electronics

MDPI

Article

Light-Weight Classification of Human Actions in Video with Skeleton-Based Features

Włodzimierz Kasprzak * and Bartłomiej Jankowski

Institute of Control and Computation Engineering, Warsaw University of Technology, 00-661 Warszawa, Poland; bartlomiej.jankowski2.stud@pw.edu.pl

* Correspondence: wlodzimierz.kasprzak@pw.edu.pl

Abstract: An approach to human action classification in videos is presented, based on knowledge-aware initial features extracted from human skeleton data and on further processing by convolutional networks. The proposed smart tracking of skeleton joints, approximation of missing joints and normalization of skeleton data are important steps of feature extraction. Three neural network models—based on LSTM, Transformer and CNN—are developed and experimentally verified. The models are trained and tested on the well-known NTU-RGB+D (Shahroudy et al., 2016) dataset in the cross-view mode. The obtained results show a competitive performance with other SOTA methods and verify the efficiency of proposed feature engineering. The network has a five times lower number of trainable parameters than other proposed methods to reach nearly similar performance and twenty times lower number than the currently best performing solutions. Thanks to the lightness of the classifier, the solution only requires relatively small computational resources.

Keywords: human actions; neural network models; skeleton data; machine learning; NTU-RGBD dataset; OpenPose

check for updates

Citation: Kasprzak, W.; Jankowski, B. Light-Weight Classification of Human Actions in Video with Skeleton-Based Features. *Electronics* **2022**, *11*, 2145. https://doi.org/10.3390/electronics11142145

Academic Editor: Muhammad Muaaz

Received: 26 May 2022
Accepted: 6 July 2022
Published: 8 July 2022

1. Introduction

Human activity recognition (HAR) has recently caught the attention of the computer vision community since it promises to drive solutions that make our life better and safer, such as human–computer interactions in robotics and gaming, smart video surveillance in hospitals, industry and outdoor events, and social activity recognition in Internet data [1,2]. The topic of recognizing a person's activity covers a wide range of approaches. Generally, activity recognition can be divided into two categories due to the way the data are obtained [3]:

1. based on signals from worn sensors or mobile devices;
2. vision-based—people are depicted in sequences of images such as movies or movie clips. This data can be obtained using various types of equipment, e.g., a stationary or mobile digital camera or a Kinect-like sensor.

The first approach requires cooperation from an observed person and the use of highly specialized equipment. The second approach involves collecting data in the form of movie clips, where data acquisition is rather a simple and inexpensive process. A vision-based approach can, therefore, be applied in everyday life due to the implementation of a new solution into already existing structures. Judging a person's activity on the basis of videos also has the advantage that it does not require a person to have any additional item, so it does not interfere in any way with the way the activity is performed.

Typically, human activity recognition in a video first requires the detection of human body parts or key points of a human skeleton. In early solutions, hand-designed features, such as edges, contours, Scale-Invariant Feature Transform (SIFT) and Histogram of Oriented Gradients (HOG) [4], have usually been used for the detection and localization of human body parts or key points in the image. More recently, Neural Network-based solutions have been successfully proposed, e.g., based on Deep Neural Networks (e.g., LSTM [5], CNN [6–8], Graph CNN [9]), as they have the capability to automatically learn rich semantic and discriminative features. Currently, the approaches to vision-based human activity recognition can be divided into two main categories: activity recognition directly in video data [10], or skeleton-based methods [6,9,11], where the 2D or 3D human skeleton are detected first, even by specialized devices, such as the Microsoft Kinect. Some popular solutions to human skeleton estimation (i.e., detection and localization) in images exist: DeepPose [12], DeeperCut [13], OpenPose [14] and HRNet [15]. The skeleton-based methods compensate for some of the drawbacks of vision-based methods, such as assuring the privacy of persons and reducing the scene lightness sensitivity.

In this work, we focus on human action recognition in video, assuming the existence of skeleton data for the selected video frames. The goal is to design a solution that proves the following hypotheses:

1. it is useful to extract meaningful information from the tracking of skeletal joints by employing classic functions instead of training relational networks to perform function approximation;
2. a feature engineering step, giving a 2D map of spatio-temporal joints locations, allows the use of a light-weight CNN-based network for action classification, replacing the need for heavy-weight LSTMs or 3D CNNs for this task.

There are four remaining sections of this work. Section 2 describes recent approaches to human pose estimation and action recognition. Our solution is presented in Section 3, where we introduce our feature engineering algorithm and three network models. In Section 4, experiments are described to verify the approach. Finally, in Section 5, we summarize our work and the achieved results.

2. Related Work

2.1. Human Pose Estimation

Human pose estimation in images means detecting and localizing humans. Typically, a single human's pose is described by key points or body parts, which are expressed in camera coordinates. In the past, mainly hand-crafted features have been used, such as edges, contours, Scale-Invariant Feature Transform (SIFT) or Histogram of Oriented Gradients (HOG). However, these approaches have produced modest performance when it comes to accurately localizing human body parts [16]. With the development of Convolutional Neural Networks (CNNs), the performance in solving human pose estimation problems has improved constantly and has been significantly higher than the traditional methods [17].

There are three fundamental architectures, AlexNet [18], VGG [19] and ResNet [20], which have been employed as the backbone architecture for many human pose estimation studies [21]. Released in 2012, AlexNet has been considered one of the backbone architectures for many computer vision models. The DeepPose software employed AlexNet for estimating human poses. Popular works in pose estimation, OpenPose [14], and human parts detection, Faster RCNN [22], have used VGG and achieved state-of-the-art performance in visual human estimation. After the release of ResNet, many works on human pose estimation have applied it as a backbone (e.g., [13,23]).

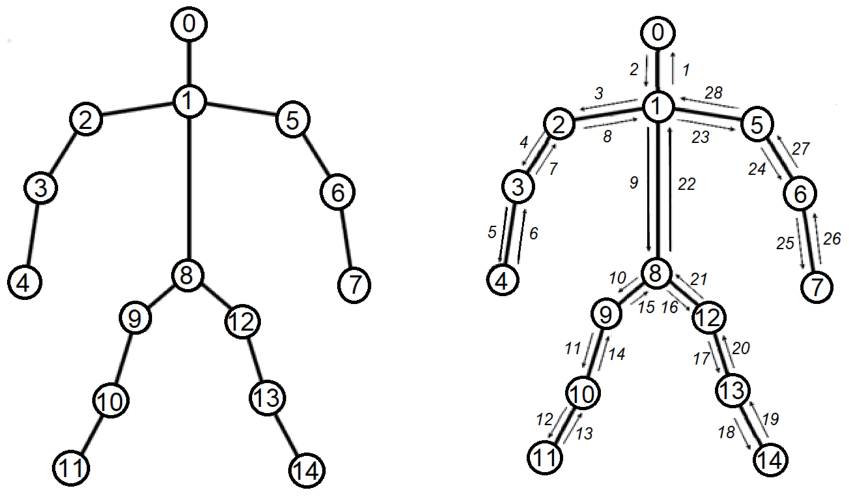
2.2. Human Action Classification

Human action is a dynamic process that can be perceived in a time sequence of images (or video frames) by a motion of the entire human body or particular body parts. The goal of a human action classifier is to assign to an image sequence an action class, i.e., to select a model from the set of class models that best matches the observed human action. In our review, we focus on solutions that classify skeleton data sequences extracted from the image sequence.

Liu et al. [5] use a modified LSTM module and a new arrangement of such modules. The modification consists of the introduction of "Trust Gates", which asses the reliability of the skeletal joint positions based on the temporal (i.e., at a previous time point) and spatial context (i.e., relative to a previously analyzed joint). This feature helps the storage cells judge which locations should not be remembered and which ones should be kept in memory. The new LSTM network, called ST-LSTM, is a 2D arrangement of modules capable of capturing spatial and temporal dependencies at the same time. In their work, the authors drew attention to the importance of properly ordering the joints into a sequence, i.e., enrolling the data from one image along the spatial direction for the network. Entering the network are features related to the skeletal joints of a person, given in a predefined order. Figure 1 shows two ways how a feature vector will take information from joints given in the following order (in the drawings, a circle with a number symbolizes a human junction, while the labels of arcs give the tree traversal order):

- SIMPLE: 0-1-2-3-4-5-6-7-8-9-10-11-12-13-14
- TREE-LIKE: 1-0-1-2-3-4-3-2-1-8-9-10-11-10-9-8-12-13-14-13-12-8-1-5-6-7-6-5-1

In the case of a tree traversal representation (TREE-LIKE), the data are given in a redundant way because the information from the joints is duplicated. However, according to the authors, this allows the network to perceive local dependencies due to the arrangement in the vicinity of the connectors in a natural way. This early study (in 2016) achieved a modest action classification performance of 77.7% on the NTU-RGB+D dataset [17] (in the CV mode).



(**a**) (**b**)

Figure 1. The mapping of a skeleton to a feature vector: (**a**) SIMPLE order, (**b**) TREE-LIKE order.

In the work of Liang et al. [8], 2D convolutional networks are used, following the interesting idea of analyzing symbolic data sequences arranged as pseudo-images. The feature vectors created for successive video frames are combined into a matrix. Thus, one matrix dimension provides temporal context information, whereas the other encodes a geometric context. A 2D convolution operator applied on such a pseudo-image finds local relations in the image-time space. The authors use three types of features extracted from skeletal data: positions of joints, line segment lengths and their orientations. These data are processed by a three-stream CNN (3S-CNN). The network is also divided into three phases. In the layers belonging to the first phase, the three data streams are processed independently. The second phase combines different streams of features and also processes them through successive layers. The last phase of the network consists of three independent, fully connected layers producing one probability vector per layer at the output. Classification consists of voting based on these three independent vectors. This recent study (in 2019) achieved a high action classification performance of 93.7% on the NTU-RGB+D dataset (in the CV mode).

The work by Duan et al. [24] applies 3D convolutional networks for human action classification. A network model, among others, consists of three layers of the ResNet50 network—the second, third and fourth. Heat maps of joint locations, obtained for subsequent image frames, are presented at the input to this network. A single heat map is created by a two-step approach. At first, an estimate of the area occupied by a person is made by the Faster-RCNN network. Then, this image area is passed to the HRNet-w32 for pose estimation. The resulting heat maps, as pose representations in single frames, are combined into one tensor—a stack of maps. This stack is passed to the 3D CNN with a fully connected classification layer. According to the achieved results, the use of heat maps instead of strict junction locations increases the accuracy by 0.5–2%. This recent solution (from 2021) achieved the best performance of 97.1% on the NTU-RGB+D dataset (in the CV mode).

2.3. Analysis

The available methods and tools for human pose estimation are characterized by high efficiency. Their use in our solution to action recognition will save a huge implementation effort. The action classification approaches are mainly based on silhouette detection or skeleton data estimation.

A natural and effective approach to image sequence analysis has always seemed to be recursive networks, e.g., LSTM. Architectures based on recursive networks are designed to include a time dimension by default. The sequence of skeleton joints can easily be transformed into a pseudo-image or a heat map, a proper input to a CNN. Recent research results and the referred work show that it is possible to apply convolutional networks to problems of a sequential nature. It would, therefore, be reasonable to apply a method based on a CNN family. The transformation of the joints to the image potentially brings the problem back to image analysis, but the image no longer contains any background noise.

It also seems important to pay attention to the order of the given joints of the human silhouette and the way of representing the skeletal information. Manipulating the order of joints can help networks find local dependencies. Making it easier to find dependencies, on the other hand, may allow you to reduce the number of weights by creating a smaller network.

2.4. Contribution of This Work

Based on the above conclusions, we designed an approach to human action classification that has the following novelty:

1. inventing a new mapping of the skeleton joints set to a feature vector, which makes the representation of local neighborhoods explicit; such a representation is experimentally proven to be most suitable for a CNN classifier;
2. proposing a feature engineering algorithm, which refines the skeleton information by canceling noisy joints, filling gaps of joints information by tracking skeletons in time, and normalizing the spatial information contained in feature vectors;
3. defining a network architecture based on a CNN with skip connections that proves to perform better than LSTM and Transformer networks, even with a much lower number of trainable parameters than in competitive solutions;
4. the proposed approach can be applied both to 2D and 3D skeleton data extracted from video frames or depth images; several extremely light-weight models (with up to 250 k parameters only) were trained and tested on a popular dataset, the NTU-RGB+D video dataset [25].

2.5. Video Datasets

The dataset "NTU RGB+D" [25] is the basic set used in this work. It was made by ROSE (Rapid-Rich Object Search Lab), which is the result of a collaboration between Nanang Technological University in Singapore and Peking University in China. Many works on human action recognition have already been validated on this dataset, and a website collects the achieved performance scores [17]. The set can be characterized as follows:

- Contains RGB videos with a resolution of 1920×1080 (pixels).
- Includes depth and infrared maps with a resolution of 512×424 (pixels).

- Each behavior of the set is captured by three cameras.
- Behaviors were performed by people in two settings (showing activities from different viewpoints).
- It consists of 56.880 videos showing 60 classes of behavior.

Among the classes of behavior, the most popular one is the "everyday activities" subset. It consists of 40 classes, as listed in Table 1.

Table 1. The "everyday activities" subset (40 classes) of the NTU-RGB+D dataset.

A1: drink water	A2: eat meal	A3: brush teeth	A4: brush hair
A5: drop	A6: pick up	A7: throw	A8: sit down
A9: stand up	A10: clapping	A11: reading	A12: writing
A13: tear up paper	A14: put on jacket	A15: take off jacket	A16: put on a shoe
A17: take off a shoe	A18: put on glasses	A19: take off glasses	A20: put on a hat/cap
A21: take off a hat/cap	A22: cheer up	A23: hand waving	A24: kicking something
A25: reach into pocket	A26: hopping	A27: jump up	A28: phone call
A29: play with phone/tablet	A30: type on a keyboard	A31: point to something	A32: taking a selfie
A33: check time (from watch)	A34: rub two hands	A35: nod head/bow	A36: shake head
A37: wipe face	A38: salute	A39: put palms together	A40: cross hands in front

The collection used consists of 37.920 video clips divided between 40 classes. Depth and infrared maps are omitted in this work.

The UTKinect-Action3D dataset [26] is the second set of people's activities used in this work. This set will be a secondary set, which means that only a testing of the developed model will be performed on it. The UTKinect dataset can be described by the following:

- Includes RGB videos with 640 × 480 resolution (pixels).
- Includes depth maps with a resolution of 320 × 240 (pixels).
- Activities were performed by 10 people. Each person repeated the performed activity 2 times. There are 10 activity classes.

In Table 2, the 10 classes of activities in UTKinect are listed.

Table 2. The 10 activity classes of the UTKinect-Action3D dataset.

A1: walk	A2: sit down	A3: stand up	A4: pick up
A5: carry	A6: throw	A7: push	A8: pull
A9: wave hands	A10: clap hands		

The clips in this collection are sourced as a series of photos in catalogs. To view a specific video, images within each catalog must be collected. A single video contains a person performing a series of 10 actions. A video is labeled with the action class, start photo and end photo of every action.

3. The Approach

The state-of-the-art in human skeleton detection in images is the OpenPose library. We shall rely on it to generate visual features that are representative of human pose classification. Our special interest is in the evaluation of images showing single persons performing some movement that can be interpreted as an action of a class. When more persons are visible in a video, we focus on the main person in the foreground.

3.1. Structure

We designed a complete solution for the classification of single-person actions in video clips. The solution consists of four main steps (Figure 2):

1. Frame selection: selecting a sparse sequence of frames from the video clip.
2. Pose estimation.
   This step is performed by an external OpenPose library (if not already existing in the dataset)—detecting and localizing 2D or 3D human skeletons in images or video frames.
3. Feature engineering.
   This step is accomplished by skeleton tracking, joint refinement and normalization and final ordering of joints. Several persons could potentially be present in a frame, but we are interested in the "main" person. The positions and size of persons can vary—a smart normalization is needed. Some joints are not always detected, and some others are detected with low certainty. Thus, smart tracking of a skeleton and refinement of the missing joints are needed.
4. Model training and testing: different networks and configurations are trained on the main dataset.
   A trained model, thanks to which it becomes possible to analyze the behavior of a person, is applied for the action classification of a feature map that is provided as its input. Several models are tested on the two datasets, and their performance is measured.

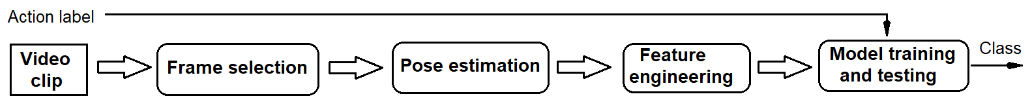


Figure 2. General structure of our approach.

Figure 3 shows how the representation of a clip is created.

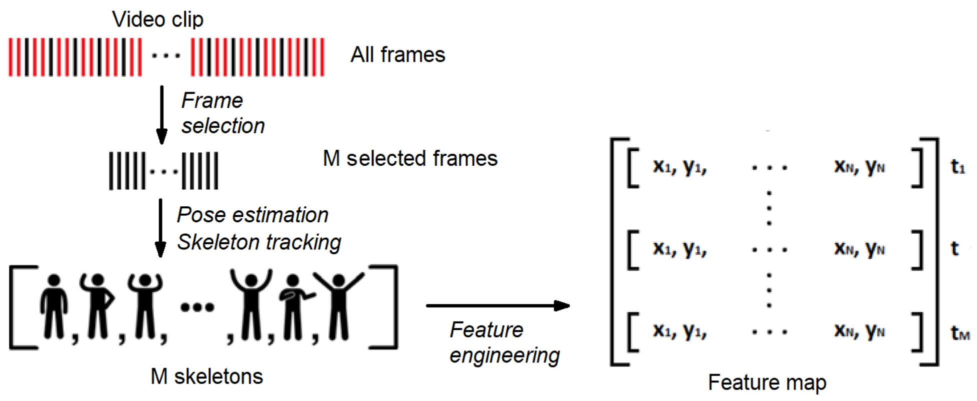


Figure 3. The feature map created from a time sequence of vectors of normalized and refined joint locations.

3.2. Frame Selection

Videos that are analyzed by the system have different lengths, i.e., they have different frame numbers. It is important that the input data of the action classifier is always of equal length. The way to achieve this unification is to choose M frames from all the frames in such a way that they almost uniformly cover the time interval between the first and the last frame of the clip. At the top of Figure 3, the video clip is illustrated by a sequence of line segments representing frames—among them, the black frames are selected for further processing. Let N be the number of frames in the clip (e.g., $N = 90$) and M the number of frames to be selected. We obtain a time interval increment as: $\delta = N/(M + 1.0)$. The M frames being selected, referred to by index $k = 1, \ldots, M$; have corresponding video indexes $I_k = round(k \cdot \delta)$.

3.3. Skeleton Estimation

OpenPose provides the ability to obtain 2D or 3D information about detected joints. If one selects the 3D option, views must be specified for the library to perform triangulation. The library returns for every joint the following data:

- (x, y, p)—for 2D joints;
- (x, y, z, p)—for 3D joints.

where (x, y) are the image coordinates of a pixel, z is the depth associated with the given pixel, p is the certainty of detection and is a number in the range <0:1>. We shall make use of the certainty value—if p falls below a threshold or is even 0, which means "no detection", the joint information will be canceled and later approximated from its neighbors. The OpenPose library offers the possibility to choose a model of a human figure. There are three models: "15 body", "18 body", and "25 body". The number in the name refers to the number of detectable joints. Table 3 lists the typically selected model of the "25 body".

Table 3. List of joints in the "25 body" model.

Number	Description
0	The main point of the head
1	Neck base
2, 5	Shoulders
3, 6	Elbows
4, 7	Wrists
8	The base of the spine
9, 12	Hips
10, 13	Knee
11, 14	Cube
15, 16, 17, 18	Extra head points
19, 20, 21, 22, 23, 24	Extra foot points

3.4. Feature Engineering

3.4.1. Skeleton Tracking

This task consists of the tracking of the most prominent skeletons over the frames of a video clip. In the case of many-person scenes, the sets of skeletons generated by OpenPose, are not uniquely indexed over the frame sequence. There may also be falsely detected skeletons for objects in the background, or a large mirror can lead to a second skeleton of the same person. Examples of wrongly detected skeletons in the frames of a video clip from the NTU RGB+D collection are shown in Figure 4. One can force the pose estimator automatically to select one skeleton from a set, but this would not give reliable results, as the decision must be made for every frame independently. As a result, one cannot avoid skips between persons in a scene to which the single available skeleton is assigned (see Figure 5).

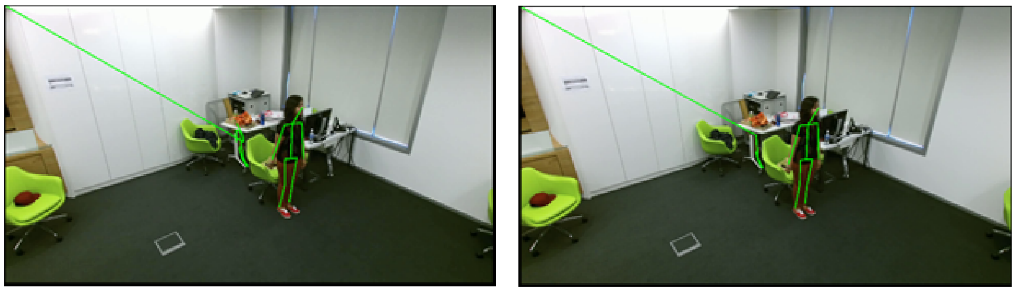


Figure 4. In multi-skeleton mode, OpenPose can return additional faulty skeletons and miss some joints. A default location of a nonexistent joint is (0, 0).

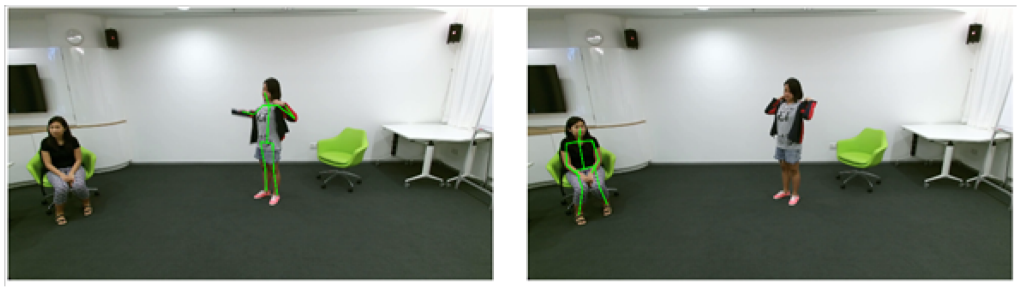


Figure 5. Forcing OpenPose to return one skeleton per image is not a proper solution.

The algorithm for tracking skeletons in many-skeleton frames can be seen as a multiple path search and can be solved in many ways. For example, beam search or some type of dynamic programming search could be used. Our algorithm for matching two sets of joints is described in Algorithm 1. It uses the *distance* function for two skeletons s_1, s_2, which is defined as the mean Euclidean distance between the locations of the corresponding joints of the two skeletons. If some joint was not detected, then its distance to the corresponding joint is omitted in the calculations of *distance*. The algorithm initializes paths for skeletons detected in the first frame and then runs a loop over the remaining frames trying to extend every path by the nearest skeleton detected in the next frame that is sufficiently close to the path's last skeleton. New paths can also start in later frames when apparently new persons appear on the scene. The main loop ends with an eventual pruning of weak paths by the function **SelectPaths** (see Algorithm 2). Let us note that for the last frame, the parameter *beam* is set to 1, and this function selects the final single best path. The main work here is performed by two subfunctions. The subfunction **GetScore** counts for the given *path* the sum of all detected joints weighted by their certainty for all unique skeletons assigned to the given path. By the term ″unique″ we mean that a repetition of skeletons, virtually added to continue a path, is avoided. Another subfunction **PrunePaths** is reorganizing the entire matrix P by keeping the b best-scored paths only in the first b columns of P.

3.4.2. Filling of Empty Joints

The selected sequence of skeletons is sometimes deteriorated by missing joints of some skeleton or by missing an entire skeleton in a frame. These misses of joints or virtual replications of skeletons introduce noise and may also destabilize the feature normalization step when the locations of required joints are missing. The procedure for the refinement of joints can be summarized as follows:

- for the initial frame, the position of a missing joint is set to its first detection in the frame sequence;
- in the middle of the frame sequence, a missing joint is set to a linear interpolation of the two closest-time known positions of this joint;
- joints that are lost at the end of the frame sequence receive the positions last seen.

In Figure 6, an example of the joint refinement process is shown. The lines in the left image that are connected to the upper left corner represent the lost joint (an undetected joint is represented by location (0, 0)). After this joint is approximated, it fits into the person-populated image region (as shown in the right image).

Algorithm 1 Skeleton tracking (the brackets /* */ represent comments of the pseudo-code).

```
SkeletonTracking($\mathbf{S}, \mathbf{P}, \mathbf{m}, M, N$)
{
 /* Let $\mathbf{S}$ be a set of all skeleton sets found in the frame set of the current video clip. Let
 $\mathbf{P}[f_{id}, s_{id}]$ be a matrix holding paths of skeletons, $s_{id} = 1, 2, \ldots, M$, over the frame sequence
 indexed by $f_{id} = 1, 2, \ldots, N$.
 Initially, $P[1,:]$ holds references to all skeletons detected in the first frame selected from the
 current movie clip. The number of simultaneously tracked paths is limited to $M$. Every
 empty place in $\mathbf{P}$ is marked as $dummy$. Let the pointer $m[f_{id}]$ refers to the number of true
 (non-dummy) skeletons in frame $f_{id}$. */
 for $t = 2, 3, \ldots, N$; do
 {
  /* Let $\mathbf{S}_t$ denotes the skeleton set detected in frame $f_t$. The set is assumed to be ordered
  according to decreasing size, as the foreground skeleton is expected to be large */

  (1) $\mathbf{IS} = refer(\mathbf{S}_t)$; /* introduce a list that temporary refers all new skeletons */
  (2) for every $n = 1, \ldots, m[t-1]$ do
   {$s_n = \mathbf{P}[t-1, n]$;
    if $s_n \neq dummy$ then
    {/* find the nearest skeleton $s_p \in \mathbf{IS}$ */ $s_p = \arg min_{p \in IS}\, distance(s_n, s_p)$;
     if $distance(s_n, s_p) < d_{thr}$ then {$\mathbf{P}[t, n] = refer(s_p)$; $\mathbf{IS}[p] = dummy$}
    }
  (3) set $l = 0$; for every $p = 1, \ldots, m[t]$ do
   {$s_p = \mathbf{P}[t-1, p]$;
    if $s_p \neq dummy$ then { $l = l + 1$; refer $s_p$ in $\mathbf{P}[t, m[t-1] + l]$ }
    $m[t] = m[t-1] + l$
  }
  (4) for every $n = 1, \ldots, m[t-1]$ do
   {if $\mathbf{P}[t, n]$ is $dummy$ then $\mathbf{P}[t, n] = \mathbf{P}[t-1, n]$
    /* virtually extend the previous path by previous-frame skeleton */
   }
  (5) SelectPaths($\mathbf{S}, \mathbf{P}, M, N, m, t$);
   /* eventually prune weak paths if needed; at the end of sequence—select the best one */
 }
```

Algorithm 2 Select paths.

```
SelectPaths($\mathbf{S}, \mathbf{P}, M, N, m, t$)
if $t == N$ (the last frame) then $beam = 1$ else $beam = 5$;
if $m[t] > beam$ do
{
for $k = 1, \ldots, m[t]$ do: $scores[k] = $ GetScore($\mathbf{S}, \mathbf{P}[k], M, N, k$);
PrunePaths($\mathbf{P}, M, N, m, t, beam, scores$);
}
```

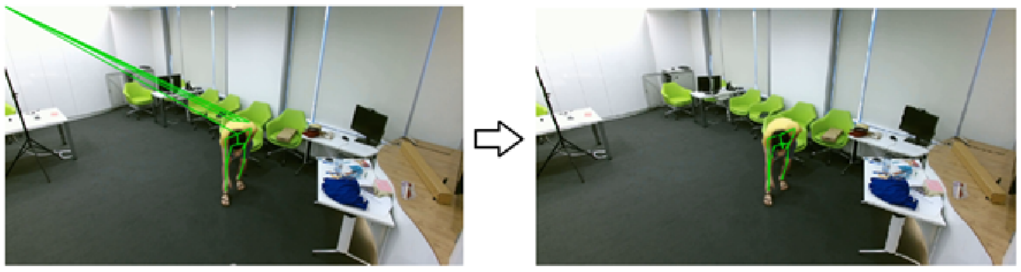


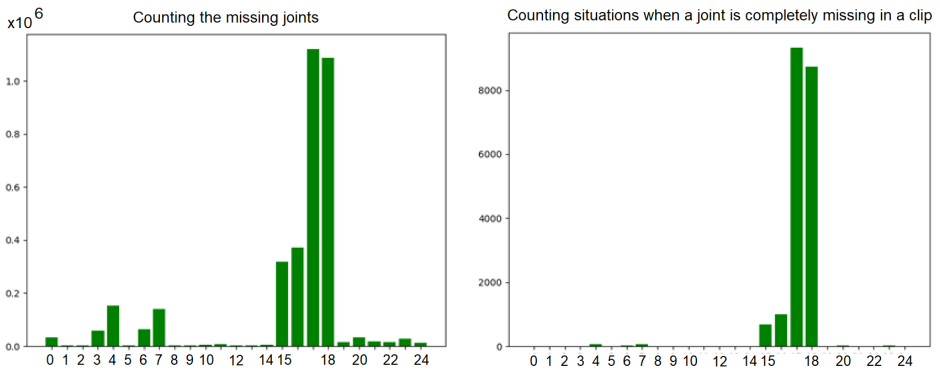
Figure 6. Filling empty joint slots by interpolation in space or time.

3.4.3. Normalization of Joints

The features presented to the classifier should be translation- and scaling-invariant. Thus, we need to define a reference point in the image and a reference line segment length for the entire sequence of frames. Both reference values should be related to the skeleton sequence. We have considered three ways of selecting the reference values for the normalization procedure:

1. **Min_max** —the reference point is the lower-left corner of the boundary box covering the action being performed over all frames—this point is found within all frames, so it does not change its position over time; the reference line segment is the diagonal of the area that a person occupied.
2. **Independent min_max**—a method similar to "min_max" with the difference that there are two reference sections—one for coordinates counted as the width of the frame and the other for coordinates counted as the height of the frame; scaling is performed independently for coordinates using the length of the corresponding line segments.
3. **Spine_using**—the reference point becomes the position of joint number 8, and the reference segment is the line segment between joints 1 and 8 (to be interpreted as the spine length)—the longest such segment in all considered frames of a video clip is taken; let us note that these two joints are usually well detected in an image.

In order to justify the choice of joints 1 and 8 in the spine method, two histograms were obtained. Figure 7a presents a histogram with the number of times the particular joints were missing in frames over a total of 37.920 clips. Figure 7b shows the number of times specific joints were not detected at all in a whole clip. In the first histogram, you can see that joints number 1 and 8 disappear very rarely. In addition, judging by the second histogram, it never happens that during a video clip, these key points of the silhouette did not appear at least once. One can also notice that the joints associated with the hands sometimes disappear throughout a clip. A detailed check showed that at least one hand always appears in a clip. Joints associated with the head (additional points of the head) very often disappear in the data returned by OpenPose. Thus, we shall skip them.



(**a**) (**b**)

Figure 7. Missing rates of particular joints: (**a**) in single frames of a clip; (**b**) in an entire clip.

3.4.4. Palm-Based Ordering of Joints

We propose a new ordering of joints in the feature vector, called palm-based (Figure 8).

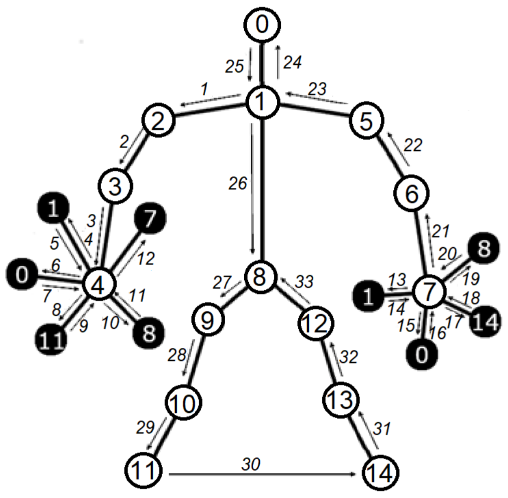


Figure 8. The proposed order of skeleton joints—the "palm-based" arrangement of 15 joints.

In the final feature vector, only the normalized location information of joints will be represented, as their certainty values will be explored in the feature engineering step. Some of the joints in the "palm regions" are represented multiple times. There are not any new joints added, but only a symbolic placement of existing hand joints. The feature vector is created in the direction indicated by the arrows starting from number 1 and ending at number 33. Thus, the feature vector holds joint locations in the following order:

1-2-3-4-1-4-0-4-11-4-8-4-7-1-7-0-7-14-7-8-7-6-5-1-0-1-8-9-10-11-14-13-12-8

It can be noted that in the created vector, the joints numbered 4 and 7 (representing palms) are repeated many times. Hands are the effectors of most activities. For this reason, we assume that the evaluation by a CNN of neighborhood relations between hands and other body parts should positively affect the ability to perceive human behavior descriptors.

The idea behind the palm-based arrangement of joints is to map 2D or 3D spatial neighborhoods between joint 4 (right palm) or joint 7 (left palm) and their spatial neighbor joints into explicit neighbors in the 1D feature vector. It is like performing a tree traversal of the subtrees with a root of 4 or 7, respectively, of the graph shown in Figure 8. Thus, the feature vector contains the two traversal paths "3-4-1-4-0-4-11-4-8-4-7" and "1-7-0-7-14-7-8-7-6", respectively. The relationship between both roots, joints 4 and 7, must be added to the feature vector one time only. We have assigned this neighborhood to the first subtree without repeating it in the other subtree. Thus, both subtrees are not completely symmetric. The subtrees represent relations between the palms and some most important remaining joints but not all of them. Joints numbered as 0, 1 and 8 are located at the main vertical axis of the silhouette—they represent the face and upper and lower spine ends. They are spatial neighbors for both palm joints and appear in both subtrees. There are two pairs of joints representing symmetric joints appearing in one of the subtrees only. Joints 3 and 6 represent the right and left elbow, while joints 11 and 14 represent the right and left ankle. In our view, it is sufficient to represent explicit neighborhoods of the left and right-side joints only. The relationships of the palm joints and remaining joints are not made explicit, as the latter appears in kinematic chains with the already considered "more important" neighbors.

3.4.5. Feature Map

From the skeletal vector obtained in a given frame, a data vector should be created for a specific person. Based on the analysis of joint detection reliability, we decided to use location information of a truncated set of joints but differently arranged into a sequence. In the experiments, the performance of the proposed networks will be compared with four alternative sequences of joints on their input (the joints from the 15-elementary subset are illustrated in Figure 1a):

- SIMPLIFIED: 0-1-2-4-5-7-8-11-14 (a 9-elementary simplification of the 15-elementary joints set, without joints representing elbows, knees and hips)
- SIMPLE: 0-1-2-3-4-5-6-7-8-9-10-11-12-13-14 (Figure 1a)
- TREE-LIKE: 1-0-1-2-3-4-3-2-1-8-9-10-11-10-9-8-12-13-14-13-12-8-1-5-6-7-6-5-1 (Figure 1b)
- Palm-Based: 1-2-3-4-1-4-0-4-11-4-8-4-7-1-7-0-7-14-7-8-7-6-5-1-0-1-8-9-10-11-14-13-12-8 (Figure 8)

3.5. Neural Network Models

For the implementation of the classifier, we consider three network types: LSTM, Transformer [27] and CNN. The first two types naturally deal with the time-sequences of data. The LSTM has been applied early for skeleton-based action recognition in the video. Transformer is a more recent network for data-sequence analysis with success applied in speech and language recognition. The CNN is our target network, as the input data will be organized into a 2D pseudo-image, representing the skeletal features ordered along the two axes: discrete time (image index) and skeletal joint index. Let M be the number of frames, N be the dimension of a feature vector per frame and K be the number of classes.

3.5.1. LSTM

The input to the LSTM is an M-sequence of N-dimensional vectors. We consider an architecture of P LSTM layers, where each module returns an L-dimensional embedding. A 50% dropout layer follows after every LSTM layer. Finally, a dense layer is given with K class likelihoods.

3.5.2. Transformer

Transformer networks use the MHSA (Multi-Head Self Attention) mechanism [27]. The input to the network is the $M \times N$ pseudo-image. At first, each frame vector receives a time embedding in the “PatchEncoder” layer. This is necessary due to the fact that this type of network is not based on a memory mechanism. For this reason, the elements (vectors) entering the network must carry this time information in their values. The next important layer is MHSA—it “learns” during training which values should be highlighted and which should be suppressed in order to improve the classification result. After the input is weighted by the attention and the result is normalized, the Transformer is completed by a multilayer perceptron. There can be P consecutive Transformers arranged. The $M \times N$ output is normalized and flattened, then goes through a 50% dropout layer and finally through a fully connected classification layer.

3.5.3. CNN

Figure 9 gives the structure of our parameterized CNN model. This is a two-stage CNN with a parallel skip connection, followed by a fully connected stage. Every CNN stage consists of two convolutional layers performing the main processing task and one parallel layer in the skip connection part. The main part initially processes the pseudo-image by a convolution layer with a kernel size 4×4 and stride $(1, 2)$ to obtain information about the four time moments of the two consecutive joints. In this way, information based on pairs of joints is created. This information is sent to a second convolution layer with a kernel size 3×3 and stride $(1, 1)$, whose task is to extract further information about the pairs of joints already connected by the previous layer. In parallel, the second part consists of a convolution layer without an activation function and serves as a skip connection mechanism. Skipping a connection is fulfilled by an additional layer. Finally, the first stage is completed by “MaxPooling” with a window size of 2×2. The second CNN stage differs from the first only in the number of filters. The two convolution stages are followed by two fully connected layers with L and K neurons, respectively, separated from the convolution network and from each other by two “dropout” layers with a rate of 50%. An early decision was to use a CNN with skip connections for the feature transformation stage.

In experiments, the best "light-weight" configuration of such a CNN was found to consist of two stages with two convolutional layers each in the main processing part.

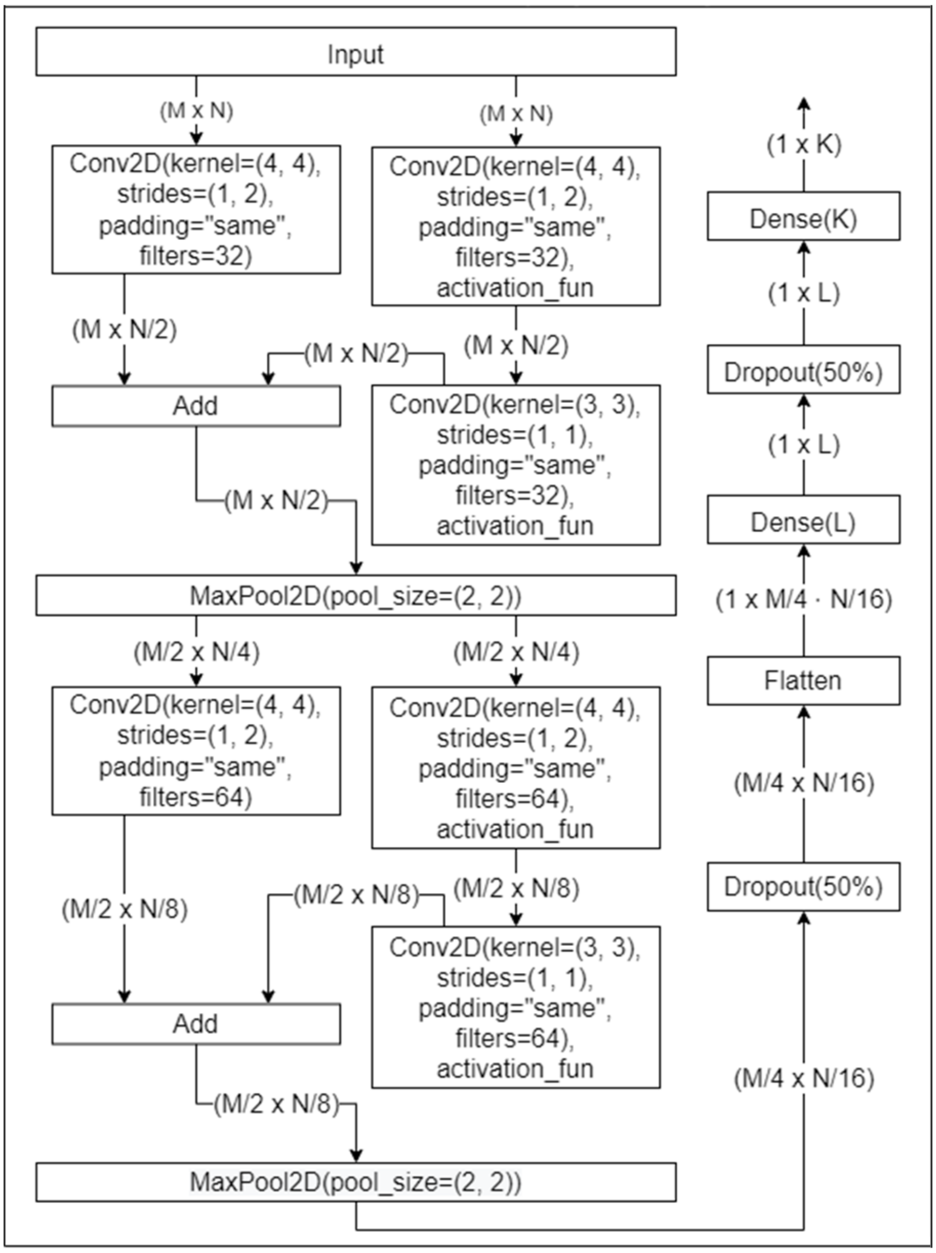

Input
(M x N)
(M x N)
Conv2D(kernel=(4, 4), strides=(1, 2), padding="same", filters=32)
Conv2D(kernel=(4, 4), strides=(1, 2), padding="same", filters=32), activation_fun
(M x N/2)
(M x N/2)
(M x N/2)
Add
Conv2D(kernel=(3, 3), strides=(1, 1), padding="same", filters=32), activation_fun
(M x N/2)
MaxPool2D(pool_size=(2, 2))
(M/2 x N/4)
(M/2 x N/4)
Conv2D(kernel=(4, 4), strides=(1, 2), padding="same", filters=64)
Conv2D(kernel=(4, 4), strides=(1, 2), padding="same", filters=64), activation_fun
(M/2 x N/8)
(M/2 x N/8)
(M/2 x N/8)
Add
Conv2D(kernel=(3, 3), strides=(1, 1), padding="same", filters=64), activation_fun
(M/2 x N/8)
MaxPool2D(pool_size=(2, 2))
(M/4 x N/16)
Dropout(50%)
(M/4 x N/16)
Flatten
(1 x M/4 · N/16)
Dense(L)
(1 x L)
Dropout(50%)
(1 x L)
Dense(K)
(1 x K)

Figure 9. The CNN model (M—frame number, N—feature number, K—class number, L—number of neurons in the layer).

3.6. Implementation

The approach was implemented in Python v. 3.8 and consisted of four basic parts (three scripts and a Browser part): ProcessClips.py, TrainModel.py, EvaluateModel.py and the MLflow for user supervision, activated in an Internet Browser. The first script was responsible for the interface with video clips and OpenPose, and for feature engineering. It prepared the data for training and model evaluation and stored them in .csv files corresponding to video clips.

The second script provided settings for ANN training ran the training and prepared its results—the training curves, model parameters and confusion matrix (Figure 10). The third script ran the test process and prepared results both as skeletons and classes added to the video clips, confusion matrices and text results in .csv files (Figure 11).

4. Results

The NTU-RGB-D dataset is typically applied to verify the performance of a solution in two modes: a cross-subject (CS) or a cross-view (CV) mode. In the cross-subject case, samples used for the training set show actions performed by several pre-specified actors, while test samples show the actions of the remaining actors. In the cross-view setting, samples recorded by two pre-specified cameras are used for training, while samples recorded by the remaining camera are used for testing. In this work, only the cross-view mode will be applied.

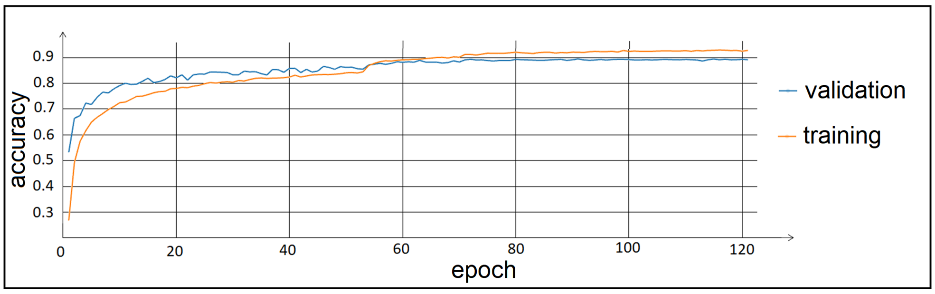
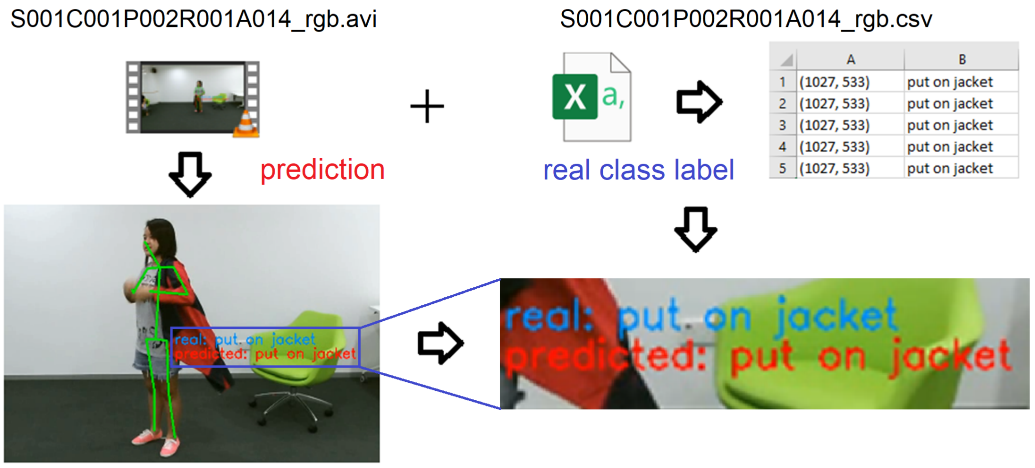


Figure 10. The "TrainModel" script monitors the training process and generates files with training results, such as performance curves and confusion matrix.



	A	B
1	(1027, 533)	put on jacket
2	(1027, 533)	put on jacket
3	(1027, 533)	put on jacket
4	(1027, 533)	put on jacket
5	(1027, 533)	put on jacket

Figure 11. The "EvaluateModel" script monitors the testing process. For example, it is possible to compare the ground truth class with the actual class for a given video clip.

4.1. Model Comparison

This section will present the results of the tests, based on which of the three model parameters and learning hyperparameters were selected. For tests of the LSTM model, the following parameters were fixed:

- Hyperparameter: initial learning constant $= 1 \times 10^{-2}$, constant learning reduction rate = 0.3, minimum value of learning constant = 1×10^{-4}, batch size = 100;
- Data sampling: frame number = 20;
- Feature engineering: normalization method = independent min max.

The following optimal settings of the LSTM model were obtained:

- Network parameter: number of LSTM units (L) = 100, number of LSTM layers (P) = 1, hidden layers activation function = tanh;
- Feature engineering: joint order = SIMPLE.

Regarding the required number of LSTM layers, the value of P between 1 and 4 was tested. The number of units (neurons) of the LSTM network (L) was checked from 10 to 200. Four types of activation functions were tested: sigmoid, tanh, ReLu and GeLu. It turned out that only tanh brought decent results. Finally, results obtained for different joint ordering schemas are given in Table 4.

Table 4. Optimal results for the LSTM network under different joint ordering schemas.

Accuracy	Simplified	Simple	Tree-like	Palm-Based
training	95.3	94.2	90.3	92.8
test	86.1	86.8	83.5	85.3
parameters	142.140	146.940	158.140	162.140

For the Transformer model, the following parameters were fixed or optimal settings were found:

- Hyperparameters (fixed): initial learning constant $= 1 \times 10^{-2}$, constant learning reduction rate = 0.3, minimum value of learning constant = 1×10^{-4}, batch size = 100;
- Data sampling (fixed): frame number = 20;
- Network parameters: number of MHSA units per layer (L) = 100, number of Transformer layers (P) = 2, hidden layers activation function = ReLu;
- Feature engineering: normalization method (fixed) = "independent min max"; joint order = SIMPLE.

For the number P of Transformer layers, values 1, 2, 3, 5, and 10 were checked. The number of units per layer (L) was alternatively set to 1, 2, 3, 5, 10 and 20. The best results were obtained for P = 2 or 3 and L = 10 or 20. However, it is difficult to notice any dominant value of these characteristics. This shows the small impact of these parameters on the model's performance. The results of the Transformer network, obtained for different joint ordering schemas, are given in Table 5. Changing the activation function does not have a great impact on the results of the training. However, ReLu and GeLu gave the best results, and ReLu was chosen.

Table 5. Optimal results for the Transformer network under different joint ordering schemas.

Accuracy	Simplified	Simple	Tree-like	Palm-Based
training	92.5	96.5	87.4	84.2
test	81.5	84.7	80.3	78.3
parameters	76.794	185.790	620.434	836.844

An important conclusion from the tests of the first two models is that the special schemas of tree-like and palm-based ordering of the joints are not suitable for them. They even lead to a visible deterioration of accuracy, especially in the Transformer model. This observation can be explained by the nature of relationships already created in their first layers by the LSTM and Transformer networks. They are looking for relationships between all the features in the global sense. When compared with the simplified feature vector, the higher dimensional "tree-like" or "palm-based" feature vectors provide no additional input information for them. The extension of the input vector only leads to a higher number of trainable parameters in the first layer and the generation of redundant relationship information.

The CNN model was initially tested by fixing the following parameters:

- Hyperparameter: initial learning constant $= 1 \times 10^{-3}$, constant learning reduction rate = 0.3, minimum value of learning constant = 1×10^{-5}, batch size = 100;
- Network layers: two convolution stages with three layers each (arranged in two streams: 2 + 1), two dense layers with two 50% dropout layers; 32 and 64 filters in the convolution layers of stage 1 and 2, respectively;
- Data sampling: frame number = 20;
- Feature engineering: normalization method = Independent min_max.

The following optimal settings for the CNN model were obtained:

- Network parameter: number of CNN units (L) = 100, number of CNN layers (P) = 1, hidden layers activation function = GeLu, convolution layer activation function = GeLu;
- Feature engineering: joint order = Palm-based.

The selection of the activation function was subjected to experiments. For the convolution layers, GeLu turned out to be better than ReLu. For hidden layers, the best two functions (out of the four tested) were tanh and GeLu.

For the number P of CNN layers, values 1, 2, 3, 5, and 10 were checked. The number of units per layer (L) was alternatively set to 1, 2, 3, 5, 10 and 20. The best results were obtained for P = 2 or 3 and L = 10 or 20. The results of the CNN network, obtained for different joint ordering schemas, are given in Table 6.

Table 6. Initial results for the CNN network under different joint ordering schemas.

Accuracy	Simplified	Simple	Tree-like	Palm-Based
training	95.4	96.5	94.7	97.7
test	86.0	87.5	87.0	89.7
parameters	149.068	181.068	213.068	245.068

Based on the initial tests, it can be noted that the CNN model already performed well with the standard joint ordering in which no duplication of joints appears. However, the introduction of joint redundancy for the purpose of local context representation is beneficial for this model. For the "palm-based" feature vector, the results are improved by 2.2% at the expense of a jump in the number of learning parameters from 181 to 245 thousand (26.5%). This result can be explained by the nature of CNN. The first layer pays attention to local relationships in the image (or feature map), and steadily, with every next layer, more abstract and global relations are unveiled. Exactly to suit this mechanism, the relationships between spatially different joints were made explicitly local in the "palm-based" feature vector.

4.2. CNN Model Optimization

We evaluated the influence of our new joint ordering scheme (palm-based) on classification accuracy by comparing results obtained with this scheme and the basic SIMPLE scheme. The aim of the first series of experiments was to explore the performance of the CNN model for different lengths of the input frame sequence. The batch size was set to 50. The results are summarized in Table 7.

Table 7. The influence of frame number per video clip ([7–25]) and the joint ordering scheme (SIMPLE vs. Palm-Based) on the test classification accuracy of the CNN model.

Joints Order	7 Frames	10 Frames	15 Frames	20 Frames	25 Frames
SIMPLE (%)	83.8	85.7	87.2	88.1	**88.5**
- Parameters	135k	143k	156k	180k	194k
Palm-Based (%)	85.0	87.6	89.4	90.1	**90.3**
- Parameters	153k	168k	194k	219k	245k

It can be observed that the classification accuracy is steadily increasing when the frame number increases and that the palm-based feature version steadily keeps an advantage over the basic version, ranging from 1.2% to 1.8%.

The aim of the second experiment series was to explore the performance of the CNN model for different lengths of the batch size during training. The frame number was set to 20. The results are summarized in Table 8.

Table 8. The influence of batch size ([10–100]) and the joint ordering scheme (SIMPLIFIED vs. Palm-Based) on the test classification accuracy for the CNN model.

Batch Size	**10**	**50**	**75**	**100**
SIMPLE (%)	87.8	**88.3**	87.5	87.4
Palm-Based (%)	88.8	**90.1**	89.7	89.7

It can be observed that for both feature versions that the classification accuracy achieves its maximum for a batch size of 50 and that the palm-based feature version is more sensitive to batch size changes at the low values of the batch size, i.e., [10–50], than on the high end values, i.e., [50–100].

In the third experiment series, we explored the performance of the CNN model under three different feature normalization strategies—min_max, independent min_max, spin_using. The frame number was 20 (as seen before, it would be optimal to set it to 25). The results are summarized in Table 9.

Table 9. The influence of three feature normalization strategies and the joint ordering scheme (SIMPLE vs. Palm-Based) on the test classification accuracy for the CNN model.

Normalization	**Min_Max**	**Independent Min_Max**	**Spin_Using**
SIMPLE (%)	87.2	87.5	**88.3**
Palm-Based (%)	89.8	89.9	**90.1**

It can be observed that for both feature versions, there is no significant influence of the feature normalization strategy on the classification accuracy. Still, the spin_using strategy induces a small advantage for both feature versions. As observed before, the palm-based feature version has an advantage in classification accuracy by 1.8–2.6%.

4.3. Experiment with the UTKinect Dataset

Finally, we also ran a cross-dataset experiment using an NTU RGB+D pre-trained CNN for testing on the UTKinect set. A problematic issue when using both collections is that the second test contains only 5 image sequences and 10 action types. For this reason, a 5-frame CNN network was pre-trained on the “NTU RGB+D” set. In this network, the last layer has been changed from 40 classes of actions to 10 classes. The UTKinect set was divided into a training and test set in a ratio of 9:1. No validation subset was created. The training process has been limited to 50 epochs. The number of network weights to be trained was 110 k. The averaged results of five tries with different splits into training and test subsets are as follows: training accuracy 98%, test accuracy 90%.

4.4. Comparison with Top Works

Our two best-performing solutions are compared to the top solutions of action recognition tested on the NTU RGB+D dataset (see Table 10).

Table 10. Comparison of our best solutions with related work for the NTU-RGB+D dataset (40 action classes, CV mode).

Method (Ref. No., Year)	**Test Accuracy (%)**	**Parameters**
ST-LSTM ([5], 2016)	77.7	>2 M
3S-CNN ([8], 2019)	93.7	*unknown*
Pose3D ([24], 2021)	97.1	2 M
CNN—SIMPLE	88.5	181 k
CNN—Palm-Based	90.3	235 k

There is a tradeoff visible between the complexity of a solution and classification accuracy (ours vs. 3S-CNN). However, the relatively early methods have both performed worse even with 10 times heavier networks (ours vs. ST-LSTM). Only the very recent mature solution, which tops all rankings and is based on Graph CNNs, surpasses our solution.

4.5. Analysis of Class Confusions

Figure 12 shows the confusion matrix obtained for 40 classes of the NTU-RGB+D dataset. Due to a lack of space, only the main confusion cases are shown in more detail. The confused classes can be divided into three groups. Actions numbered 11, 12, 29 and 30 form the first group. People performing these activities are usually inclined over one of several objects, i.e., a phone, a book, a notebook or a tablet. The skeletal system of these persons is quite similar. The second group of activities consists of numbers 10 and 34. These actions are clapping and rubbing hands, where people's stature is often quite similar. The last group consists of activities numbered 31 and 32, i.e., pointing with a finger and taking a photo with a selfie phone. These actions may seem to be much different. However, it should be known that the simplified skeleton model does not have a representation of fingers. Thus, these two behaviors are observed as putting a hand in front of a person. The accuracy of the network with 40 classes was, in this particular case, equal to 89.6%. Let us combine the three subsets of "similar" classes into separate meta-classes ($M1, M2, M3$): $\{11, 12, 29, 30\} \rightarrow M1$, $\{10, 34\} \rightarrow M2$, $\{31, 32\} \rightarrow M3$. Thus, we get a 35-class problem (32 "normal" classes and 3 meta-classes) The mean accuracy of such a classification problem would increase to 93.5%.

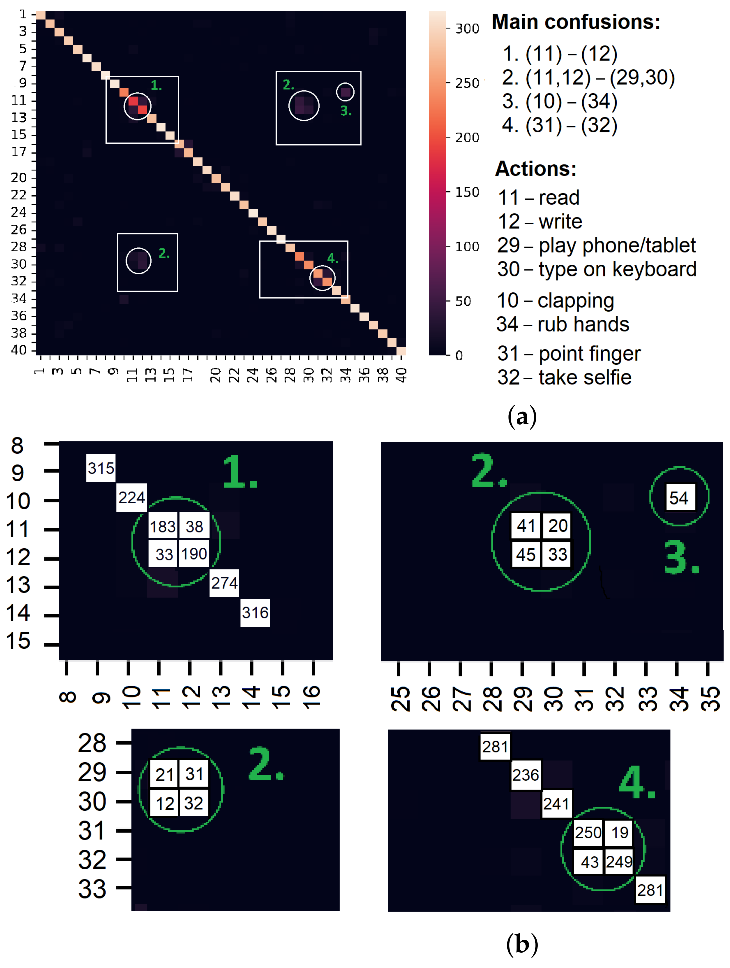


Figure 12. The main confusion between 40 classes of the NTU-RGB+D action dataset: (**a**) identification of four main confusion cases (given by green numbers 1,2,3 and 4, where confusion no. 2 is a symmetric relation) in the 40×40 confusion matrix, (**b**) a close view of the main four cases of class confusion.

5. Discussion

A light-weight approach to human action recognition in video sequences was proposed, implemented and tested. For the detection of human skeletons in the image, the state-of-the-art OpenPose library, an efficient deep network solution, was applied. Our main focus was on the improvement of skeleton data quality and on a light-weight network model. Particular novelty constitutes the palm-based arrangement of skeleton joints, which supports the use of a small CNN network. Thus, the main benefit of this approach is its lightness, which makes it easily applicable on mobile devices and on robotic platforms. A drawback of our approach is its lower accuracy compared to most of the top solutions, which are based on heavy-weight networks.

The work was experimentally validated on two image datasets. The newly proposed arrangement of joints has increased the classification accuracy by 1.8%, although an increase of parameter number by 26.5% was needed. This use of popular datasets allowed a performance comparison of our approach with other methods described in the literature. The performance of our light-weight solution is comparable with computationally more costly methods (−3.4% vs. more than 10 times lighter); it is of higher value than in relatively old heavy solutions (+12.4% and 9 times lighter) and is only overpowered by the recent top solutions (−6.8% vs. 10 times lighter).

The limitations of this study are threefold: a focus on the actions of main body parts, the use of a single performance measure and assuming a single action per video clip. As the feature vector is based on the subset of the 15 most reliably detected skeleton joints, human actions performed mainly by feet, hands and fingers, which are not included in this subset, cannot be properly distinguished from each other. The evaluation process of the proposed approach could include other popular measures, such as the precision-recall curve and AUC. Thus far, our implementation is processing well-prepared video clips with single actions per clip only.

Our future work should focus on more extensive training and testing of various network architectures (e.g., on the NTU-RGB+F 120 dataset) and on the design of more advanced feature engineering steps (e.g., adding motion information to joints). We are going to extend the action recognition problem to a two-person interaction classification. Apart from the testing on well-prepared video clips, our aim is to create a tool that can classify actions and interactions in longer videos, requiring a running window processing mode.

Author Contributions: Conceptualization, W.K. and B.J.; methodology, W.K. and B.J.; software, B.J.; validation, B.J., writing—initial editing, W.K. and B.J.; writing—review and editing, W.K.; visualization, W.K. and B.J.; project administration, W.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research and the APC was funded by "Narodowe Centrum Badań i Rozwoju" (Warszawa, Poland), grant No. CYBERSECIDENT/455132/III/NCBR/2020—APAKT.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The dataset can be downloaded after previous registration from the page: https://rose1.ntu.edu.sg/login/?next=/dataset/actionRecognition/request. For more information about the dataset go to: https://rose1.ntu.edu.sg/dataset/actionRecognition/.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Coppola, C.; Cosar, S.; Faria, D.R.; Bellotto, N. Automatic detection of human interactions from RGB-D data for social activity classification. In Proceedings of the 2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), Lisbon, Portugal, 28–31 August 2017.
2. Zhang, S.; Wei, Z.; Nie, J.; Huang, L.; Wang, S.; Li, Z. A review on human activity recognition using vision-based method. *J. Healthc. Eng.* **2017**, *2017*, 3090343. [CrossRef] [PubMed]
3. Hussain, Z.; Sheng, M.; Zhang, W.E. Different Approaches for Human Activity Recognition: A Survey. *J. Netw. Comput. Appl.* **2020**, *167*, 102738. [CrossRef]

4. Dalal, N.; Triggs, B. Histograms of Oriented Gradients for Human Detection. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–25 June 2005; Volume 1, pp. 886–893.
5. Liu, L.; Shahroudy, A.; Xu, D.; Wang, G. Spatio-Temporal LSTM with Trust Gates for 3D Human Action Recognition. In Proceedings of the Computer Vision—ECCV 2016, Amsterdam, The Netherlands, 11–14 October 2016; Springer International Publishing: Cham, Switzerland, 2016; pp. 816–833. [CrossRef]
6. Li, C.; Zhong, Q.; Xie, D.; Pu, S. Skeleton-based Action Recognition with Convolutional Neural Networks. *arXiv* **2017**, arXiv:1704.07595.
7. Bevilacqua, A.; MacDonald, K.; Rangarej, A.; Widjaya, V.; Caulfield, B.; Kechadi T. Human Activity Recognition with Convolutional Neural Networks. In *Machine Learning and Knowledge Discovery in Databases*; LNAI; Springer: Cham, Switzerland, 2019; Volume 11053, pp. 541–552.
8. Liang, D.; Fan, G.; Lin, G.; Chen, W.; Pan, X.; Zhu, H. Three-Stream Convolutional Neural Network with Multi-Task and Ensemble Learning for 3D Action Recognition. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Long Beach, CA, USA, 16–17 June 2019. [CrossRef]
9. Li, M.; Chen, S.; Chen, X.; Zhang, Y.; Wang, Y.; Tian, Q. Actional-Structural Graph Convolutional Networks for Skeleton-Based Action Recognition. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 3590–3598.
10. Liu, M.; Yuan, J. Recognizing Human Actions as the Evolution of Pose Estimation Maps. In Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, 18–22 June 2018; pp. 1159–1168.
11. Cippitelli, E.; Gambi, E.; Spinsante, S.; Florez-Revuelta, F. Evaluation of a skeleton-based method for human activity recognition on a large-scale RGB-D dataset. In Proceedings of the 2nd IET International Conference on Technologies for Active and Assisted Living (TechAAL 2016), London, UK, 24–25 October 2016.
12. Toshev, A.; Szegedy, C. DeepPose: Human Pose Estimation via Deep Neural Networks. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 1653–1660.
13. Insafutdinov, E.; Pishchulin, L.; Andres, B.; Andriluka, M.; Schiele, B. Deepercut: A deeper, stronger, and faster multi-person pose estimation model. In Proceedings of the Computer Vision—ECCV 2016, Amsterdam, The Netherlands, 11–14 October 2016; LNCS; Springer: Cham, Switzerland, 2016; Volume 9907, pp. 34–50.
14. Cao, Z.; Hidalgo, G.; Simon, T.; Wei, S.-E.; Sheikh, Y. OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *43*, 172–186. [CrossRef] [PubMed]
15. Wang, J.; Sun, K.; Cheng, T.; Jiang, B.; Deng, C.; Zhao, Y.; Liu, D.; Mu, Y.; Tan, M.; Wang, X.; et al. Deep High-Resolution Representation Learning for Visual Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *43*, 3349–3364. [CrossRef] [PubMed]
16. Felzenszwalb, P.F.; Girshick, R.B.; McAllester, D.; Ramanan, D. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *32*, 1627–1645. [CrossRef] [PubMed]
17. [Online]. NTU RGB+D 120 Dataset. Papers With Code. Available online: https://paperswithcode.com/dataset/ntu-rgb-d-120 (accessed on 30 June 2022).
18. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; p. 25.
19. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2015**, arXiv:1409.1556.
20. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA , 27–30 June 2016.
21. Munea, T.L.; Jembre, Y.Z.; Weldegebriel, H.T.; Chen, L.; Huang, C.; Yang, C. The Progress of Human Pose Estimation: A Survey and Taxonomy of Models Applied in 2D Human Pose Estimation. *IEEE Access*, **2020**, *8*, 133330–133348. [CrossRef]
22. Wei, K.; Zhao, X. Multiple-Branches Faster RCNN for Human Parts Detection and Pose Estimation. In Proceedings of the Computer Vision—ACCV 2016 Workshops, Taipei, Taiwan, 20–24 November 2017.
23. Su, Z.; Ye, M.; Zhang, G.; Dai, L.; Sheng, J. Cascade feature aggregation for human pose estimation. *arXiv* **2019**, arXiv:1902.07837.
24. Duan, H.; Zhao, Y.; Chen, K.; Shao, D.; Lin, D.; Dai, B. Revisiting Skeleton-based Action Recognition. *arXiv* **2021**, arXiv:2104.13586.
25. Shahroudy, A.; Liu, J.; Ng, T.-T.; Wang, G. NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis. *arXiv* **2016**, arXiv:1604.02808.
26. [Online] UTKinect-3D Database. Available online: http://cvrc.ece.utexas.edu/KinectDatasets/HOJ3D.html (accessed on 30 June 2022).
27. Plizzari, C.; Cannici, M.; Matteucci, M. Skeleton-based Action Recognition via Spatial and Temporal Transformer Networks. *Comput. Vis. Image Underst.* **2021**, *208–209*, 103219. [CrossRef]