in a given row is shown in bold and indicates the least erroneous classifier.

n	PNN	kNN	SVM	MLP	TBG	SNN	QUA
1	16.67%	12.53%	16.67%	77.24%	18.72%	—	**6.87%**
2	11.65%	8.56%	11.07%	29.95%	9.23%	9.42%	**4.13%**
3	9.05%	6.50%	8.27%	15.57%	5.79%	6.78%	**3.03%**
4	7.39%	5.20%	6.68%	8.67%	4.28%	5.33%	**2.45%**
5	6.15%	4.21%	5.40%	6.09%	3.31%	4.28%	**2.10%**
6	5.14%	3.47%	4.38%	4.20%	2.52%	3.23%	**2.03%**
7	4.26%	2.74%	3.48%	3.65%	1.94%	2.29%	**1.65%**
8	3.25%	1.93%	2.61%	2.39%	1.82%	1.77%	**1.43%**
9	2.45%	1.23%	1.77%	1.86%	1.68%	1.09%	**1.14%**

Table 3. Gesture recognition error rate. The first column, labelled n, denotes the number of training samples per class. The best result in a given row is shown in bold and indicates the least erroneous classifier.

n	PNN	kNN	SVM	MLP	TBG	SNN	QUA
1	9.71%	7.47%	9.71%	79.23%	17.21%	—	**3.96%**
2	6.40%	4.74%	6.16%	27.26%	8.17%	5.56 %	**2.65%**
3	4.83%	3.62%	4.47%	12.00%	4.84%	4.18 %	**1.97%**
4	3.95%	2.84%	3.44%	7.51%	3.22%	3.50 %	**1.63%**
5	3.15%	2.19%	2.68%	5.28%	2.25%	2.78 %	**1.33%**
6	2.57%	1.78%	2.14%	3.84%	1.67%	2.42 %	**1.18%**
7	2.06%	1.45%	1.71%	3.05%	**0.95%**	2.08 %	1.00%
8	1.57%	1.27%	1.30%	2.07%	**0.52%**	2.14 %	1.05%
9	1.05%	0.91%	0.95%	1.68%	1.00%	1.91 %	**0.55%**

5.3. Performance Results

Training and testing times of the new algorithm and other existing algorithms were compared. To benchmark algorithms a computer running Linux Mint 20 Ulyana with Intel(R) Core(TM) i7-9700KF CPU @ 3.60 GHz CPU and 32 GB of Crucial DDR4 RAM at 2667 MT/s was used. The SNN implementation was tested using Gigabyte graphic card with GeForce RTX 2070 SUPER graphic processor unit with driver version 455.32.00 and CUDA version 11.1. The results are presented as follows. In Table 4 there is an efficiency comparison of algorithms in the person identification experiment, while in Table 5 there is a comparison in the gesture recognition experiment.

Table 4. Algorithm performance for person identification, time in milliseconds. The training time was measured for classes size of 1. The classification time was measured for one sample.

	PNN	*k*NN	SVM	MLP	TBG	SNN	QUA
training	12.75	3.17	0.42	92.20	57.55	59.16 s	11.87
classification	0.0514	0.4819	<0.1 µs	0.0574	0.4290	151.84	10.3926

Table 5. Algorithm performance for gesture recognition, time in milliseconds. The training time was measured for classes size of 1. The classification time was measured for one sample.

	PNN	*k*NN	SVM	MLP	TBG	SNN	QUA
training	15.71	4.00	1.92	95.77	61.76	60.74 s	14.93
classification	0.0376	0.5081	<0.1 µs	0.0289	0.2638	104.32	11.50

6. Discussion

Both experiments that were performed for the tasks of person identification and gesture recognition indicated that the new classifier provides higher accuracy than all other well-known algorithms. The difference is the most significant when the number of training samples is small, up to about 6 to 9 samples per class.

In addition to methods listed in Section 1 some other ones were also tested. For methods like Gaussian process classifier, an adaptive neuro-fuzzy inference system, decision trees, radial basis function-based neural networks and generalized regression neural networks either the computation time or achieved results were not sufficient to reach fully comparable results and thus they are not discussed in more detail.

It can be observed that the new algorithm results in the most accurate classification when the 20-dimensional extended observation space is quantized into a relatively large number of clusters. Depending on the problem, there are either 2100 or 4620 points grouped into from 140 to 200 clusters. This step reduces a continuous-variable classification problem into a discrete one which is one of the sources of generalization power of the proposed method. It is also notable that the algorithm has only five discrete free parameters (k_1, k_2, d_{clust}, d_{kNN} and d_{VSM}), which makes overfitting less likely then in most classification methods.

The data is further reduced in steps 5 of 6 of the training phase, where the frequency table approach is used. It reduces the need to perform curve registration by only considering how long the gesture execution stayed in a particular discrete state. The temporal component of the data is however still present in the form of the difference components calculated in step 1 of training and prediction procedures.

The experiments have shown that the time taken by the training phase of the new algorithm was average compared to benchmark results of other algorithms. The classification time was not the longest among all algorithms however, but substantially longer than the fastest one. The main bottleneck of the new classification algorithm is located in Step 2. where replacing observations are replaced by symbol. For this step the *k*NN algorithm is used. The shortest recorded survey (gesture execution) lasts 360 ms and the mean classification time of about 11 ms to 13 ms is less than 4% of it. In real applications of person identification or gesture recognition this classification time should not be noticeable. However, future work on the new algorithm should be concentrated on improving the performance of Step 2.

7. Conclusions

In this article the new time series classification algorithm was presented. The algorithm is based on vector quantization of recorded observations, transforming them into sequences of discrete symbols comparing them using a vector space model. The algorithm was tested, evaluated and compared to state-of-the-art methods using hand gesture recordings in tasks of person identification and gesture recognition. It was shown that the new algorithm is more accurate than other methods, especially on small training sets.

Supplementary Materials: The following are available online at http://www.mdpi.com/1424-8220/20/24/7279/s1: pseudocodes of the proposed algorithm, gesture execution records, confusion matrices.

Funding: This scientific work is supported by the AGH University of Science and Technology in the year 2020 as research project No. 16.16.120.773.

Acknowledgments: The author would like to thank Mateusz Baran for constructive criticism of the manuscript.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Murphy, K.P. *Machine Learning: A Probabilistic Perspective*, 1st ed.; Adaptive Computation and Machine Learning; The MIT Press: Cambridge, MA, USA, 2012.
2. Rutkowski, L. *Computational Intelligence: Methods and Techniques*, 1st ed.; Springer: New York, NY, USA, 2008.
3. James, G.; Witten, D.; Hastie, T.; Tibshirani, R. *An Introduction to Statistical Learning: With Applications in R*; Springer: New York, NY, USA, 2014.
4. Altman, N.S. An introduction to kernel and nearest-neighbor nonparametric regression. *Am. Stat.* **1992**, *46*, 175–185.
5. Specht, D.F. Probabilistic neural networks. *Neural Netw.* **1990**, *3*, 109–118. [CrossRef]
6. Hinton, G.E. Connectionist Learning Procedures. *Artif. Intell.* **1989**, *40*, 185–234. [CrossRef]
7. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
8. Rasmussen, C.E.; Williams, C.K.I. *Gaussian Processes for Machine Learning*; Adaptive Computation and Machine Learning; The MIT Press: Cambridge, MA, USA, 2005.
9. Sugeno, M. *Industrial Applications of Fuzzy Control*; Elsevier Science Pub. Co.: Amsterdam, The Netherlands, 1985.
10. Broomhead, D.S.; Lowe, D. Radial Basis Functions, Multi-Variable Functional Interpolation and Adaptive Networks. *Complex Syst.* **1988**, *2*, 321–355.

11. Bromley, J.; Guyon, I.; LeCun, Y.; Säckinger, E.; Shah, R. Signature Verification Using a "Siamese" Time Delay Neural Network. In *NIPS'93: Proceedings of the 6th International Conference on Neural Information Processing Systems*; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1993; pp. 737–744.
12. El-Abed, M.; Giot, R.; Hemery, B.; Rosenberger, C. A study of users' acceptance and satisfaction of biometric systems. In Proceedings of the 44th Annual 2010 IEEE International Carnahan Conference on Security Technology, San Jose, CA, USA, 5–8 October 2010; pp. 170–178. [CrossRef]
13. Oh, U.; Findlater, L. The Challenges and Potential of End-User Gesture Customization. In *CHI'13: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*; Association for Computing Machinery: New York, NY, USA, 2013; pp. 1129–1138. [CrossRef]
14. Han, Z.; Ban, X.; Wang, X.; Wu, D. Robust and customized methods for real-time hand gesture recognition under object-occlusion. *arXiv* **2018**, arXiv:1809.05911.
15. Sriram Krishna, N.S. Gestop: Customizable Gesture Control of Computer Systems. *arXiv* **2020**, arXiv:2010.13197.
16. Morgado, L. Cultural Awareness and Personal Customization of Gestural Commands Using a Shamanic Interface. *Procedia Comput. Sci.* **2014**, *27*, 449–459. [CrossRef]
17. Ascari, R.E.O.S.; Silva, L.; Pereira, R. Personalized Interactive Gesture Recognition Assistive Technology. In Proceedings of the 18th Brazilian Symposium on Human Factors in Computing Systems, IHC '19, Vitoria, Brazil, 22–25 October 2019; Association for Computing Machinery: New York, NY, USA, 2019. [CrossRef]
18. Muralidhar, P.; Saha, A.; Sateesh, P. Customizable Dynamic Hand Gesture recognition System for Motor Impaired people using Siamese neural network. In Proceedings of the 2019 International Conference of Artificial Intelligence and Information Technology (ICAIIT), Yogyakarta, Indonesia, 13–15 March 2019; pp. 354–358. [CrossRef]
19. Mezari, A.; Maglogiannis, I. An Easily Customized Gesture Recognizer for Assisted Living Using Commodity Mobile Devices. *J. Healthc. Eng.* **2018**, *2018*, 3180652. [CrossRef]
20. Gyöngyössy, N.M.; Domonkos, M.; Botzheim, J.; Korondi, P. Supervised Learning with Small Training Set for Gesture Recognition by Spiking Neural Networks. In Proceedings of the 2019 IEEE Symposium Series on Computational Intelligence (SSCI), Xiamen, China, 6–9 December 2019; pp. 2201–2206. [CrossRef]
21. Kurakin, A.; Zhang, Z.; Liu, Z. A real time system for dynamic hand gesture recognition with a depth sensor. In Proceedings of the 2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO), Bucharest, Romania, 27–31 August 2012; pp. 1975–1979.
22. Cha, S.H. Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions. *Int. J. Math. Model. Methods Appl. Sci.* **2007**, *1*, 300–307.
23. Pławiak, P.; Sośnicki, T.; Niedźwiecki, M.; Tabor, Z.; Rzecki, K. Hand Body Language Gesture Recognition Based on Signals From Specialized Glove and Machine Learning Algorithms. *IEEE Trans. Ind. Inform.* **2016**, *12*, 1104–1113. [CrossRef]
24. Wahid, M.F.; Tafreshi, R.; Al-Sowaidi, M.; Langari, R. An efficient approach to recognize hand gestures using machine-learning algorithms. In Proceedings of the 2018 IEEE 4th Middle East Conference on Biomedical Engineering (MECBME), Tunis, Tunisia, 28–30 March 2018; pp. 171–176. [CrossRef]
25. Baran, M. Closest paths in graph drawings under an elastic metric. *Int. J. Appl. Math. Comput. Sci.* **2018**, *28*, 387–397. [CrossRef]
26. Baran, M.; Siwik, L.; Rzecki, K. Application of Elastic Principal Component Analysis to Person Recognition Based on Screen Gestures. In *Artificial Intelligence and Soft Computing*; Rutkowski, L., Scherer, R., Korytkowski, M., Pedrycz, W., Tadeusiewicz, R., Zurada, J.M., Eds.; Springer: Cham, Switzerland, 2019; pp. 553–560.
27. Rzecki, K.; Siwik, L.; Baran, M. The Elastic k-Nearest Neighbours Classifier for Touch Screen Gestures. In *Artificial Intelligence and Soft Computing*; Rutkowski, L., Scherer, R., Korytkowski, M., Pedrycz, W., Tadeusiewicz, R., Zurada, J.M., Eds.; Springer: Cham, Switzerland, 2019; pp. 608–615.
28. Przybyło, J.; Kańtoch, E.; Augustyniak, P. Eyetracking-based assessment of affect-related decay of human performance in visual tasks. *Future Gener. Comput. Syst.* **2019**, *92*, 504–515. [CrossRef]
29. Schramm, R.; Jung, C.R.; Miranda, E.R. Dynamic Time Warping for Music Conducting Gestures Evaluation. *IEEE Trans. Multimed.* **2015**, *17*, 243–255. [CrossRef]
30. Zabatani, A.; Surazhsky, V.; Sperling, E.; Ben Moshe, S.; Menashe, O.; Silver, D.H.; Karni, T.; Bronstein, A.M.; Bronstein, M.M.; Kimmel, R. Intel® RealSense™ SR300 Coded light depth Camera. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**, *42*, 2333–2345. [CrossRef]

31. Rzecki, K.; Pławiak, P.; Niedźwiecki, M.; Sośnicki, T.; Leśkow, J.; Ciesielski, M. Person recognition based on touch screen gestures using computational intelligence methods. *Inf. Sci.* **2017**, *415–416*, 70–84. [CrossRef]
32. Wu, Y.; Jiang, D.; Liu, X.; Bayford, R.; Demosthenous, A. A Human-Machine Interface Using Electrical Impedance Tomography for Hand Prosthesis Control. *IEEE Trans. Biomed. Circuits Syst.* **2018**, *12*, 1322–1333. [CrossRef]
33. Zhang, Y.; Harrison, C. Tomo: Wearable, Low-Cost Electrical Impedance Tomography for Hand Gesture Recognition. In Proceedings of the 28th Annual ACM Symposium on User Interface Software and Technology, UIST '15, Charlotte, NC, USA, 8–11 November 2015; ACM: New York, NY, USA, 2015; pp. 167–173. [CrossRef]
34. Kim, Y.; Toomajian, B. Hand Gesture Recognition Using Micro-Doppler Signatures With Convolutional Neural Network. *IEEE Access* **2016**, *4*, 7125–7130. [CrossRef]
35. Ren, Z.; Yuan, J.; Meng, J.; Zhang, Z. Robust Part-Based Hand Gesture Recognition Using Kinect Sensor. *IEEE Trans. Multimed.* **2013**, *15*, 1110–1120. [CrossRef]
36. Lu, W.; Tong, Z.; Chu, J. Dynamic Hand Gesture Recognition with Leap Motion Controller. *IEEE Signal Process. Lett.* **2016**, *23*, 1188–1192. [CrossRef]
37. Marin, G.; Dominio, F.; Zanuttigh, P. Hand gesture recognition with jointly calibrated Leap Motion and depth sensor. *Multimed. Tools Appl.* **2016**, *75*, 14991–15015. [CrossRef]
38. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.