Noname manuscript No.
(will be inserted by the editor)

LSTM-Based Real-Time Action Detection and Prediction in Human Motion Streams

Fabio Carrara · Petr Elias · Jan Sedmidubsky · Pavel Zezula

Received: date / Accepted: date

Abstract Motion capture data digitally represent human movements by sequences of 3D skeleton configurations. Such spatio-temporal data, often recorded in the stream-based nature, need to be efficiently processed to detect high-interest actions, for example, in human-computer interaction to understand hand gestures in real time. Alternatively, automatically annotated parts of a continuous stream can be persistently stored to become searchable, and thus reusable for future retrieval or pattern mining. In this paper, we focus on multi-label detection of user-specified actions in unsegmented sequences as well as continuous streams. In particular, we utilize the current advances in recurrent neural networks and adopt a unidirectional LSTM model to effectively encode the skeleton frames within the hidden network states. The model learns what subsequences of encoded frames belong to the specified action classes within the training phase. The learned representations of classes are then employed within the annotation phase to infer the probability that an incoming skeleton frame belongs to a given action class. The computed probabilities are finally compared against a learned threshold to automatically determine the beginnings and endings of actions. To further enhance the annotation accuracy, we utilize a bidirectional LSTM model to estimate class probabilities by considering not only the past frames but also the future ones. We extensively evaluate both the models on the three use cases of real-time stream annotation, offline annotation of long sequences, and early action detection and prediction. The experiments demonstrate that our models outperform the state of the art in effectiveness and are at least one order of magnitude more efficient, being able to annotate 10 k frames per second.

F. Carrara
ISTI-CNR, Pisa, Italy
E-mail: fabio.carrara@isti.cnr.it

P. Elias · J. Sedmidubsky · P. Zezula
Masaryk University, Brno, Czech Republic
E-mail: petr.eli.cz@gmail.com, xsedmid@fi.muni.cz, zezula@fi.muni.cz

1 Introduction

Motion capture data, or simply *motion data*, are 3D trajectories of human-skeleton joints recorded in a frame-by-frame manner. Current research mainly focuses on recognizing the classes of already segmented actions, based on the pre-classified training data [35,47,10,12,18,5,28,32,47]. Recognizing actions is quite difficult because the same actions can be performed by various subjects in a number of alternatives that vary in spatio-temporal configurations, i.e., speeds, timings or positions in space. To effectively learn the spatial and temporal motion characteristics, the state-of-the-art approaches often involve deep convolutional [23,35] and Long-Short Term Memory (LSTM) neural networks [28,47,32,36]. Compared to the traditional model-based approaches [42, 39,18] and classifiers [34,45,38], deep learning exhibits a great ability to recognize patterns in multimedia data [2,3].

All the action recognition approaches mentioned above, however, can classify only the actions that are already manually pre-segmented in advance. But motion data are often recorded continuously in form of unsegmented skeleton *sequences* or pseudo-infinite *streams*. Only a few approaches [30,41,46,11, 43,6,37,27] are able to detect actions within such unsegmented sequences or streams. The task of *action detection*, also referred to as stream or sequence *annotation*, constitutes a much more difficult problem than recognizing classes of short action clips because the beginnings and endings of actions are unknown and have to be determined precisely. It is even harder in time-critical applications (e.g., security) where high-interest actions (e.g., drawing a gun) are required to be detected immediately, within a maximum delay in order of tens to hundreds of milliseconds.

Based on the application scenario, the annotation task can be solved by *offline* or *online* algorithms. The online ones can process only the currently accessible part of a skeleton stream and have to make irrevocable decisions in real time, while the offline ones can process an input skeleton sequence as a whole and analyze not only preceding but also succeeding frames. In this paper, we focus on both scenarios of offline sequence annotation and online annotation of continuous streams. We also concentrate on action early-detection and prediction of actions shortly before they happen. For these scenarios, we perform a task of *supervised* and *multi-label* annotation, i.e., we expect that multiple actions can happen simultaneously and that some small amount of annotated sequences is available for the training purposes.

2 Related Work

While the action recognition paradigm is well-solved by many state-of-the-art classifiers [9], the offline and online annotation tasks still lack effective and efficient approaches. Effective unsupervised (category-blind) annotation approaches [13,21,44] exist, but they are hardly applicable for the real-time annotation as they need to process the whole motion sequence in advance to

discover and learn the discriminative repeating patterns first. On the other hand, supervised annotation approaches [41,46,11,43] can effectively process motion sequences in the stream-based nature to discover class-relevant frames or segments. In the following, we describe existing motion features and compare the traditional segment-based approaches with the frame-based variants of annotation algorithms.

2.1 Motion Features and Their Comparison

In the majority of related work as well as in this work, the spatio-temporal motion data is normalized [33] to become invariant towards the subject's position, orientation, and skeleton size. In particular, the skeleton in each frame is moved by pinning the root joint (see Figure 1) into the origin $(0, 0, 0)$ and rotated so that the skeleton hips are facing the fixed direction. The normalized data are further processed to extract descriptive frame-based [10,41,46, 47,30] or segment-based features [16,12,35]. Features can be compared for (dis)similarity by various distance measures, such as the Euclidean distance in [35], Hamming distance in [40], Dynamic Time Warping (DTW) in [5] or its variants in [30,13]. To avoid costly feature extraction and comparison, features can be embedded in the stateful automatons, graph-based models [41], or, similarly to our approach, extracted from hidden states of deep neural networks [37,27].

2.2 Segment-based Annotation

A continuous motion sequence or stream is gradually partitioned into short segments, typically using a no-prior knowledge partitioning [30,11,4]. The segments are identified using the traditional principle of sliding window that is shifted in a systematic and overlapping manner, which introduces a non-trivial replication of the input data. For instance, a multiple simultaneous sliding windows of different lengths in [11] cause a 20-fold increase in the data replication. To avoid a high segmentation overlap, a non-linear segmentation [6] is proposed to produce varying-length segments that carry a semantic power. The segments need to be processed to extract segment-level features that are used to retrieve the nearest matches within the features of the predefined class samples. If the similarity between a segment and its nearest match is high, the nearest-match class is considered as the segment label [11,30]. However, comparing the features of a large number of segments with a lot of provided action samples can be computationally demanding, especially when the costly DTW function of quadratic time complexity is employed [30]. Moreover, the labelled segments do not have to straightforwardly mark the precise beginnings and endings of actions. Another disadvantage is that each segment has to be first read from a stream before its processing begins, implying that annotations are discovered with a slight delay. Due to these disadvantages, the frame-level annotation based on deep recurrent learning is preferred in this work.

2.3 Frame-based Annotation

To avoid the disadvantages of the segment-based approach, per-class probabilities can be estimated for each frame to immediately infer the labels of classes, by exploiting the learned class representations. To enhance the annotation quality, the contextual information of so-far scanned frames is continuously encoded, for example, in the recurrent frame-based features [46], hidden states of auto-encoders [7], deep beliefs [41] or recurrent neural networks [17]. The LSTM recurrent neural networks [37,27] are very effective in preserving semantic context in the hidden states of the LSTM cells, compared for example to linear SVM classifiers that need to model the temporal context within the frame-level features [39]. In [37], two separate attention network modules are embedded, one spatial for discovering the most descriptive joints and one temporal for discovering the most important key poses. Similarly, in [27] a two-branch model is used, one branch is used for classification and one for estimating beginnings and endings of actions. In both cases, only one class can be annotated at the same time simultaneously. Also, the networks are very deep, containing multiple (3+) layers of LSTM cells and (4+) fully connected layers. We propose rather a light-weight architecture with only one LSTM cell for online annotation and two for offline annotation, which makes the training and annotation process more efficient.

2.4 Early Detection and Prediction

Learning-based frame-level approaches [37,27] and traditional segment-based retrieval approaches [11] are able to annotate motion sequences or streams in real time; meaning that the average frame annotation rate is higher than the actual frame rate. The major difference is that the frame-based annotation approaches [29,26,27] are additionally suitable for the task of early detection, i.e., discovering the beginnings of actions even before they finish. By increasing the accepting sensitivity, actions can be even predicted several frames ahead [7, 17], usually at the cost of lower recognition accuracy. Action prediction is important in time-critical applications, for example, in gesture recognition for game play [20].

2.5 Our Contributions

To significantly outperform the current algorithms for annotating skeleton sequences, we employ the LSTM-based recurrent neural networks that have already proved successful in recognizing short pre-segmented actions [28,32, 47]. In particular, we propose an online action detection algorithm (Online-LSTM) that is able to recognize precise beginnings and endings of concurrent actions within motion streams. We show that the beginnings of actions are detected immediately, without the necessity to wait for their termination and

that by increasing the sensitivity of accepting thresholds, actions can even be predicted a few hundreds milliseconds ahead. Additionally, we propose an offline algorithm (Offline-LSTM) that utilizes a bidirectional LSTM network to further enhance the annotation accuracy by analyzing future-to-past context. In contrast to standard algorithms, both approaches provide a multi-label annotation of actions that can be performed concurrently.

Both online and offline annotation scenarios are experimentally evaluated on real-life skeleton streams as well as unsegmented sequences on a standardized dataset from both effectiveness and efficiency points of view. We further analyze how the annotation quality is influenced by changing the frame-per-second rate. The achieved results are measured using well-established metrics (F-measure and Average Precision) and compared with the state-of-the-art approaches. Our approach achieves not only clearly higher effectiveness but shows a superior performance as it is at least one order of magnitude more efficient than other state-of-the-art annotation algorithms.

3 Methodology

We formally define motion capture data and the problem of their multi-label annotation. Then, we introduce two approaches (Online-LSTM and Offline-LSTM) based on recurrent neural networks to perform online annotation of streams and offline annotation of sequences.

3.1 Problem Definition

A *motion sequence* (or simply *motion*) is represented by a sequence $(P_1, \ldots, P_n)$ of consecutive skeleton *poses* $P_i\,(i \in [1, n])$. The total number n of skeleton poses determines the motion *length*. The i-th pose $P_i \in \mathbb{R}^{93}$, captured at the time moment i $(1 \leq i \leq n)$, consists of 3D coordinates of 31 tracked *joints*, as graphically illustrated in Figure 1.

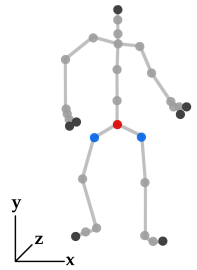

Fig. 1 Skeleton with 31 joints. The root is depicted by red color, hips by blue color.

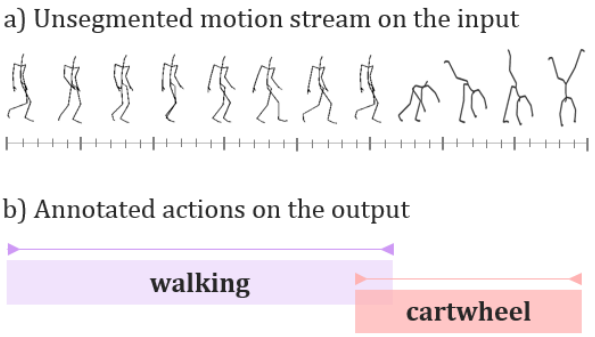

Fig. 2 Simple illustration of the multi-label annotation.

Table 1 Table of symbols.

Symbol	Description
P_i	i-th skeleton pose consisting of 3D joint coordinates
n	total number of sequence poses
C_j	j-th action class
m	total number of action classes
$y_{i,j}$	ground-truth label of i-th pose and j-th action class
$p_{i,j}$	probability that the i-th pose belongs to the j-th action class

A *stream* is a motion sequence that is pseudo-infinite (i.e., the length n is not bounded). In a given time moment, only a limited number of past frames can be accessed in main memory and if they are not processed or stored, they are lost. The difference between motion sequences and streams is that sequences can be processed offline and as a whole, while streams require real-time processing without any knowledge of the content coming in the future.

Simply illustrated in Figure 2, *annotation*, sometimes referred to as *action detection*, is the problem of determining what subsequences of a stream (resp. sequence) correspond to the predefined *classes* of actions. The predefined training set $\mathcal{C} = (C_1, \ldots, C_m)$ consists of m classes, where each class $C_j\,(j \in [1, m])$ is characterized by a non-empty set of *action samples*. The supervised multi-label annotation task is formally defined as follows: given the training set of classes $\mathcal{C} = (C_1, \ldots, C_m)$ and a sequence $(P_1, \ldots, P_n)$ of skeleton poses, determine for each pose the probabilities $p_{i,j} \in [0, 1]$ that the pose P_i belongs to the class C_j , $j \in [1, m]$. Consequently, the j-th class is assigned to the i-th pose P_i if $p_{i,j}$ is greater than some fixed [11] or variable [13] threshold. Since multiple classes of actions can happen simultaneously, the assigned annotations are not necessarily exclusive, i.e., multiple classes can be assigned to a single pose. Thus, an independent set of probabilities for each class is predicted, i.e., $\sum_{j=1}^{m} p_{i,j} \neq 1$. To train neural networks and verify annotation results, the ground-truth labels are needed. They are denoted as: $\mathbf{y} \in \{0, 1\}^{n \times m}$, where $y_{i,j} = 1$ if the i-th pose P_i belongs to class C_j; $y_{i,j} = 0$ otherwise.

Note that all the notations used throughout this paper are summarized in Table 1.

3.2 Online-LSTM: Stream Annotation

We model all the probabilities $p_{i,j}$ of each incoming stream pose P_i ($i \in \mathbb{N}$) to belong to each class C_j ($j \in [1, m]$) with an architecture based on recurrent neural networks (RNN). In particular, we adopt the Long Short-Term Memory (LSTM) [15] in the recurrent part of the network due to its effectiveness already proved in several sequence-modeling tasks [32, 28, 47]. LSTM is comprised of learnable gating functions specifically designed to better manage the internal

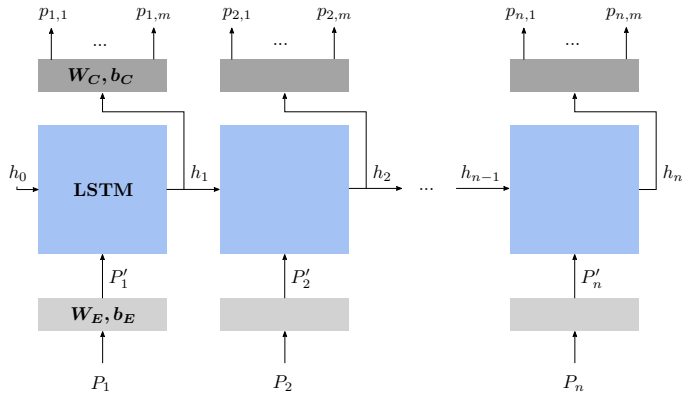

Fig. 3 Online-LSTM architecture.

memory state when coping with long sequences and mitigate the vanishing gradient problem in RNNs.

The recurrent cell, which is applied to each pose of the stream, is defined as follows. First, we embed each stream pose $P_i \in \mathbb{R}^{93}$ into a E-dimensional space with a linear projection (with parameters $W_E \in \mathbb{R}^{93\times E}$ and $b_E \in \mathbb{R}^E$) followed by a ReLU activation:

$$P'_i = \text{ReLU}(W_E \cdot P_i + b_E). \tag{1}$$

Learning a projection of the original data in the end-to-end training phase permits us to work with data of reduced dimensionality and a higher level of abstraction in the rest of the pipeline, with both effectiveness and efficiency advantages with respect to original motion data.

The obtained embedded poses $P'_i \in \mathbb{R}^E$ are then fed to a unidirectional LSTM cell, which produces a H-dimensional *hidden state* vector $h_i \in \mathbb{R}^H$:

$$h_i = \text{LSTM}(P'_i, h_{i-1}). \tag{2}$$

The initial state h_0 is set to zero. This state vector together with the consecutive pose P_{i+1} are given as input to the next step.

At each step, the hidden state h_i is used to efficiently encode the information about the history of the poses seen so far. The prediction for each class probability for the i-th pose P_i is obtained from the hidden state h_i as follows:

$$p_{i,j} = \sigma_j(W_\mathcal{C} \cdot h_i + b_\mathcal{C}) \qquad j \in [1, m], \tag{3}$$

where $W_\mathcal{C} \in \mathbb{R}^{H\times m}$ and $b_C \in \mathbb{R}^m$ are the parameters of a linear projection with m outputs, and $\sigma_j(\cdot)$ denotes the result of the sigmoid function applied to the j-th component of its argument. The whole architecture is denoted as Online-LSTM; it is graphically illustrated in Figure 3.

We optimize the parameters (W_E, b_E, $W_\mathcal{C}$, $b_\mathcal{C}$, and LSTM parameters) by minimizing the binary cross-entropy between the pose-level predictions and the targets. Let $y_{i,j}$ be the ground-truth annotation of pose P_i (i.e., $y_{i,j} = 1$ if P_i

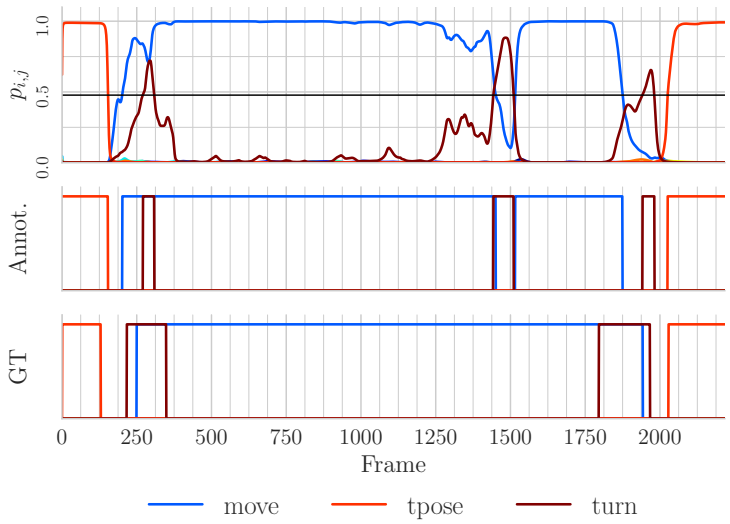

Fig. 4 The per-frame probability output $p_{i,j}$ for three different classes of "move" "pose" and "turn" of the trained Online-LSTM model annotating a motion sequence of approximately 2,100 frames. The horizontal black line in the first row indicates the value of accepting threshold. The resulting annotations ("Annot.") and the ground-truth annotations ("GT") are shown below.

belongs to an action of class C_j; $y_{i,j} = 0$ otherwise) and $p_{i,j}$ be the probability estimation of the network for pose P_i and class C_j, the loss function is then defined as follows:

$$L = -\sum_{i=1}^{n}\sum_{j=1}^{m} y_{i,j} \cdot \log(p_{i,j}) + (1 - y_{i,j}) \cdot \log(1 - p_{i,j}), \tag{4}$$

where n is the length of the training sequence, and m the number of classes to be recognized.

Once trained, in the test phase we obtain a hard annotation for each stream pose P_i and each class C_j by applying a threshold to all the $p_{i,j}$ predictions, as depicted in Figure 4. Moreover, we can define such a threshold independently for each class. In the experiments, we evaluate different values of thresholds and also introduce the way of their automatic tuning during the training phase.

This Online-LSTM architecture is particularly suited to be applied in online stream processing scenarios because the annotation of the incoming pose P_{i+1} is obtained from the newly arrived data and the current hidden state h_i only.

3.3 Offline-LSTM: Enhanced Sequence Annotation

Albeit the Online-LSTM model described in Section 3.2 is eligible for online annotation of a theoretically infinite stream, it is unable to exploit information

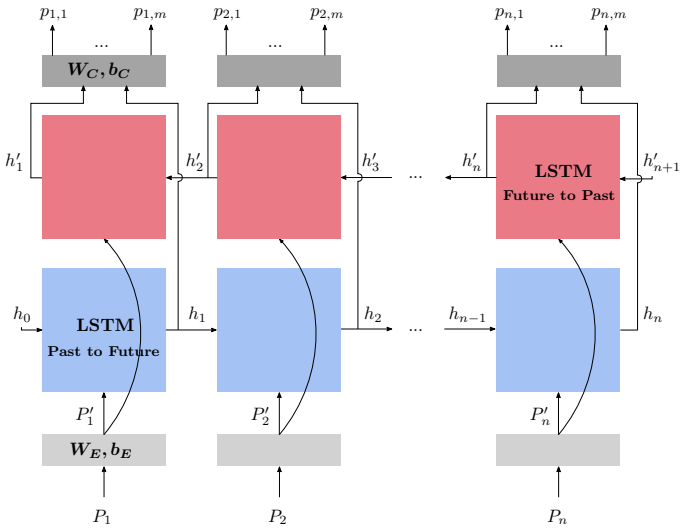

Fig. 5 Offline-LSTM architecture.

about the future poses to refine the prediction of the current pose. For scenarios where sequences of finite lengths can be annotated offline, we propose an augmentation of the model in which we integrate the information about the future in the current pose classification. We employ a bidirectional LSTM as the recurrent neural network, which adds a second LSTM cell to process the sequence in reverse, from the future to past. This cell is responsible to encode the information of the future poses. Formally, the output of this newly added LSTM is the following:

$$h'_i = \text{LSTM}(P'_i, h'_{i+1}), \tag{5}$$

where P'_i is the embedding of pose P_i (defined in the same way as in Equation 1) and h'_i is the hidden state. This hidden state is computed starting from the current pose P_i and the previous state (in the future) h'_{i+1}, thus encoding the "future history" of all the poses $P_i, P_{i+1}, \ldots, P_n$. The initial state h'_{n+1} is set to zero.

In order to obtain an enhanced classification of the pose P_i, we combine the past-to-future h_i and future-to-past h'_i states. Consequently, we obtain a prediction as in Equation 3 that now, however, depends on the entire sequence:

$$p_{i,j} = \sigma_j(W_\mathcal{C} \cdot [h_i|h'_i] + b_\mathcal{C}) \qquad j \in [1, m], \tag{6}$$

where $[h_i|h'_i]$ is the concatenation of both past-to-future h_i and future-to-past h'_i states, and the rest being the same as in Equation 3. This Offline-LSTM architecture is graphically illustrated in Figure 5. Since the Offline-LSTM model contains a second LSTM, we halve the dimensionality of hidden states for both LSTMs (i.e., $h_i, h'_i \in \mathbb{R}^{H/2}$) to have roughly the same model capacity as in the Online-LSTM model, described in Section 3.2. The Offline-LSTM model is optimized using the same loss function as defined in Equation 4.

4 Experimental Evaluation

The annotation quality of both the proposed Online-LSTM and Offline-LSTM models is experimentally analyzed in terms of effectiveness and efficiency. The models are also evaluated on three different application use cases: real-time stream annotation, offline annotation of sequences, and action early-detection and prediction. In addition, each use case is validated on three different subsets of the HDM05 dataset [31]. The motivation for using this dataset is that it provides overlapping ground truth, which is suitable for the evaluation of multi-label annotation task, and recognizes up to 130 classes, which is the highest number of classes with respect to existing datasets. The best-achieved results are compared with the state-of-the-art approaches evaluated on the same HDM05 dataset.

4.1 Dataset

The HDM05 dataset [31] contains 324 sequences performed by 5 different subjects. The total length of all sequences is about 3.5 hours, which corresponds to 1.5 M frames with a sampling frequency of 120 Hz. The dataset provides the following ground-truth sets that label subsequences of selected sequences by different motion classes.

- **HDM05-15**: $1,464$ actions in 15 non-uniformly populated classes, such as "rotate arms", "kick" or "exercise". The shortest action takes 0.34 s (41 frames) while the longest one has 17.2 s ($2,063$ frames). These actions are annotated within 102 motion sequences ($491,847$ frames in total $\sim$ **68** minutes).
- **HDM05-130**: $2,345$ actions in 130 classes, with the shortest and longest action of 0.1 s and 7.5 s (13 and 900 frames), respectively. The actions are labeled within 238 sequences ($1,125,652$ frames $\sim$ **156** minutes). This ground truth has the additional two following variants:
  - **HDM05-65**: A more granular categorization that divides all the $2,345$ actions into 65 more general classes, combining primarily repetitive movements into the same class, e.g., "clap 1 repetition" and "clap 5 repetitions". This categorization is also proposed in [10].
  - **HDM05-122**: The same ground-truth as the HDM05-130 without the 8 least-populated classes containing a very low number of samples ($2,328$ actions are categorized in 122 classes in total). This categorization also appears in [35].

The Online-LSTM and Offline-LSTM models are evaluated on the variants of HDM05-15, HDM05-65 and HDM05-122 ground-truth sets. For simplicity, we call individual ground-truth variants as *datasets*. For each variant, available motion sequences are split roughly into halves in order to apply the 2-fold cross-validation procedure. The sizes of splits are specified in Table 2 for the individual variants.

Dataset	Fold	Sequences	Frames	Minutes	Annotations
HDM05-15	#1	38	225,319	31.3 min	742
	#2	64	266,528	37.0 min	722
HDM05-65/122	#1	105	486,788	67.6 min	1,151
	#2	133	638,864	88.7 min	1,177

Table 2 Partitioning sequences of each dataset into two folds (the HDM05-65 and HDM05-122 datasets have the same folds since they are defined over the same sequences) to contain a similar number of annotated subsequences (the *Annotations* column). The column *Sequences* denotes the number of sequences contained in a specific fold, while *Frames* and *Minutes* columns state the total length of all sequences in the number of frames and minutes.

4.2 Experimental Protocol

In the first pass of the 2-fold cross-validation procedure, the first fold is used for training and the second one for testing the annotation. Within the training phase, the training sequences are used to train neural-network architectures of Online-LSTM and Offline-LSTM models, based on the description in Section 3.2 and Section 3.3. In the second pass, the training and test folds are swapped.

4.2.1 Training Details

We train our models using the Adam optimizer [19] with a learning rate of 0.0005. We update the parameters after every sequence is fed to the network (i.e., the batch size is 1). As regularization techniques, we apply an $L2$ weight decay of 0.0001, we clip the norm of the gradients to a maximum value of 10, and apply dropout with a keep probability of 0.5 after the embedding layer and before the classifier. We train both models for 200 epochs, selecting as the final model the one yielding the best micro-AP metric on the test fold. The dimensionality of the embeddings E and of the hidden state vector H are selected to be respectively 64 and $1,024$ for both models, as they achieve the most reasonable trade-off between accuracy and performance when detecting 122 action classes – we empirically observed that higher dimensionalities only provide negligible or no improvements.

4.2.2 Metrics

For each pass, efficiency is measured as the average time to annotate a single frame based on the total annotation time and the total number of frames. The accuracy is quantified by two standard metrics: *F-measure* (F_1) *score* and *Average Precision* (AP) score. Both these scores are based on the precision and recall metrics that are computed on the level of individual frames as:

– *Precision*: the ratio of correctly annotated frames and all the model-annotated frames on test sequences;

– *Recall*: the ratio of correctly annotated frames and all the ground-truth annotated frames on test sequences.

The values of the precision and recall are sensitive to the settings of an acceptance threshold on action-pose probability. Higher values increase the precision, but many annotations can be falsely rejected. On the other hand, selecting very low values causes that nearly all poses are labeled by all the classes. This is the reason why we select all possible settings of thresholds to report the trade-off between recall and precision. Considering $t \in \mathbb{N}$ different threshold settings, the F_1 and AP scores are defined as:

– F_1 *score*: the harmonic mean between precision ($Precision_i$) and recall ($Recall_i$) with respect to the i-th threshold value ($i \in [1, t]$) computed as: $F_1 = 2 \cdot (Precision_i \cdot Recall_i)/(Precision_i + Recall_i)$;
– AP *score*: the area under the precision-recall curve computed as a weighted mean of precisions at each unique threshold value i, where the increase in recall from the previous threshold is used as the weight:

$$AP = \sum_{i=1}^{t} (Recall_i - Recall_{i-1}) \cdot Precision_i. \tag{7}$$

While the F_1 score is traditionally reported for the best-performing threshold with the most optimistic values of precision and recall, the AP score is a more realistic metric expressing the annotation behavior disregarding one fixed, possibly extrapolated, threshold.

These scores are obtained for each sequence and for each class, and they can be averaged on either *micro* or *macro* level to obtain a unique global metric. The *micro*-averaging method computes the metric over all model-provided annotations with respect to the ground truth annotations globally, disregarding the cardinality of classes. Micro-F_1 and micro-AP metrics are computed using the micro-averaged precision (micro-P) and recall (micro-R) defined as:

$$\text{micro-}P = \frac{\sum_{j=1}^{m} \mathrm{TP}_j}{\sum_{j=1}^{m} \mathrm{TP}_j + \mathrm{FP}_j}\,, \quad \text{micro-}R = \frac{\sum_{j=1}^{m} \mathrm{TP}_j}{\sum_{j=1}^{m} \mathrm{TP}_j + \mathrm{FN}_j}\,, \tag{8}$$

where $\mathrm{TP}_j, \mathrm{FP}_j, \mathrm{FN}_j$ are respectively the number of True Positives, False Positives, and False Negatives of class j. Consequently, the micro-averaged metric can be strongly biased towards more frequent classes, while ignoring the less frequent ones. In such case, the *macro*-averaging method is useful as it computes the average from all the per-class metrics disregarding the individual class sizes:

$$\text{macro-}F_1 = \frac{1}{m} \sum_{j=1}^{m} F_{1,j}\,, \quad \text{macro-}AP = \frac{1}{m} \sum_{j=1}^{m} AP_j\,, \tag{9}$$

where $F_{1,j}$ and AP_j are respectively the F_1 and AP scores computed for the j-th class. The accuracy as well as the efficiency of a given annotation model are finally expressed as an average over both folds.

4.2.3 Thresholds on Action-Pose Probability

To determine the beginnings and endings of actions, an acceptance threshold on pose probability has to be specified. We define either a single *global* (GL) action-pose threshold same for all the classes, or multiple *class-based* (CB) thresholds computed for each class independently. The values of both variants of global and class-based thresholds can be either automatically derived from the training data only, or purposely set to best fit on the test data. These variants are further analyzed in more detail.

4.3 Thresholds and Annotation Accuracy

To express the trade-off between precision and recall, each experiment is evaluated using all possible threshold values (within interval $[0, 1]$) on action-pose probability. Figure 6 illustrates such trade-offs calculated for both Online-LSTM and Offline-LSTM using the global-threshold approach on the sequences of the HDM05-15 dataset. The computed precision-recall curves constitute an overall annotation accuracy – the larger under-curve area, the higher annotation quality. We quantify the areas by the AP score, which reaches high values of 88.22 % and 88.88 % for the Online-LSTM and Offline-LSTM, respectively.

In contrast to the AP score, the F_1 score depends on the specific threshold selection. We automatically derive the suitable threshold as the value that achieves the highest F_1 score on the training sequences. We denote this fair-fit threshold-selection scenario as FF. Alternatively, the best-fit threshold-selection scenario is computed as the threshold achieving the highest annotation accuracy on the test data and is denoted as BF. The best-fit threshold reflects the upper bound on the annotation accuracy and is commonly used in the state-of-the-art papers [11,30].

Table 3 presents the accuracy results of Online-LSTM for both scenarios of the fair-fit (FF) and best-fit (BF) threshold selection. Both these scenarios are further evaluated using global (GL) and class-based (CB) thresholds. For ex-

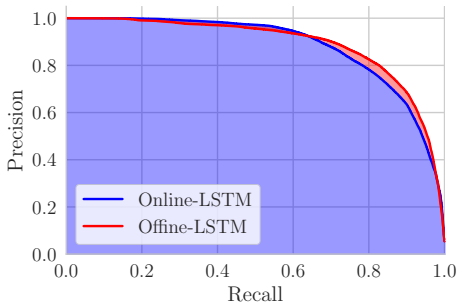

Fig. 6 Illustration of the precision-recall curves and AP scores ($AP = 88.22\,\%$ for Online-LSTM and $AP = 88.88\,\%$ for Offline-LSTM) computed using the global-threshold approach on sequences of the HDM05-15 dataset.

Metric	Thresh.	HDM05-15	HDM05-65	HDM05-122
AP	-	88.22±0.02%	64.00±2.45%	47.03±2.21%
F_1	FF-GL	79.17±0.04%	59.81±2.80%	47.66±1.73%
	FF-CB	79.10±0.41%	61.23±1.10%	46.17±2.53%
F_1	BF-GL	79.47±0.32%	60.86±2.06%	48.37±1.97%
	BF-CB	80.25±0.18%	63.80±0.86%	48.89±1.25%

Table 3 Annotation accuracy of the Online-LSTM model based on the global (GL) and class-based (CB) thresholds derived from either training (FF-), or test (BF-) data. The AP and F_1 metrics are computed on the micro-level approach.

Metric	Thresh.	HDM05-15	HDM05-65	HDM05-122
AP	-	88.88±0.06%	70.21±0.25%	60.59±0.69%
F_1	FF-GL	80.69±0.60%	64.82±0.28%	57.66±1.32%
	FF-CB	80.93±0.18%	68.01±0.24%	45.61±16.57%
F_1	BF-GL	81.08±0.26%	66.03±1.44%	58.83±1.02%
	BF-CB	82.11±0.21%	72.21±0.95%	65.18±0.97%

Table 4 Annotation accuracy of the Offline-LSTM model based on the global (GL) and class-based (CB) thresholds derived from either training (FF-), or test (BF-) data. The AP and F_1 metrics are computed on the micro-level approach.

ample, the FF-CB variant automatically learns the 15, 65 and 122 independent threshold values for the HDM05-15, -65 and -122 datasets, respectively. The difference between fair-fit and best-fit results are surprisingly low – around 1 percentage point on average across the datasets –, which demonstrates the ability of the approach to automatically derive appropriate thresholds from the training sequences only. The second observation is that the class-based thresholds in the best-fit scenario (BF-CB) obviously outperform the single global threshold (BF-GL). Unexpectedly, in the fair-fit scenario, the class-based thresholds (FF-CB) perform worse on the HDM05-122 dataset with respect to the global threshold (FF-GL). This ambiguity is caused by over-fitting of the fair-fit thresholds since the categories are very sparse and have uneven distribution of samples, which results in two different thresholds across the two folds. This phenomenon is even more noticeable in the FF-CB scenario of Offline-LSTM in Table 4, where a set of one-fold thresholds is much narrower than the other one. Consequently, very high standard deviations in FF-CB can be observed.

Otherwise, the trends of Offline-LSTM depicted in Table 4 are similar to the trends of Online-LSTM. As expected, Offline-LSTM outperforms the results of Online-LSTM on all the scenarios and datasets because it additionally leverages the time-reverse knowledge. The noticeable improvement of 16 percentage points (i.e., 33 %) is reached on the BF-CB variant applied to the most difficult HDM05-122 dataset.

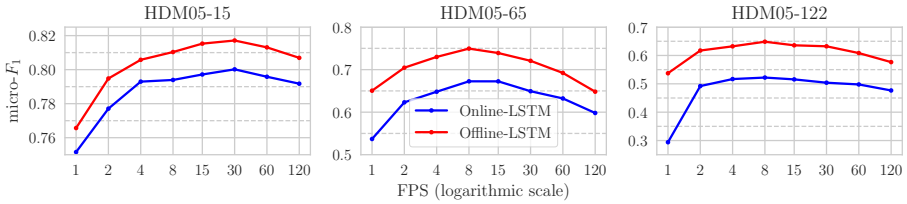

Fig. 7 Annotation accuracy of both Online-LSTM and Offline-LSTM evaluated using the FF-GL scenario on the sequences of the three datasets with a varying frame-per-second rate (on the x-axis of a logarithmic scale).

In the following experiments, we always consider the fair, train-data-derived, global-threshold scenario (FF-GL), due to its universality and stability in the achieved results across the folds and datasets.

4.4 Influence of the FPS-Rate

We analyze the robustness of the annotation models with respect to the input quality of motion data, in terms of different frame-per-second (fps) rates. We simulate a decreasing data quality by reducing the original 120-fps rate to the 60-, 30-, 15-, 8-, 4-, 2- and 1-fps rate. The lower-quality sequence of the i-fps rate is obtained by considering only each $(120/i)$-th frame of the original motion sequence. In this way, we modify the training and test sequences of both folds as well as the ground truth provided for all the three datasets.

Figure 7 depicts how the annotation accuracy of both the Online-LSTM and Offline-LSTM models changes with respect to the varying fps rate on a logarithmic scale. We can see that even a lower fps rate (around 8) is sufficient to achieve a very high accuracy. On the HDM05-65 and HDM05-122 datasets, the 8-fps rate even reaches the top accuracy. This is caused by the fact that the shorter sequences are simpler for recurrent network learning while the key movement characteristics are still preserved at this fps rate. Also, the beginning and ending frames of actions at lower fps rates become less ambiguous to be recognized. In the following sections, we always evaluate the experiments on the highest data quality, i.e., recorded with the 120 poses per second.

4.5 Use Case 1: Real-Time Stream Annotation

To simulate a stream, the sequences of each test fold (out of 2) are concatenated into one long sequence, separately for each of the three datasets HDM05-15, HDM05-65, and HDM05-122. The total lengths of concatenated sequences are in order of dozens of minutes – the exact times are specified in Table 2 for individual variants. To annotate such long sequences in the stream-like environment, we can only utilize the Online-LSTM model that continuously processes the sequence from the past to future. In particular, Online-LSTM

Metric	HDM05-15	HDM05-65	HDM05-122
micro-AP	86.46 ± 0.05%	38.73 ± 1.58%	26.87 ± 1.96%
macro-AP	79.49 ± 2.09%	46.78 ± 0.24%	28.75 ± 1.14%
micro-F_1	77.80 ± 0.30%	42.36 ± 0.79%	33.01 ± 1.49%
macro-F_1	73.13 ± 1.99%	31.92 ± 0.64%	19.73 ± 2.75%

Table 5 Annotation accuracy of Online-LSTM evaluated using the FF-GL scenario on simulated stream data, independently produced for each of the three datasets.

processes the stream in a frame-by-frame fashion and requires only to access the very last pose (i.e., the current frame). The most recent history is already embedded in the hidden state. This is a great advantage against common annotation approaches [30, 11] that usually require a much longer time window to analyze the past and current data at each time moment.

Table 5 depicts the results of stream-annotation effectiveness. For each dataset, the average AP and F_1 scores along with standard deviations over both folds are evaluated on the micro as well as macro (i.e., class-average) level. As expected the annotation accuracy decreases with an increasing number of classes to be recognized. In comparison with the results of AP and F_1 on the same FF-GL scenario in Table 3, the separate test sequences are more effectively annotated than the same sequences concatenated into one single stream. It is caused by the fact that artificially-linked stream data are much longer than the training sequences and also contain hard cuts between the concatenated sequences, on which the Online-LSTM model has not been trained.

The total annotation time proportionally depends on the actual length of the processed stream. Based on our measurements, a single pose is processed and annotated in 0.13 ms. For example, the 120-fps-rate stream data corresponding to the second-fold sequence of 37.0 minutes of the HDM05-15 dataset are processed in 35.3 seconds. This demonstrates that (potentially multiple) streams can clearly be annotated in real time. A more detailed evaluation of efficiency is discussed in Section 4.8.

4.6 Use Case 2: Offline Sequence Annotation

In scenarios where motion data do not need to be processed in real time, the Offline-LSTM model can be utilized to enhance the annotation effectiveness. This model can be applied to the stored test sequences that are processed from the past to future and simultaneously from the future to past. Table 6 demonstrates the results of Offline-LSTM on the FF-GL scenario and all the three datasets. Compared to the stream-annotation use case in Table 5, all the results are significantly better, mainly for the HDM05-65 and -122 datasets. We mainly highlight a more than 80 % F_1 accuracy on the HDM05-15 dataset. This approach might be suitable to significantly reduce human costs for tasks of automatic annotation of very large unannotated motion corpora, or segmen-

Dataset	HDM05-15	HDM05-65	HDM05-122
micro-AP	88.88±0.06%	70.21±0.25%	60.59±0.69%
macro-AP	84.39±1.68%	73.81±0.36%	63.72±2.54%
micro-F_1	80.69±0.60%	64.82±0.28%	57.66±1.32%
macro-F_1	75.39±1.61%	57.52±2.35%	44.12±0.18%

Table 6 Annotation accuracy of Offline-LSTM evaluated using the FF-GL scenario on the test sequences of the three datasets.

tation of very long and continuous motion sequences to become searchable and reusable.

4.7 Use Case 3: Early-Detection and Prediction of Actions

The majority of annotation approaches detect actions when they finish or even with a short delay. This delay is caused, for example, by using a segmentation policy that has to wait for the segment to be read as a whole before it can be processed [11], or by a costly post-processing that is required to enhance the detection of beginnings and endings of annotations [30]. Since the proposed Online-LSTM model enables real-time stream annotation on the level of individual frames, it can detect actions even before they finish, with high confidence and only with a minimal delay.

The delay of detected actions is evaluated on the same setup as the first use case, i.e., on the stream-based HDM05-15 dataset using the Online-LSTM model. The delay is measured as a difference between the beginning times of the action detected by Online-LSTM and that of the ground truth. A negative value means that the action has been predicted before it even started, while a positive one denotes how long it took to detect the action after its start. An average delay is computed as the mean of all the annotation delays. Since the average delay depends on the selection of an acceptance threshold, the experiments are conducted for 500 threshold values evenly spaced within the $(0, 1)$ interval.

To fairly evaluate action-detection delays, we focus only on "correctly found annotations" that have a high overlap with the ground-truth ones. The size of overlap is quantified by IoU – the intersection of frames over the union of frames, between the found and ground-truth annotations. The scatter plot in Figure 8 shows that the found annotations that highly overlap with the ground-truth ones have more stable delay values, whereas the delay of less accurate annotations becomes quite spread. For this reason, we measure the delay and accuracy only for those annotations that are closely related, i.e., have $IoU \geq 0.5$.

Figure 9 depicts how the average delay relates to the annotation quality expressed by the F_1 score – the experiment is computed for the annotations having $IoU \geq 0.5$ for all the tested threshold values between $(0, 1)$. As expected, higher thresholds give higher F_1 scores at the expense of some delay

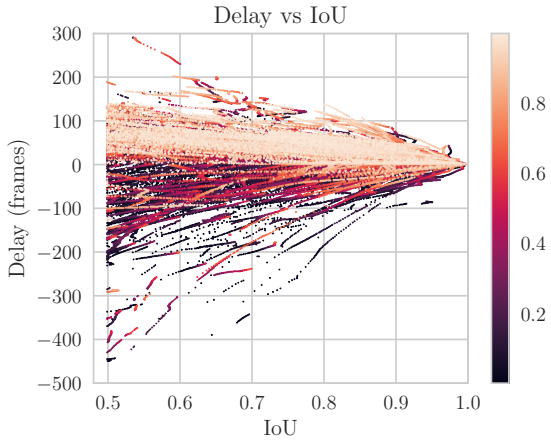

Fig. 8 A scatter plot of Online-LSTM-based annotations that are correctly related to the HDM05-15 ground truth with the corresponding IoU (intersection of frames over the union of frames). The 500 different thresholds between $(0, 1)$ for which the experiment is conducted are color-coded on the right axis.

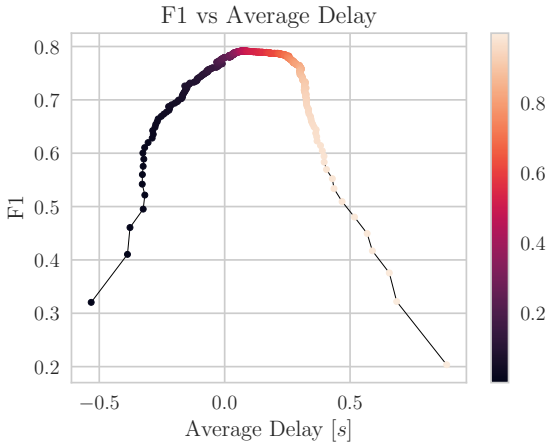

Fig. 9 Average delays and F_1 scores obtained by Online-LSTM model on the HDM05-15 dataset for 500 different thresholds between $(0, 1)$ color-coded on the right axis.

in the prediction. The highest F_1 score is achieved with an average delay of approximately 100 ms, which is slightly faster than average human reaction time (0.25–0.35 seconds). Focusing on negative average delays, the actions can be predicted 0.3 seconds before they happen with an F_1 score of 0.5. Such action prediction is especially useful in time-critical applications, for example, in security or human-computer interaction, when actions are required to be automatically detected at their early beginnings.

4.8 Efficiency Evaluation

Efficiency is quantified in seconds and measured independently for the training and annotation processes on a Linux machine with an i7-4790 3.60 GHz CPU and one NVIDIA Tesla K40 card. We use PyTorch to implement and train our models with GPU acceleration. The training process is quite efficient and does not require partitioning the sequence data into segments and extracting their features, as used in related papers [11]. The whole training takes roughly 5 and 3.5 hours to learn the Online-LSTM and Offline-LSTM model, respectively, on the first HDM05-15 fold containing 38 sequences of the total length of 31.3 minutes.

The second HDM05-15 fold of 64 sequences (37 minutes in total) is very efficiently annotated in 35.3 and 20.6 seconds by the Online-LSTM and Offline-LSTM model. This means that a single pose is processed and annotated in **0.13 ms** by Online-LSTM and in **0.08 ms** by Offline-LSTM. Note that Offline-LSTM is faster than Online-LSTM since matrix multiplications of two LSTMs with hidden-state size of 512 (utilized by Offline-LSTM) are computationally less expensive than a single LSTM with hidden-state size of $1,024$ (employed by Online-LSTM).

The measured annotation times have the same trend also on the other folds of the HDM05-65 and HDM05-122 datasets, as the performance is proportionally dependent to the sequence length. This implies that annotation time is about two orders of magnitude faster than the actual duration of a sequence, even sampled with the 120-fps rate. With lower fps rates, the sequence is annotated even much faster. For example, considering the high-accurate 8-fps rate, the 37-minute sequence is annotated in 1.5 seconds by the Offline-LSTM model. At this reduced fps rate, as many as $1,500$ streams could be possibly processed in real time at once.

4.9 Comparison with the State of the Art

We quantitatively compare effectiveness of our approach on the HDM05-15 dataset against two multi-label annotation approaches [11,30], and qualitatively to the methods [27,37,6,46,41] that closely relate to our work. However, these approaches are evaluated on different datasets that do not provide the multi-label (i.e., overlapping) ground-truth. Finally, a comparison of annotation efficiency is provided in terms of annotation frame-per-second rate.

Effectiveness Quantitative Comparison. Evaluated on the same train/test data and frame-level metric, our approach significantly outperforms approaches [30, 11], as demonstrated in Table 7. In particular, the retrieval-based approach of Elias et al. [11] replicates the test sequences into multitude of short overlapping segments organized in a hierarchical way and matches them with the training action samples using k-NN search. The best reported *micro*-F_1 score of 68.65 % is achieved. To avoid repetitive segment to action matching, each

	Train data [min]	Test data [min]	Train time [h]	Per-frame processing time F. ext. [ms]	Annot. [ms]	Total [ms]	micro-F_1 [%]
Müller (online) [30]	24	60	N/A	1.88	2.29	4.17	61.00
Müller (offline) [30]	24	60	N/A	1.88	0.15	2.03	75.00
Elias (online) [11]	17	51	2.0	7.13	0.48	7.61	68.65
Online-LSTM BF-GL	17	51	5.0	-	0.13	0.13	74.95
Offline-LSTM BF-GL	17	51	3.5	-	0.08	0.08	78.78

Table 7 Comparison with state-of-the art approaches on the HDM05-15 dataset.

action class is summarized by compact templates in [30]. These templates are matched with data segments by the online annotation algorithm with the 61 % F_1 score. This result is significantly improved to 75 % by the offline genetic algorithm that extracts characteristic motion key-poses from the whole data sequence.

Effectiveness Qualitative Comparison. The most relevant approaches are the LSTM recurrent networks [27,37] that achieve superior annotation results on datasets captured by Kinect (OAD, G3D, PKU-MMD) that are challenging due to their low frame-rate and high tracking errors. Both approaches utilize very deep multi-layered architecture. In [27] two modules are used that require multi-stage training – one for classification and one for detection of beginnings and endings. Two attention modules are used also in [37]. We rather propose a fast and light-weight solution that models classification and annotation within one single LSTM cell simultaneously. However, the key difference is that we provide multi-label output, whereas in [27,37] only a single action class can be annotated at one time. Model-based approaches [41,46,6] reach very competitive results on the MSRC-12 dataset [14], however, the reported F_1 scores are measured on the benevolent action level and not on the strict frame level.

Efficiency Comparison. We compare efficiency only among the online action detection algorithms [27,37,6,46,41,30,11] since the requirement on timeliness is more important than in offline processing. The comparison of annotation efficiency is shown in Table 8 in terms of the annotation frame-per-second (fps) rate, denoting the number of frames that could be possibly annotated within 1 second in real time. We can observe that all the compared methods provide a real-time annotation for data of fps rate of 130 or lower. Our approach demonstrates superior performance that is one order of magnitude more efficient that related approaches, having the ability to annotate 40 streams simultaneously at standard 30 fps rate in real time. This is primarily due to the light-weight architecture of the proposed solution. Table 8 also summarizes the ability of approaches to provide early detection (i.e., annotation of actions before they finish) and prediction (i.e., annotation of actions before they start).

Approach	Prediction	Early detection	Annot. fps rate
JCR-RNN [27]	✓	✓	1,230
STA-LSTM [37]	✓	✓	N/A
CuDi3D [6]	✓	✓	670
SSS Squared+L21 [46]	N/A	✓	500
DBN+HMM [41]	×	∼ 1 s delay	N/A
Müller et al. [30]	×	×	240
Elias et al. [11]	×	×	130
Online-LSTM (this work)	✓	✓	7,700

Table 8 Comparison with the state-of-the art real-time annotation approaches.

5 Conclusions

Inspired by the recent advances in recurrent neural networks, we successfully apply both unidirectional and bidirectional LSTM architectures to a demanding task of multi-label action detection in unsegmented skeleton sequences[1]. In particular, we introduce the Online-LSTM model suitable for time-critical action early detection in skeleton streams, and the Offline-LSTM one that further increases the accuracy when annotating long sequences offline. Both approaches establish a new annotation baseline in accuracy on the HDM05-65 and HDM05-122 datasets – still being the largest publicly available dataset in the number of classes – and additionally outperform the state-of-the-art result on the HDM05-15 dataset. The accuracy is measured not only for the best-performing threshold selection (F_1 score) as presented in the majority of related work but also by the threshold-independent average precision (AP) that better reflects the behavior between the precision and recall over all the possible threshold settings. The ability to detect actions with a minimum or even no delay is demonstrated only with a slight decrease in precision. Moreover, the ability to predict actions few hundreds milliseconds before they happen is observed for annotations having a high IoU with the ground truth. A major improvement in efficiency is demonstrated as it takes only 0.13 ms (Online-LSTM) and 0.08 ms (Offline-LSTM) to annotate a single frame, which is much more than one order of magnitude faster than the state-of-the-art action detection approaches.

In the future, we plan to combine both the Online-LSTM and Offline-LSTM models together to enhance the accuracy of online stream annotation at the cost of very short delays. Also we want to extend the idea to other related areas, such as video processing [24,25] and motion understanding from streams of 2D skeleton poses [1,8]. Understanding entitativity and social interactions in groups [22] are another promising future research directions.

[1] The code to reproduce the experiments is publicly available at `https://github.com/fabiocarrara/mocap`

Acknowledgements This research was supported by Smart News, "Social sensing for breaking news", CUP CIPE D58C15000270008, by Automatic Data and documents Analysis to enhance human-based processes (ADA), CUP CIPE D55F17000290009, and by ERDF "CyberSecurity, CyberCrime and Critical Information Infrastructures Center of Excellence" (No. CZ.02.1.01/0.0/0.0/16_019/0000822). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for this research.

References

1. Aberman, K., Wu, R., Lischinski, D., Chen, B., Cohen-Or, D.: Learning character-agnostic motion for motion retargeting in 2d. ACM Trans. Graph **38**(4) (2019)
2. Asadi-Aghbolaghi, M., Claps, A., Bellantonio, M., Escalante, H.J., Ponce-Lpez, V., Bar, X., Guyon, I., Kasaei, S., Escalera, S.: A survey on deep learning based approaches for action and gesture recognition in image sequences. In: 2017 12th IEEE International Conference on Automatic Face Gesture Recognition (FG 2017), pp. 476–483 (2017)
3. Baltruaitis, T., Ahuja, C., Morency, L.: Multimodal machine learning: A survey and taxonomy. IEEE Trans. on Pattern Analysis and M. Intelligence **41**(2), 423–443 (2019)
4. Barbič, J., Safonova, A., Pan, J.Y., Faloutsos, C., Hodgins, J.K., Pollard, N.S.: Segmenting motion capture data into distinct behaviors. In: Proceedings of Graphics Interface 2004, pp. 185–194. Canadian Human-Computer Communications Society (2004)
5. Barnachon, M., Bouakaz, S., Boufama, B., Guillou, E.: Ongoing human action recognition with motion capture. Pattern Recognition **47**(1), 238–247 (2014)
6. Boulahia, S.Y., Anquetil, E., Multon, F., Kulpa, R.: Cudi3d: Curvilinear displacement based approach for online 3d action detection. Computer Vision and Image Understanding (2018)
7. Butepage, J., Black, M.J., Kragic, D., Kjellstrom, H.: Deep representation learning for human motion prediction and classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6158–6166 (2017)
8. Cao, Z., Simon, T., Wei, S., Sheikh, Y.: Realtime multi-person 2d pose estimation using part affinity fields. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1302–1310 (2017)
9. Chen, C., Jafari, R., Kehtarnavaz, N.: A survey of depth and inertial sensor fusion for human action recognition. Multimedia Tools and Applications **76**(3), 4405–4425 (2017)
10. Du, Y., Wang, W., Wang, L.: Hierarchical recurrent neural network for skeleton based action recognition. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1110–1118 (2015)
11. Elias, P., Sedmidubsky, J., Zezula, P.: A real-time annotation of motion data streams. In: 19th Int. Symposium on Multimedia, pp. 154–161. IEEE Computer Society (2017)
12. Evangelidis, G., Singh, G., Horaud, R.: Skeletal quads: Human action recognition using joint quadruples. In: 22nd Int. Conference on Pattern Recognition (ICPR 2014), pp. 4513–4518 (2014)
13. Field, M., Stirling, D., Pan, Z., Ros, M., Naghdy, F.: Recognizing human motions through mixture modeling of inertial data. Pattern Recognition **48**(8), 2394–2406 (2015)
14. Fothergill, S., Mentis, H., Kohli, P., Nowozin, S.: Instructing people for training gestural interactive systems. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '12, pp. 1737–1746. ACM, New York, NY, USA (2012)
15. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
16. Hussein, M.E., Torki, M., Gowayyed, M.a., El-Saban, M.: Human action recognition using a temporal hierarchy of covariance descriptors on 3D joint locations. Joint Conference on Artificial Intelligence (IJCAI 2013) pp. 2466–2472 (2013)
17. Jain, A., Zamir, A.R., Savarese, S., Saxena, A.: Structural-rnn: Deep learning on spatio-temporal graphs. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 5308–5317 (2016)
18. Kadu, H., Kuo, C.C.J.: Automatic human mocap data classification. IEEE Transactions on Multimedia **16**(8), 2191–2202 (2014)

19. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
20. Kratz, L., Smith, M., Lee, F.: Wiizards: 3d gesture recognition for game play input. Proceedings of the 2007 Conference on Future Play, Future Play '07 pp. 209–212 (2007)
21. Krüger, B., Vögele, A., Willig, T., Yao, A., Klein, R., Weber, A.: Efficient unsupervised temporal segmentation of motion data. IEEE Transactions on Multimedia **19**(4), 797–812 (2017)
22. Lakens, D.: Movement synchrony and perceived entitativity. Journal of Experimental Social Psychology **46**(5), 701 – 708 (2010)
23. Laraba, S., Brahimi, M., Tilmanne, J., Dutoit, T.: 3d skeleton-based action recognition by representing motion capture sequences as 2d-rgb images. Computer Animation and Virtual Worlds **28**(3-4) (2017)
24. Li, K., He, F.Z., Yu, H.P.: Robust visual tracking based on convolutional features with illumination and occlusion handing. Journal of Computer Science and Technology **33**(1), 223–236 (2018)
25. Li, K., He, F.z., Yu, H.p., Chen, X.: A correlative classifiers approach based on particle filter and sample set for tracking occluded target. Applied Mathematics-A Journal of Chinese Universities **32**(3), 294–312 (2017)
26. Li, S., Li, K., Fu, Y.: Early recognition of 3d human actions. ACM Trans. Multimedia Comput. Commun. Appl. **14**(1s), 20:1–20:21 (2018)
27. Li, Y., Lan, C., Xing, J., Zeng, W., Yuan, C., Liu, J.: Online human action detection using joint classification-regression recurrent neural networks. In: B. Leibe, J. Matas, N. Sebe, M. Welling (eds.) Computer Vision – ECCV 2016, pp. 203–220. Springer International Publishing, Cham (2016)
28. Liu, J., Wang, G., Duan, L., Hu, P., Kot, A.C.: Skeleton based human action recognition with global context-aware attention LSTM networks. IEEE Transactions on Image Processing **27**(4), 1586–1599 (2018)
29. Ma, S., Sigal, L., Sclaroff, S.: Learning activity progression in lstms for activity detection and early detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1942–1950 (2016)
30. Müller, M., Baak, A., Seidel, H.P.: Efficient and Robust Annotation of Motion Capture Data. In: ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 2009), pp. 17–26. ACM Press (2009)
31. Müller, M., Röder, T., Clausen, M., Eberhardt, B., Krüger, B., Weber, A.: Documentation Mocap Database HDM05. Tech. Rep. CG-2007-2, Universität Bonn (2007)
32. Nunez, J.C., Cabido, R., Pantrigo, J.J., Montemayor, A.S., Velez, J.F.: Convolutional neural networks and long short-term memory for skeleton-based human activity and hand gesture recognition. Pattern Recognition **76**, 80–94 (2018)
33. Poppe, R., Van Der Zee, S., Heylen, D.K.J., Taylor, P.J.: Amab: Automated measurement and analysis of body motion. Behavior Research Methods **46**(3), 625–633 (2014)
34. Raptis, M., Kirovski, D., Hoppe, H.: Real-time classification of dance gestures from skeleton animation. In: ACM SIGGRAPH Eurographics Symposium on Computer Animation (SCA 2011), SCA 2011, pp. 147–156. ACM (2011)
35. Sedmidubsky, J., Elias, P., Zezula, P.: Effective and efficient similarity searching in motion capture data. Multimedia Tools and Applications **77**(10), 12,073–12,094 (2018)
36. Singh, D., Merdivan, E., Psychoula, I., Kropf, J., Hanke, S., Geist, M., Holzinger, A.: Human activity recognition using recurrent neural networks. In: A. Holzinger, P. Kieseberg, A.M. Tjoa, E. Weippl (eds.) Machine Learning and Knowledge Extraction, pp. 267–274. Springer International Publishing, Cham (2017)
37. Song, S., Lan, C., Xing, J., Zeng, W., Liu, J.: Spatio-temporal attention-based lstm networks for 3d action recognition and detection. IEEE Transactions on Image Processing **27**(7), 3459–3471 (2018)
38. Vieira, A., Lewiner, T., Schwartz, W., Campos, M.: Distance matrices as invariant features for classifying mocap data. In: 21st Int. Conference on Pattern Recognition (ICPR 2012), pp. 2934–2937 (2012)
39. Wang, C., Wang, Y., Yuille, A.L.: An approach to pose-based action recognition. In: Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition, CVPR '13, pp. 915–922. IEEE Computer Society (2013)

40. Wang, Y., Neff, M.: Deep signatures for indexing and retrieval in large motion databases. In: 8th ACM SIGGRAPH Conference on Motion in Games, pp. 37–45. ACM (2015)
41. Wu, D., Shao, L.: Leveraging hierarchical parametric networks for skeletal joints based action segmentation and recognition. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 724–731 (2014)
42. Xia, L., Chen, C.C., Aggarwal, J.K.: View invariant human action recognition using histograms of 3D joints. CVPR Workshops pp. 20–27 (2012)
43. Xu, Y., Shen, Z., Zhang, X., Gao, Y., Deng, S., Wang, Y., Fan, Y., Chang, E.C.: Learning multi-level features for sensor-based human action recognition. Pervasive and Mobile Computing **40**, 324–338 (2017)
44. Yu, X., Liu, W., Xing, W.: Behavioral segmentation for human motion capture data based on graph cut method. Journal of Visual Lang. & Computing **43**, 50–59 (2017)
45. Zanfir, M., Leordeanu, M., Sminchisescu, C.: The moving pose: An efficient 3d kinematics descriptor for low-latency action recognition and detection. In: International Conference on Computer Vision (ICCV 2013), pp. 2752–2759 (2013)
46. Zhao, X., Li, X., Pang, C., Sheng, Q.Z., Wang, S., Ye, M.: Structured streaming skeleton – a new feature for online human gesture recognition. ACM Trans. Multimedia Comput. Commun. Appl. **11**(1s), 22:1–22:18 (2014)
47. Zhu, W., Lan, C., Xing, J., Zeng, W., Li, Y., Shen, L., Xie, X.: Co-occurrence feature learning for skeleton based action recognition using regularized deep LSTM networks. In: 30th AAAI Conference on Artificial Intelligence, AAAI 2016, pp. 3697–3703. AAAI Press (2016)