# Skeleton-Based Recognition of Traditional Greek Dance Steps using Machine Learning Algorithms

Konstantinos Tragiannis*, Thanasis Balafoutis†, Vasiliki Balaska‡ and Antonios Gasteratos§
Department of Production and Management Engineering, Democritus University of Thrace
Xanthi, Greece
Email: *ktragian@pme.duth.gr, †abalafou@pme.duth.gr, ‡vbalaska@pme.duth.gr, §agaster@pme.duth.gr

***Abstract*—Classification of dance types and moves is a challenging application in the field of action recognition. Owing to its vast applicability and robustness regardless of the image background, skeleton-based recognition is increasingly attracting the attention of scholars in the robotics community. This paper presents a novel dataset of videos of five individuals performing five different types of traditional Greek dances, which can be utilized for that purpose. In addition, we create a benchmark for future work by performing a comparative study of different classifiers' performance on dance steps from this dataset. For that purpose, the dataset has been divided into steps, with each step consisting of a time series of skeleton data extracted from the videos. We use an assortment of classifiers based on different artificial neural network (NN) types, including convolutional and recurrent NNs as well as traditional ones such as Gaussian naive Bayes, decision trees, support vector machines, and $k$-nearest neighbors. For each classifier, we have obtained the classification accuracy first on steps from the same dance type and then on the entire dataset. We assess the classifiers' performance both on randomly split test sets and when performing cross-validation. Out of the classifiers used, the best performance is achieved by the CNN-based classifier.**

***Index Terms*—Skeleton-based action recognition, Neural Networks, Intangible Cultural Heritage, Dance Classification**

## I. Introduction

Performance arts are an important part of Intangible Cultural Heritage (ICH). In many countries, the local traditional music and dances are a focal point of national pride and identity. Therefore, the documentation and analysis of folkloric dances are crucial to ensure that the knowledge of how to perform them can be preserved through time. Technological advancement and the increased availability of devices and software, such as high-resolution video cameras as well as the ability to store video content on the cloud, have made it easier to document, store and disseminate content, therefore making the preservation of ICH more feasible [1]. Another advancement that has aided in preserving and analyzing ICH material is the advent of stereo vision algorithms, cameras, and sensors [2]. These have been utilized in various tasks, including action recognition and pose estimation [3]. Skeleton-based action recognition, in particular, utilizes skeleton data in the form of the coordinates of certain body joints, which have been extracted from a stereoscopic video to track the movement of a human body. The task of skeleton-based action recognition can be applied to the classification of traditional Greek dance steps. This is possible because each type of traditional Greek dance typically consists of a fixed choreography of repeated steps following the same pattern [1], [4].

In this paper, a novel dataset is introduced comprising videos of five individuals performing five different traditional Greek dances. The process of dividing each video into steps constituting the particular dance's choreography has been carried out. In that way, each frame of the videos in the dataset is assigned to one of 22 possible classes, each of the classes corresponding to specific movements of the limbs. 3-dimensional skeleton data were provided by a ZED 2 stereo camera, which was used to record the videos. Our contribution lies in creating a dataset of performances of traditional dances that had not been recorded before. In addition, we create a benchmark for future research by comparing the performance of several types of classification algorithms applied to the dataset.

The rest of this paper is structured as follows: in Section II, past contributions and advancements on the topics of skeleton-based action recognition as well as on the analysis and identification of Traditional Greek Dances are reviewed. In Section III, the methodology used to film the dataset and to implement each classification algorithm is presented. Section IV presents the results of our experiments, which are discussed in Section V. Finally, the conclusions of our work are presented in Section VI. In addition, the possibilities for future work stemming from the results of this paper are discussed in that section.

## II. Related Work

This section contextualizes our research by reviewing the relevant literature, identifying gaps in the field, and situating our study within a broader scholarly conversation. By providing a foundation for our argument, this section helps readers understand the current state of the field and the rationale for our work. Within the field of computer vision, action recognition is an area of research that presents many challenges in the present day. In order to successfully recognize the actions of a human, certain other challenging secondary tasks must be implemented, such as human detection and pose estimation. Several data types can be utilized for this purpose, including RGB, depth-based, and skeleton-based data [5].

Owing to the advancement of depth cameras and sensors, such as the Kinect sensor, Skeleton-based data have been increasingly used recently for action recognition [6], as they

can describe the positions of joints or larger body parts. In previous work, several Artificial Neural Network (ANN) architectures such as Recurrent Neural Networks (RNNs) [7], as well as Convolutional Neural Networks (CNNs) [8] [9] have been used to process this data type. In [10], the aforementioned data were utilized to recognize dance gestures. The classifier achieved a significantly high average accuracy.

More specifically, analysis and classification of traditional Greek dance types and moves or steps have been examined in past research endeavors, often focusing on identifying key postures [11]. In [12], the results of multiple depth cameras are fused to create more accurate skeleton data. Later, these are divided into posture classes using $k$-means clustering, and then a classifier is used to place posture sequences into classes corresponding to dance figures. In [13], multiple depth sensors are again used to partition the dance sequence into periods and patterns. More recently, in [3], classification algorithms are used on skeleton data to identify dance poses and classify the type of dance performed based on the poses found in certain important frames. In [14], a deep CNN architecture was used for dance pose identification on RGB data.

The work which shares the most similarities with ours is probably the one presented in [4], where the performances of several algorithms are compared on the task of classifying different postures, which are part of traditional Greek dance choreography. Each frame, represented by skeleton data, is assigned to a posture class, and the sequence of postures is used to describe the dance steps. The classifiers used are $k$-nearest neighbors, naive Bayes, discriminant analysis, decision trees, support vector machines, and ensemble methods. There are two main differences from our work: 1) we attempt to classify the different dance steps using information from the whole sequence of frames corresponding to a step rather than classifying each frame individually, and 2) our work expands on these traditional classifiers with various ANN-based ones.

## III. Methodology

### A. Pre-processing

To record the videos for our dataset, the ZED 2 stereo camera [15] was used and the recording process was as follows. Every video begins with the dancer at one end of the camera frame and ends when they reach the other end. In each video, we captured only one dancer performing, and the same audio file was used for each type of dance. In many traditional Greek dances, it is typical for the series of steps to be performed in a circle. To determine the impact of the angle at which the dancer faces the camera on the success of classification, we recorded two subsets of videos. One with the dancers performing the steps in a straight line and always facing the camera and one with them dancing in a semicircle. The videos in the second subset start and end with the dancers' bodies roughly vertical to the camera and facing toward the camera in the middle of the video.

When recording, the camera produces two videos (left and right), one for each of its two image sensors, each with a resolution of up to $2208 \times 1242$ pixels and a frame rate of up to 100 fps. With the aid of an AI model, depth data are generated by utilizing the imagery from these two videos. For our dataset, the left video, as well as the depth data, were kept. The settings used were a frame rate of 60 frames per second and a resolution of $1280 \times 720$.

To perform classification on the steps, we annotated the videos dividing them into their constituent steps. Since all videos from each dance genre follow the same rhythm, it was possible to use the timing of the beat in the audio track to annotate all of the videos of the same dance automatically. The audio tracks had to be manually divided into beats because we noticed the rhythm of the audio files not being completely steady throughout their length.

The camera provided us with a software development kit (SDK) that performs human recognition in each frame of the video and creates 3-dimensional $(x_0, x_1, x_2)$ skeleton coordinates, which correspond to the position of the joints relative to the camera. Out of the available formats for the skeleton joint coordinates, we chose the one with 34 points. The layout of the skeleton joints is presented in Figure 1 using a sample from the dataset. Since our goal was to classify each step based on the movements performed by the dancer, we performed a series of transformations on the coordinates to keep only the relevant information :

- First, we translated the coordinates. The new center of coordinates was moved to the middle between the hips at the beginning of each step. Let $\boldsymbol{x_{(j,t)}} = [x_{(j,t),0}, x_{(j,t),1}, x_{(j,t),2}]$ be the coordinate vector of the joint with label $j$ at time $t$ and $\boldsymbol{x_c}$ be the vector of coordinates of the new center. Considering that the left and right hip are represented by labels 18 and 22, respectively:
$$\boldsymbol{x_c} = \frac{\boldsymbol{x_{(18,0)}} + \boldsymbol{x_{(22,0)}}}{2}. \tag{1}$$
The new vectors of coordinates $\boldsymbol{x'_{(j,t)}}$ are calculated as follows:
$$\boldsymbol{x'_{(j,t)}} = \boldsymbol{x_{(j,t)}} - \boldsymbol{x_c}. \tag{2}$$
- To eliminate the influence of the angle at which the dancer faces the camera, we rotated the coordinates around the $y$-axis making the body always face forward. The angle $\phi$ of counter-clockwise rotation is calculated as follows:
$$\phi = tan^{-1}(\frac{x_{(18,0),2}}{x_{(18,0),0}}). \tag{3}$$
After that, the rotation matrix $R_y(\phi)$ is calculated and the rotated coordinates are the product of this matrix and the old coordinates:
$$\boldsymbol{x''_{(j,t)}} = R_y(\phi)\boldsymbol{x'_{(j,t)}} \tag{4}$$
- Finally, we scaled the coordinates, making the distance between the two hips always identical so that the dancer's identity does not make a difference. Assuming the actual distance between the hips is $l$ and the desired distance is $L$:
$$\boldsymbol{x'''_{(j,t)}} = \boldsymbol{x''_{(j,t)}}\frac{L}{l}, \tag{5}$$

where $l$ is calculated as the Euclidean distance between $\boldsymbol{x}_{(18,0)}$ and $\boldsymbol{x}_{(22,0)}$.

### B. Composition of the Dataset

The dataset consists of videos of five individuals performing five types of Greek traditional dance. The dance types that are included are named *Mazomenos, Gikna, Dilbera, and Tapeinos*, as well as a dance originating in the Epirus region of Greece called *Sta Dyo*. The choreography of each type of dance consists of a cycle that is repeated. We divided each cycle into steps, each unique step corresponding to a class. A few of these steps are repeated twice within a cycle or shared between several dances.

The dataset contains 3494 dance steps belonging to 30 different step classes (6 for each dance). The classes were labeled with numbers from 0 to 29. Each step sample contains the 3-dimensional coordinates for 34 joints of the human body for each frame from the beginning to the end of the step. The composition of the dataset is presented in Table II. To ensure that the speed of the dancing does not have an impact on the classification as well as that each sample contains the same number of parameters, we performed downsampling, keeping 20 randomly selected frames for each step, which was the lowest amount of frames in any of the original steps. The selected frames are placed in chronological order. The number of parameters in each sample is derived as follows:

$$parameters = \#Coordinates \cdot \#Joints \cdot \#Frames \quad (6)$$

Overall, this results in 2040 (3 coordinates, 34 joints, and 20 frame samples) parameters. For the rest of the equations, we consider that the coordinate vectors $symbolx'''_{(j,t)}$ are concatenated into a vector $\boldsymbol{v} = [v_1, v_2, ..., v_{2040}]$.In Table I, we present a description of the series of steps performed in each cycle for the dance *Mazomenos*:

TABLE I
DESCRIPTION OF THE STEPS OF MAZOMENOS

| Throughout this dance the arms are raised perpendicular to the body. | |
|---|---|
| ***Step number*** | ***Step Description*** |
| 0 | This step begins with the right foot raised. The dancer moves their foot to the right ending with their legs open. |
| 1 | The dancer moves their left foot in front of their right foot ending with their feet crossed. |
| 2 | The dancer moves their right foot to the right ending with their legs open. |
| 3 | The dancer raises their left foot. |
| 4 | The dancer lowers their left foot. |
| 5 | The dancer raises their right foot. |

### C. Experimental Setup and Classification Algorithms

For this work, nine classification algorithms are implemented in the Python programming language. Five of these

TABLE II
COMPOSITION OF THE DATASET

| ***Dance Type*** | ***Step Labels*** | ***Number of Dance Steps*** |
|---|---|---|
| Mazomenos | 0-5 | 683 |
| Gikna | 6-11 | 876 |
| Dilbera | 12-17 | 629 |
| Tapeinos | 18-23 | 798 |
| Sta Dyo | 24-29 | 508 |

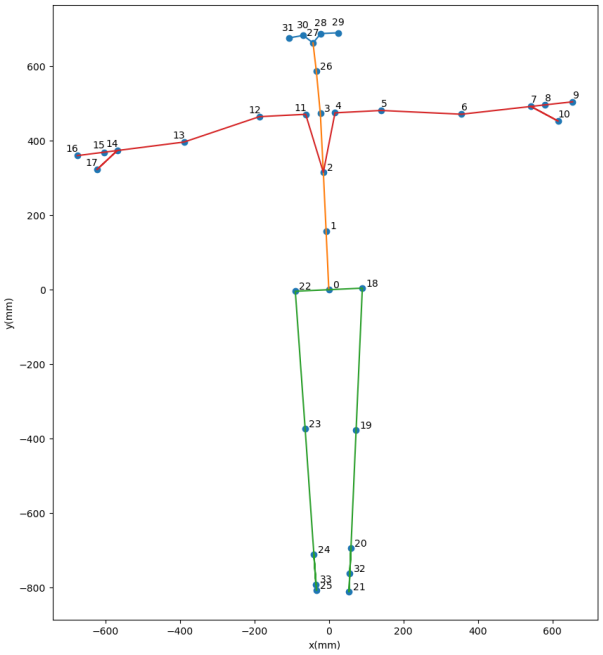


Fig. 1. Labels and coordinates (x,y) in mm of the skeleton joints at the beginning of step 0 of *Mazomenos*. Note the slightly raised right foot of the dancer.

algorithms are types of Neural Networks, while the other four are traditional classification algorithms. We used the Pytorch library for the neural networks and the Sklearn library for the rest of the algorithms. Wherever a split between the training and test data was performed randomly, we used 20% of the original dataset as the test set. For the neural networks, we used the Adam optimizer with a learning rate of 0.001 and a batch size of 32. We ran the machine learning algorithm for 1000 epochs each time with the learning process set to stop early if the accuracy had not improved in the last 100 epochs. The algorithms we used are the following:

***Gaussian Naive Bayes***: The Naive Bayes algorithm is a classification algorithm that calculates the conditional probability of a sample belonging to a class $C$ given its coordinates $\boldsymbol{v} = [v_1, v_2, ..., v_{2040}]$. This probability $P(C|\boldsymbol{v})$ is calculated using Bayes' theorem [16].

The probability of each class $P(C)$ is calculated to be the number of samples belonging to the class $|C|$ divided by the total size of the dataset $|D|$. The coordinates $\boldsymbol{v}$ are considered

conditionally independent, therefore:

$$P(\boldsymbol{v}|C) = \prod_{i=1}^{2040} P(v_i|C). \tag{7}$$

In the Gaussian version of the algorithm used here, each coordinate's value is assumed to follow a Gaussian distribution.

***Decision tree***: The decision tree classification algorithm classifies samples using a series of rules based on data features and represented as nodes. According to [16], the initial dataset is divided into smaller subsets by the nodes, with the ultimate goal of each subset representing a single class.

***k-Nearest Neighbors***: For each step with coordinates $\boldsymbol{v}$, the $k$-Nearest neighbor algorithm finds the subset $N$ of $k$ steps with the smallest Euclidean distance to $\boldsymbol{v}$ and places $\boldsymbol{v}$ in the class with the most instances inside $N$ [17].

In our work, we tried a range of values for $k$, concluding that the classification performance improves for smaller values, with $k = 1$ giving the best results. We did not perform a random split into training and test data for this algorithm. Instead, for each step in the original dataset, we found its $k$-nearest neighbors from the entire dataset.

***Support Vector Machine (SVM)***: SVM algorithms use a margin that separates space into two areas, each of them corresponding to a class [18]. New samples are placed in one of the two classes based on which boundary of the margin they are closer to. When using a linear kernel as we do here, the boundaries of this margin are two hyperplanes described by the following equations:

$$\begin{aligned} \boldsymbol{w^T v} - b &= 1 \\ \boldsymbol{w^T v} - b &= -1. \end{aligned} \tag{8}$$

Ideally, the aim is for all instances of each of the two classes to be on opposite sides of the margin and the two boundaries of the margin as far away from each other as possible. Since finding such boundaries is often impossible, we apply an optimization process that penalizes them for every sample on the wrong side of the boundary.

SVMs can be expanded to be usable in cases with more than two classes. In our case, a one-versus-one scheme is used, where a separate SVM classifier is used for every different pair of classes. Every sample is assigned to one of the two classes, and that class receives a vote. In the end, each sample is placed in the class which receives the most votes.

***Feedforward Neural Network(FNN)***: This is the most basic type of artificial neural network, whose architecture does not include any cyclical connections [19]. The first component of our network is a linear layer, whose output is the product of a table $W$ of learnable weights and the input coordinates $v$. This is followed by nonlinearity in the form of a Rectified Linear Unit (ReLU) layer and then another linear layer. The hidden size of the output vector of the first and second layers and the input of the second and third layers was set to 1024.

***Recurrent Neural Network (RNN)***: An RNN is a neural network that includes at least one node whose output is connected to its input, forming a cycle. This makes RNNs suitable for treating sequential data since it is possible for information about previous parts of the sequence to be passed down using that cyclical connection. Usage of RNNs usually involves parameter sharing, where the same neurons are used for all points in the sequence, greatly reducing the computational cost. In our architecture, the network's hidden state results from a unidirectional Elman RNN [20] with a nonlinearity implemented with a hyperbolic tangent function. The hidden state is then fed into a linear layer. The following equations describe the network. For all RNNs let $\boldsymbol{i_t} = [x'''_{1,t}, y'''_{1,t}, z'''_{1,t}, x'''_{2,t}, ...z'''_{34,t}]$ be the input vector at time t, $\boldsymbol{h_t}$ be the hidden state and $\boldsymbol{o_t}$ the output:

$$\begin{aligned} \boldsymbol{h_t} &= tanh(W_{ih}\boldsymbol{i_t} + W_{hh}\boldsymbol{h_{t-1}} + \boldsymbol{b_h}) \\ \boldsymbol{o_t} &= tanh(W_y\boldsymbol{h_t} + \boldsymbol{b_y}). \end{aligned} \tag{9}$$

The input for this network and the other two recurrent networks (GRU and LSTM) is two-dimensional. One dimension is the sequence length equal to 20, and the other equals 102 $(\#OfCoordinates \cdot \#OfJoints)$. We set the size of the hidden state to 128. Both in this Neural Network and the GRU and LSTM networks, we use the network output at the end of the sequence as the output of the whole RNN.

***Long Short-term Memory (LSTM) Neural Network***: Conventional RNN architectures often face the problem of long-term dependencies. This means that information passed through the network from previous points in the sequence tends to either increase disproportionately or be reduced to the point of being irrelevant after many time steps. The LSTM Network is an RNN architecture that was introduced with the aim of solving this problem [21]. Therefore, it is particularly suitable for dealing with long sequences of data. To achieve that, the LSTM uses three gates (input, output, and forget) which influence what information from previous time steps is kept or discarded. Each gate $g$ uses the input and hidden state vectors as its inputs:

$$g_t = \sigma(W_{ig}\boldsymbol{i_t} + W_{hg}\boldsymbol{h_t} + \boldsymbol{b_g}), \tag{10}$$

where $\sigma$ represents the sigmoid function. The weight tables $W_{ig}$ and $W_{hg}$ are different for each gate. Our network uses a unidirectional LSTM network with one layer. Both the size of the hidden state and the output of the LSTM were set to 128. That output is then fed into a linear layer.

***Gated Recurrent Unit (GRU) Neural Network***: Similarly to the LSTM, the GRU is an RNN architecture that utilizes gates to deal with long-term dependencies [22]. Their main difference lies in reducing the number of gates from three to two, as the GRU lacks an output gate. For our Network, we used a single-layer unidirectional GRU and a hidden state with a size of 128, followed by a linear layer.

***Convolutional Neural Network (CNN)***: Finally, we implemented an architecture using a CNN as its main component. CNNs have been shown to perform well on various computer vision tasks and therefore have been widely utilized [14]. Our architecture comprises a layer that performs one-dimensional

convolution on the coordinates in the input, followed by a Rectified linear unit and a linear layer. We performed the convolution on three vectors $\boldsymbol{v_0}, \boldsymbol{v_1}, \boldsymbol{v_2}$ each of them with 680 $(\#ofJoints \cdot \#ofFrames)$ values. The number of output channels was set to 32. The output of channel n is calculated using the following equation:

$$\boldsymbol{out}(n) = \boldsymbol{bias}(n) + \sum_{k=0}^{2} \boldsymbol{weight}(n,k) * \boldsymbol{v_k}, \qquad (11)$$

where $\boldsymbol{weight}(n,k)$ are the vectors of weights and $*$ is the cross-correlation operator. We used a kernel with size=5 and stride=1 and no padding.

## IV. Experimental Results

Firstly, we tried the classification algorithms on five subsets, each consisting of the steps of one dance type. Afterward, we performed cross-validation by averaging out the results from several test sets. Each test set contained steps from a different dancer who had been excluded from the training set. The accuracy metrics of the algorithms when performing cross-validation are presented in Table III. From these results, it is apparent that most algorithms struggle to differentiate the steps of *Sta Dyo*. This can be explained by the fact that two *Sta Dyo* step classes (24 and 26) have similar movements.

TABLE III
AVERAGE ACCURACY OF THE CLASSIFICATION ALGORITHMS FOR EACH DANCE TYPE.

| ***Classifier*** | ***Mazomenos*** | ***Gikna*** | ***Dilbera*** | ***Tapeinos*** | ***Sta Dyo*** |
|---|---|---|---|---|---|
| GNB | 90.24% | 89.22% | 92.74% | 91.84% | 79.07% |
| DTree | 88.54% | 88.30% | 92.74% | 93.51% | 65.58% |
| RNN | 96.48% | 88.18% | 93.89% | 95.96% | 85.81% |
| FNN | 98.46% | 97.16% | 98.64% | 99.54% | 93.62% |
| LSTM | 98.17% | 91.83% | 97.96% | 98.72% | 90.88% |
| GRU | 98.58% | 92.39% | 97.87% | 98.78% | 90.72% |
| $k$-Near | 92.32% | 88.85% | 91.52% | 94.20 % | 82.05% |
| SVM | 97.53% | 97.68% | 87.92% | 95.89% | 88.38% |
| CNN | 97.86% | 96.91% | 98.15% | 99.23% | 93.93% |

Subsequently, we obtained accuracy metrics for the algorithms when they were applied to a randomly split dataset and performed cross-validation on the CNN algorithm, which showed the best accuracy. To assess whether the similarity between the steps belonging to different classes affects the performance, we compared the recall metrics of different classes, presented in Figure 2. Out of 11 steps with recall lower than 0.6, only classes 6, 10, and 11 do not have an equivalent step with identical movement. Step 27 is also roughly identical to steps 14 and 20. To improve the performance of the classifiers, we merged the step classes with identical movements. This resulted in the total number of classes being reduced to 22.

The accuracy of each Neural Network based classifier is not deterministic due to the stochastic nature of the optimizer and the random way the dataset is split into training and test data. Therefore, we decided to run the training process 100 times for each network and keep the average accuracy in order to get a result that accurately represents the capabilities of each algorithm. The results of these tests are presented in Table IV. All training was run on an NVIDIA GeForce RTX 3080 GPU.

TABLE IV
AVERAGE ACCURACY OF EACH CLASSIFICATION ALGORITHM ON THE ENTIRE DATASET

| ***Classifier*** | ***Accuracy (Random split)*** | ***Accuracy (Cross-Validation)*** |
|---|---|---|
| $k$-Near | 88.86% | 64.23% |
| DTree | 81.12% | 66.01% |
| GNB | 79.63% | 67.02% |
| RNN | 81.26% | 67.63% |
| FNN | 94.31% | 75.65% |
| GRU | 91.99% | 76.57 % |
| LSTM | 92.24% | 76.86 % |
| SVM | 94.62% | 79.55% |
| CNN | 96.95% | 87.67% |

## V. Discussion

The results we obtained regarding the accuracy of each algorithm seem to confirm what is known about their strengths and limitations. Out of the traditional machine learning algorithms, the relatively accurate performance of the SVM algorithm is unsurprising since it is well suited for classification tasks on small datasets [18]. We expected neural networks with a recurring element to be suitable for this task because of the importance of the position of each joint relative to time. Therefore, it is unsurprising that the GRU and LSTM Networks, which can deal with long-term dependencies, outperform the FNN. The CNN achieved the best performance, which can be explained by the fact that it is the only one that can consider the relationships between nearby joints. Nevertheless, it should be noted that modeling these relationships and optimizing the architecture of the neural networks is beyond the scope of this paper, which limits the resulting accuracy of the algorithms.

## VI. Conclusions and Future Work

In this paper, we studied the performance of different classifiers, when applied to identifying dance steps. For this purpose, we introduced a novel dataset consisting of videos of people performing traditional Greek dances and annotated the videos separating each performance into steps. Our experimental results indicate that out of the classifiers we utilized, the CNN-based classifier outperforms all other algorithms both on randomly split data and when performing cross-validation. Our results can serve as a benchmark for researchers who use the dataset in the future. A probable direction for future work would be expanding the dataset to include new dances and more individuals, as well as possibly adding videos with more than one performer. Another possibility would be working on refining the architecture of the CNN classifier, such as by adding depth to it, as well as comparing its performance to other Neural Network architectures not included in this paper.

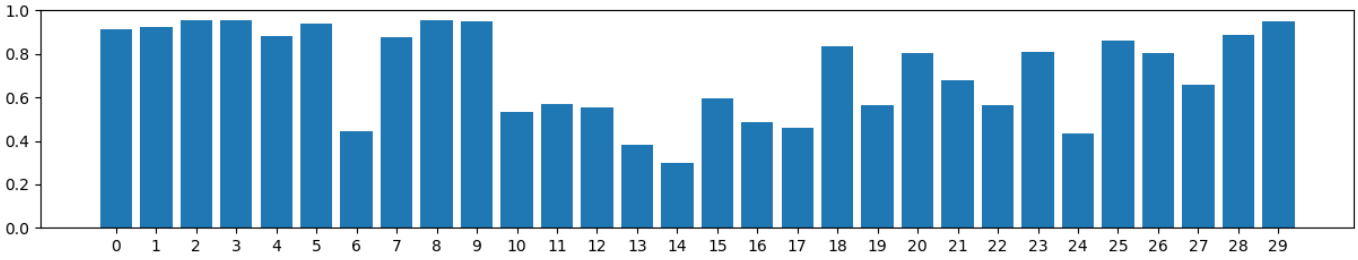


Fig. 2. Recall metric for each class when using the CNN algorithm and performing cross-validation.

## ACKNOWLEDGMENT

This research has been co-financed by the European Regional Development Fund of the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship, and Innovation, under the call RESEARCH – CREATE – INNOVATE (project code: T2EΔK-04800).

## REFERENCES

[1] I. Rallis, A. Voulodimos, N. Bakalos, E. Protopapadakis, N. Doulamis, and A. Doulamis, "Machine learning for intangible cultural heritage: a review of techniques on dance analysis," *Visual Computing for Cultural Heritage*, pp. 103–119, 2020.

[2] N. Lazaros, G. C. Sirakoulis, and A. Gasteratos, "Review of stereo vision algorithms: from software to hardware," *International Journal of Optomechatronics*, vol. 2, no. 4, pp. 435–462, 2008.

[3] E. Protopapadakis, A. Voulodimos, A. Doulamis, S. Camarinopoulos, N. Doulamis, and G. Miaoulis, "Dance pose identification from motion capture data: a comparison of classifiers," *Technologies*, vol. 6, no. 1, p. 31, 2018.

[4] N. Bakalos, E. Protopapadakis, A. Doulamis, and N. Doulamis, "Dance posture/steps classification using 3d joints from the kinect sensors," in *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*. IEEE, 2018, pp. 868–873.

[5] H.-B. Zhang, Y.-X. Zhang, B. Zhong, Q. Lei, L. Yang, J.-X. Du, and D.-S. Chen, "A comprehensive survey of vision-based human action recognition methods," *Sensors*, vol. 19, no. 5, p. 1005, 2019.

[6] L. Wang, D. Q. Huynh, and P. Koniusz, "A comparative review of recent kinect-based action recognition algorithms," *IEEE Transactions on Image Processing*, vol. 29, pp. 15–28, 2019.

[7] Y. Du, W. Wang, and L. Wang, "Hierarchical recurrent neural network for skeleton based action recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1110–1118.

[8] C. Li, Q. Zhong, D. Xie, and S. Pu, "Skeleton-based action recognition with convolutional neural networks," in *2017 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. IEEE, 2017, pp. 597–600.

[9] Y. Du, Y. Fu, and L. Wang, "Skeleton based action recognition with convolutional neural network," in *2015 3rd IAPR Asian conference on pattern recognition (ACPR)*. IEEE, 2015, pp. 579–583.

[10] M. Raptis, D. Kirovski, and H. Hoppe, "Real-time classification of dance gestures from skeleton animation," in *Proceedings of the 2011 ACM SIGGRAPH/Eurographics symposium on computer animation*, 2011, pp. 147–156.

[11] E. Protopapadakis, A. Voulodimos, A. Doulamis, and S. Camarinopoulos, "A study on the use of kinect sensor in traditional folk dances recognition via posture analysis," in *Proceedings of the 10th International Conference on PErvasive Technologies Related to Assistive Environments*, 2017, pp. 305–310.

[12] A. Kitsikidis, K. Dimitropoulos, S. Douka, and N. Grammalidis, "Dance analysis using multiple kinect sensors," in *2014 international conference on computer vision theory and applications (VISAPP)*, vol. 2. IEEE, 2014, pp. 789–795.

[13] A. Kitsikidis, N. Boulgouris, K. Dimitropoulos, and N. Grammalidis, "Unsupervised dance motion patterns classification from fused skeletal data using exemplar-based hmms," *International Journal of Heritage in the Digital Era*, vol. 4, no. 2, pp. 209–220, 2015.

[14] N. Bakalos, I. Rallis, N. Doulamis, A. Doulamis, E. Protopapadakis, and A. Voulodimos, "Choreographic pose identification using convolutional neural networks," in *2019 11th International Conference on Virtual Worlds and Games for Serious Applications (VS-Games)*. IEEE, 2019, pp. 1–7.

[15] "Zed 2 stereo camera description," https://www.stereolabs.com/zed-2/, accessed: 2023-05-03.

[16] D. M. Farid, L. Zhang, C. M. Rahman, M. A. Hossain, and R. Strachan, "Hybrid decision tree and naïve bayes classifiers for multi-class classification tasks," *Expert systems with applications*, vol. 41, no. 4, pp. 1937–1946, 2014.

[17] N. Bhatia *et al.*, "Survey of nearest neighbor techniques," *arXiv preprint arXiv:1007.0085*, 2010.

[18] S. Abe, *Support vector machines for pattern classification*. Springer, 2005, vol. 2.

[19] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[20] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[22] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.