SegmentCodeList: Unsupervised Representation Learning for Human Skeleton Data Retrieval

Anonymous Authors

Anonymous Institutions
Anonymous@emails

Abstract. Recent progress in pose-estimation methods enables the extraction of sufficiently-precise 3D human skeleton data from ordinary videos, which offers great opportunities for a wide range of applications. However, such spatio-temporal data are typically extracted in the form of a continuous skeleton sequence without any information about semantic segmentation or annotation. To make the extracted data reusable for further processing, there is a need to access them based on their content. In this paper, we introduce a universal retrieval approach that compares any two skeleton sequences based on temporal order and similarities of their underlying segments. The similarity of segments is determined by their content-preserving low-dimensional code representation that is learned using the Variational AutoEncoder principle in an unsupervised way. The quality of the proposed representation is validated in retrieval and classification scenarios by outperforming the effectiveness of state-of-the-art approaches.

Keywords: 3D skeleton sequence · segment · unsupervised feature learning · VAE · segment code list · action retrieval

1 Introduction

The rapid development of pose-estimation methods [4] enables more and more precise detection of human body keypoints (2D or even 3D) in individual frames of a standard video-camera recording. The detected keypoints are then used to simplify human motion using the spatio-temporal *skeleton sequence* representation. Since such skeleton sequences can nowadays be extracted from a common video, the analysis of human motion is becoming very popular in a broad spectrum of application domains, ranging from computer animation through robotics, security, autonomous driving, to healthcare and sports.

The skeleton sequences may generally appear in different forms – short or long, segmented or continuous, labeled or unlabeled. Current research in skeleton-data processing mainly focuses on recognizing classes of short and labeled *actions* [3,20,17] or detecting such actions in *continuous sequences* [19,24]. These tasks require examples of actions to be defined in advance, so that action classifiers or detectors can be trained in a supervised way. However, supervised training is not applicable to environments where examples of actions are not

known in advance. In environments where skeleton data are extracted as long continuous sequences without any information about their annotation or semantic partitioning, unsupervised content-based processing methods are the only possibility to make the recorded data searchable and thus reusable. One of the most general principles is to partition continuous sequences into short *segments* and access the data based on similarities of the underlying segments.

In this paper, we adopt such general segment-based processing principle by partitioning a continuous sequence into fixed-size segments and extracting the content-preserving segment representation – in the form of a low-dimensional *code* – in an unsupervised way. The most desirable property is that two codes are similar in terms of the cosine distance if their corresponding segments exhibit similar movement characteristics and vice-versa. In this way, we can represent a skeleton sequence of any length by the list of codes and determine the similarity of any two code lists based on the time-warping principle. This allows the proposed approach to be integrated within any retrieval-based operation.

2 Related Work and Our Contributions

Related approaches almost exclusively learn a skeleton-data representation on the level of *pre-segmented* actions, that are commonly provided by benchmark datasets [14,15,18]. Since the individual actions constitute standalone semantic units, the action representation can be straightforwardly learned in a *supervised* or *self-supervised* way. However, representation learning is not an easy task for the continuous (unsegmented) sequences whose content is generally unpredictable.

Approaches for Pre-segmented Actions Plenty of papers propose various architectures of *supervised* neural-network classifiers that trade between classification accuracy and the number of network parameters, pretty much influencing the training time. Such approaches are usually based on transformers [3], convolutional [17], recurrent [24], graph-convolutional [20] networks, or their combinations including attention-based mechanisms. However, they are limited to scenarios where both segmented actions and their labeling are provided in advance. Recently, *self-supervised* learning, where action labeling is not known, has become increasingly popular. In such cases, the action representation can be learned using reconstruction-based or contrastive-learning-based methods. The former group applies the encoder-decoder principle to reconstruct the original skeleton data of an action and uses the learned intermediate representation as the action feature. The latter group [26,12,10] aims at learning a meaningful metric that sufficiently reflects semantic similarity to discriminate actions belonging to different classes in the validation step. Still, all these methods are applied to scenarios where the actions are pre-segmented (known) in advance.

Approaches for Unsegmented Sequences Compared to the previous research, a very limited number of approaches provide content-based access to unsegmented skeleton data. In [23], the continuous sequences are synthetically partitioned into many overlapping and variable-size segments that are represented

by 4,096D deep features. However, such features are learned in a supervised way by exploiting supplementary knowledge about labeled actions, and indexing is very difficult due to both high feature dimensionality and a large number of segments. To move towards more efficient processing, the approaches in [1,16] quantize the segment features using k-means clustering. However, with such simple quantization, the border problem appears frequently, which decreases the effectiveness of applications with an increasing number of clusters (i.e., the vocabulary size). This problem is partly solved in [22] by applying soft quantization; however, limited effectiveness is achieved as the quantization process employs a numerical distance function that principally can not learn any semantics.

Contributions of This Paper

We propose an effective representation of unsegmented and unlabeled skeleton sequences using a list of compact codes learned in an unsupervised way. Compared to existing methods, the proposed codes are very compact (in contrast to high-dimensional features in [1]) and preserve motion semantics (in contrast to hand-crafted segment features in [22,16]). Specifically,

- we propose a lightweight residual neural-network architecture to effectively process short segments of spatio-temporal skeleton data;
- we apply the reconstruction-based Variational AutoEncoder approach in combination with the proposed architecture to learn semantic information of segment data in the form of a compact code;
- we propose to adopt the time-warping principle to determine a similarity between two code lists representing pairs of motions of any lengths;
- we verify the effectiveness and efficiency of the proposed code lists in the context of two retrieval-based application scenarios.

3 Code List Representation

In this section, we describe the whole process of transformation of a continuous skeleton sequence into a list of codes. In particular, we formally define the skeleton data domain together with the retrieval-based principle using k-nearest neighbor queries. Then, we present how continuous sequences are partitioned and how semantic codes are learned from such unlabeled segment data. Finally, we propose how to compare any two sequences represented by the lists of codes.

3.1 Problem Definition

We represent skeleton data as a continuous sequence $(P_1, \ldots, P_n)$ of n consecutive 3D *poses* P_i, where the i-th pose $P_i \in \mathbb{R}^{j \cdot 3}$ is captured at time moment i $(1 \leq i \leq n)$ and consists of xyz-coordinates of j tracked *joints*. In this paper, we use two different body models with $j = 31$ joints for the HDM05 dataset [18] and $j = 25$ joints for the PKU-MMD dataset [13]. The sequence of n poses

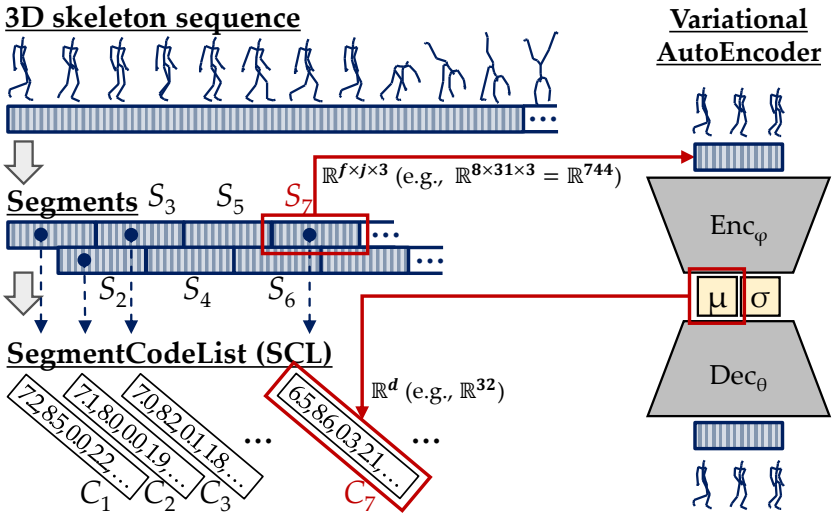

Fig. 1: A schematic illustration of the proposed transformation process of a continuous 3D skeleton sequence into the SCL representation.

is then partitioned into a list of segments $(S_1, \ldots, S_m)$, where $m \ll n$. Each segment $S_i = (P_1, \ldots, P_f)$ consists of a fixed number of f poses and is further transformed using the $codeTrans(S_i)$ function into a low-dimensional *code* representation $C_i \in \mathbb{R}^d$. The appropriate code dimensionality d typically ranges between $d \in 2^{[3..6]}$. Thus, the original high-dimensional skeleton-data sequence $(P_1, \ldots, P_n)$ is transformed into a short *code list* $(C_1, \ldots, C_m)$, so-called *SegmentCodeList* (SCL), consisting of m low-dimensional codes (e.g., on the HDM05 dataset, a 744-dimensional segment – consisting of eight 93D poses – is transformed into a 32D code). The whole transformation process is schematically illustrated in Fig. 1.

We evaluate the SCL representation on classification and retrieval scenarios using the k-nearest neighbor approach. Having a set $\{D_1, D_2, \ldots\}$ of *database sequences* $D_i = (P_1, \ldots, P_{n_i})$ and a *query sequence* $Q = (P_1, \ldots, P_n)$, the objective is to find such k database sequences that are the most similar to the query sequence Q. The similarity between the query Q and any database sequence D_i is quantified using a distance function $dist(Q, D_i)$ that operates over their corresponding SCLs. Our approach does not anyhow limit the length of query or database sequences, so they can correspond to long skeleton recordings, short pre-segmented actions, or their combinations.

3.2 Partitioning Skeleton Sequences

For efficient content-based management of especially longer skeleton sequences, it is necessary to partition them into meaningfully-sized segments. The segment-level representation constitutes the smallest processing unit that preserves a

reasonable volume of spatio-temporal information and is much better manageable in comparison with either many only-spatial poses, or hardly-processable continuous sequences. In addition, processing on the level of segments can be utilized in a broad variety of tasks.

The most straightforward way is to apply a mechanical slicing of a skeleton sequence into *fixed-size* segments. There is no optimal length of segments, but the rule of thumb suggests that such length should be upper-bounded by the length of the shortest retrievable query. Besides the segment length, the problem is that the segments originating from the query need not be perfectly aligned with the segments covering query-relevant parts within database sequences – thus, such relevant database sub-sequences can become unfindable. To overcome this problem, we apply the *overlapping* segment principle, which balances the trade-off between data findability at the price of increased data redundancy. The appropriate overlap between two consecutive segments is often set to between 50–80 %. For example, the 50 % segment overlap is illustrated in Fig. 1.

Formally, we partition a continuous skeleton sequence $(P_1, \ldots, P_n)$ into a list of segments $(S_1, \ldots, S_m)$. The i-th segment S_i ($i \in [1, m]$) is represented by the following sub-sequence of f poses:

$$S_i = (P_{(i-1)\cdot ss+1}, \ldots, P_{(i-1)\cdot ss+f}),$$

where $f \in \mathbb{N}$ is the fixed *segment length* and $ss \in \mathbb{N}\,(1 \leq ss \leq f)$ is the fixed *segment shift* (in number of poses), determining that two consecutive segments overlap in $(1 - \frac{ss}{f}) \cdot 100\,\%$ frames. For the skeleton sequence of n poses, this segmentation policy generates $\lfloor \frac{n-f}{ss} \rfloor = m$ segments in total.

3.3 Learning Codes for Segment Data

Given a set $\{S_i\}$ of unlabelled segments $S_i \in \mathbb{R}^{f \times j \cdot 3}$ that are extracted from training sequences, we want to learn an encoding function $codeTrans : \mathbb{R}^{f \times j \cdot 3} \rightarrow \mathbb{R}^d$ that maps segments $\{S_i\}$ to small semantic codes $\{C_i\}, C_i \in \mathbb{R}^d$. The similarity between codes (e.g., their cosine similarity) should reflect the semantic similarity between the original segment data. To learn $codeTrans(\cdot)$ in an unsupervised way, we apply the reconstruction-based principle by adopting a deep generative model; specifically a Variational AutoEncoder (VAE) [11].

VAE formulation. We assume that the segment space $\mathbf{S} \subset \mathbb{R}^{f \times j \cdot 3}$ is induced by a latent code space $\mathbf{C} \subset \mathbb{R}^d$. Following the commonly used VAE terminology, in our formulation, the encoder network Enc_ϕ takes as input a segment $S \in \mathbf{S}$ and produces a distribution $q_\phi(C|S)$ over the latent code space $\mathbf{C}$ describing the codes that could have generated S. Specifically,

$$\left(\mu_C, \sigma_C^2\right) = \mathrm{Enc}_\phi(S) \tag{1}$$

$$q_\phi(C|S) \sim \mathcal{N}(\mu_C, \sigma_C^2 I)\,, \tag{2}$$

where $q_\phi(C|S)$ is defined as a Gaussian distribution whose parameters (the mean $\mu_C \in \mathbb{R}^d$ and diagonal covariance matrix whose diagonal values are $\sigma_C^2 \in \mathbb{R}^d$) are

produced by $\text{Enc}_\phi(S)$. Similarly, the decoder network Dec_θ takes a code C and defines $p_\theta(S|C)$ (the distribution of sequences in $\mathbf{S}$ corresponding to the latent code C) by providing $\mu_S, \sigma_S^2 \in \mathbb{R}^{f \times j \cdot 3}$:

$$\left(\mu_S, \sigma_S^2\right) = \text{Dec}_\phi(C) \tag{3}$$

$$p_\theta(S|C) \sim \mathcal{N}(\mu_S, \sigma_S^2 I)\,. \tag{4}$$

The parameters of the encoder and decoder networks ϕ, θ are jointly optimized via mini-batch gradient descent by maximizing the evidence lower bound (ELBO):

$$\text{ELBO}(\theta, \phi, S_i) = \mathbb{E}_{C_i \sim q_\phi(C|S_i)}\left[\log p_\theta(S_i|C_i)\right] - \beta \cdot D_{KL}\left(q_\phi(C|S_i) \,||\, p(C)\right), \tag{5}$$

where β is a hyperparameter that controls the trade-off between reconstruction accuracy and latent code disentanglement [7]. When assuming a normal prior for $p(C)$ (i.e., $C \sim \mathcal{N}(0, I)$), maximizing Equation 5 reduces to minimizing the following loss function for a sample sequence S:

$$\begin{aligned}\mathcal{L}(\theta, \phi, S) = &\sum_{j=1}^{f \times j \cdot 3} \left(\frac{\left(S^{(j)} - \mu_S^{(j)}\right)^2}{2\sigma_S^{2(j)}} + \log \sigma_S^{(j)} \right) \\ &+ \frac{\beta}{2} \sum_{k=1}^{d} \left(\mu_C^{(k)} + \sigma_C^{2(k)} - 1 - \log \sigma_C^{(k)} \right),\end{aligned} \tag{6}$$

where the notation $x^{(j)}$ indicates the j-th component of the x vector. The first term in Equation 6 represents the negative log-likelihood of the sample S given the reconstruction mean μ_S and variance σ_S^2 produced by the decoder. The variance σ_S^2 can be interpreted as the uncertainty of the reconstruction [9]; the decoder is pushed to minimize uncertainty (via the $\log \sigma_S^{2(j)}$ term) but is discouraged to output a low uncertainty when the predicted mean μ_S deviates too much from the original sample S (via the $(S^{(j)} - \mu_S^{(j)})^2 / 2\sigma_S^{2(j)}$ term). The second term of Equation 6 is a regularization term that pushes codes produced by the encoder to be normally distributed, thus reducing code overfitting.

Encoder and Decoder Architectures. The encoder and decoder networks are implemented as residual 1D convolutional networks [6]. The encoder is comprised of a single-layer 64-channels convolutional stem followed by three residual blocks and a last fully connected layer. In the second and third residual blocks, the time dimension is halved by 2-stride convolutions, while channels are doubled. The decoder network is comprised of four residual blocks and a final convolutional layer; before each residual block, the input is upsampled by a factor 2 in the time dimension until it matches the correct segment size, while the channel dimension is halved by each block. A residual block is implemented as BN-ReLU-Conv-BN-ReLU-Conv, where BN and Conv are 1-dimensional batch normalization and convolutional layers, respectively. A convolutional layer is added in the shortcut path of the residual block when output and input dimensionalities do not match.

Segment Encoding. Once trained, we adopt the encoder network to transform the skeleton data of each segment into a code. Specifically, for each segment S_i, we take the mean parameter μ_C produced by $\mathrm{Enc}_\phi(S_i)$ as code C_i:

$$C_i = codeTrans(S_i) = \mathrm{Enc}_\phi(S_i)[0]\,, \tag{7}$$

where [0], with abuse of notation, indicates the selection of only the first output of the encoder. The similarity between codes is quantified using the cosine distance.

3.4 Determining Similarity of SCL Representations

For the purpose of k-nearest neighbor retrieval, there is a need to determine a similarity between the query Q and any database sequence D_i. Let us recall that the query and database sequences have to be first transformed into the SCL representation by partitioning a given sequence into segments and transforming each segment into the code using the $codeTrans(\cdot)$ function. Thus, the query Q is then represented by its SCL as $(C_1, \ldots, C_m)$ and the database sequence D_i as $(C'_1, \ldots, C'_{m'})$. Since the lengths of SCLs can be generally different, i.e., $m \neq m'$, the time-warping or bag-of-words principle constitutes possible candidates for similarity-based comparison. To respect the temporal order of codes, we have decided to apply the Dynamic Time Warping (DTW) distance function to compare two SCLs, where the similarity of two particular codes C_i and C'_j inside DTW is quantified using the cosine distance (as stated in Section 3.3):

$$codeDist(Q, D_i) = \frac{1}{\overline{m}} \cdot DTW\left((C_1, \ldots, C_m), (C'_1, \ldots, C'_{m'})\right). \tag{8}$$

We further normalize the DTW distance by the length of the warping path $\overline{m}$ ($\max\{m, m'\} \leq \overline{m} \leq m + m'$) inside the DTW matrix (i.e., by the number of identified mappings between pairs of codes) so that shorter sequences are not favored at the expense of longer ones with respect to the same query.

The disadvantage of DTW is its quadratic time complexity. On the other hand, the SCLs are typically quite short. In case long SCLs are needed to be compared, several DTW enhancements can possibly be applied to decrease processing time up to linear complexity [21].

4 SCL in Retrieval Applications

We experimentally verify that the SCL representation preserves important characteristics of 3D skeleton segment data in the context of two popular applications: *action retrieval* and *action classification*. The evaluation on the level of pre-segmented actions allows us to demonstrate that the SCL approach trained without the information about pre-segmented actions or their labels can achieve high effectiveness even when compared to the purposely trained classifiers.

4.1 Datasets

Even though there is a variety of 3D skeleton datasets, they usually provide only pre-segmented actions used for supervised or self-supervised learning tasks (such as NTU RGB+D 60/120 [15] or Kinetics 400 [8]). To properly evaluate our approach, which does not assume anything about pre-segmentation, we need datasets that provide continuous (unsegmented) skeleton sequences. The only suitable possibilities are the following two datasets.

- **HDM05** dataset [18] contains 324 continuous skeleton sequences with the total length of about 3.5 hours, which corresponds to 1.5 M frames with the frame-per-second rate (FPS) of 120 Hz. We selected a subset of 241 sequences in which 2,345 actions belonging to 130 classes are labeled.
- **PKU-MMD** dataset [13] provides 860 single-subject sequences with the total length of 20 hours captured with the FPS rate of 30 Hz. Such sequences contain almost 20 K labeled single-subject actions that are categorized in 43 classes. The dataset defines the cross-view (**CV**) and cross-subject (**CS**) evaluation scenarios that specifically divide the actions as well as sequences into training and test batches.

As recommended in most of the papers, we also pre-process the datasets by downsampling the skeleton data (downsampling HDM05 10 times from 120 to 12 and PKU-MMD 3 times from 30 to 10) and applying the position, orientation, and skeleton-size normalization.

4.2 Evaluation Methodology of Retrieval Applications

To support unsupervised segment-code learning, we use only continuous (unsegmented) skeleton sequences without any information about pre-segmented actions or their labels. We train one model for the whole HDM05 dataset and two models for the PKU-MMD dataset corresponding to the CV and CS scenarios. As discussed in Section 3.2, we synthetically partition each skeleton sequence into a series of fixed-size overlapping segments. As recommended in [1,22], we fix the segment length to 0.666 s with the segment shift of 0.133 s, which corresponds before downsampling to 80 and 24 frames with the segment shift of 16 and 5 frames for the HDM05 and PKU-MMD dataset, respectively. After downsampling, this results in 8-frame segments for both datasets. In total, 70 K segments were generated from the 241 HDM05 sequences and 1.2 M from the 860 PKU-MMD sequences. All the HDM05 segments were used to train the HDM05 model, while the subsets of PKU-MMD segments originating from the training sequences specified for the CV/CS evaluation scenarios were only used to train the models for the CV and CS scenarios. For training purposes, such segments were randomly split in the 80:20 manner to define the sets for the training and validation phases of each model.

To study the effectiveness of the proposed approach, we need a ground truth that is, however, defined only on the level of actions. Therefore, we focus on traditional action-retrieval and action-classification applications which we evaluate

using k-nearest neighbor (kNN) *queries*. In both applications, the model trained on unsegmented sequences is used to extract the SCL representation for each dataset action. Then, each kNN query is evaluated in a sequential way by computing the normalized DTW distance between the specific query-action SCL and each database-action SCL (see Equation 8). In particular, on the PKU-MMD dataset, the queries correspond to the *test actions* and the database actions to the *training actions* defined on the CS/CV scenarios. On the HDM05 dataset, there is no standard evaluation protocol, so the leave-one-out approach is applied over all the 2,345 actions.

For the action-retrieval application, we quantify *effectiveness* as the average *precision* (*Precision@k*) of all the kNN queries, where the query precision is computed as a ratio of correctly retrieved actions. An action is considered as correctly identified if it belongs to the same class as the query action. For the action-classification application, we evaluate different values of k and apply the kNN classifier as adopted in [22]. We measure the application effectiveness as the average *classification accuracy* (*Accuracy@k*) over all the queries. For both scenarios, we measure *efficiency* as the average time (in milliseconds) needed to evaluate a single query on the collection of database actions.

4.3 Effectiveness and Efficiency Results

Fig. 2 reports the effectiveness-efficiency trade-off when varying the SCL dimensionality d (blue lines) in both action-retrieval and action-classification scenarios. As a reference, we also report the results of the *baseline* configuration (green cross) that uses DTW to determine the similarity of actions on the level of individual poses; the distance between poses inside DTW is implemented as the sum of the Euclidean distances between corresponding raw joint coordinates. SCL representations deliver a much improved effectiveness in both scenarios (with $d = 256$, a Precision@1 of 87.42% vs 75.22% on HDM05, 90.03% vs 83.58% on PKU-MMD (CV), 74.41% vs 62.34% on PKU-MMD (CS), and an Accuracy@5 of 87.59% vs 77.31% on HDM05, 90.44% vs 81.19% on PKU-MMD (CV), 77.80% vs 59.43% on PKU-MMD (CS)). These results indicate that the learned SCL representations preserve semantic information. From the efficiency point of view, the comparison is quite fast as the DTW function is applied to relatively short SCLs – a single action contains 12 and 24 codes on average for the HDM05 and PKU-MMD datasets, respectively. As a result, we reduce the average query processing time by more than an order of magnitude with respect to operating on raw joint coordinates.

Fig. 3 shows the effectiveness in both scenarios when considering different numbers k of nearest neighbors during query processing. The best effectiveness is often reached when $k \in [3, 10]$, with SCL representations being more robust to the choice of k in the classification scenario with respect to the baseline. A very high effectiveness is already achieved when the code dimensionality d equals to 64 (red line), which means that the dimensionality of original segment data is reduced more than ten times.

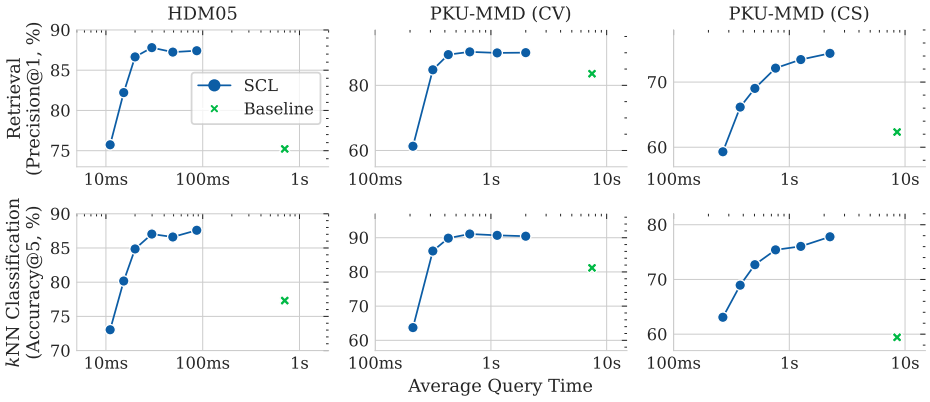

Fig. 2: Effectiveness-Efficiency trade-off of SCL. Effectiveness is measured in the kNN-based retrieval and classification scenarios as Precision@1 and Accuracy@5, respectively. Efficiency is measured as the average query-processing time needed to evaluate a single kNN query, and plotted on a logarithmic scale.

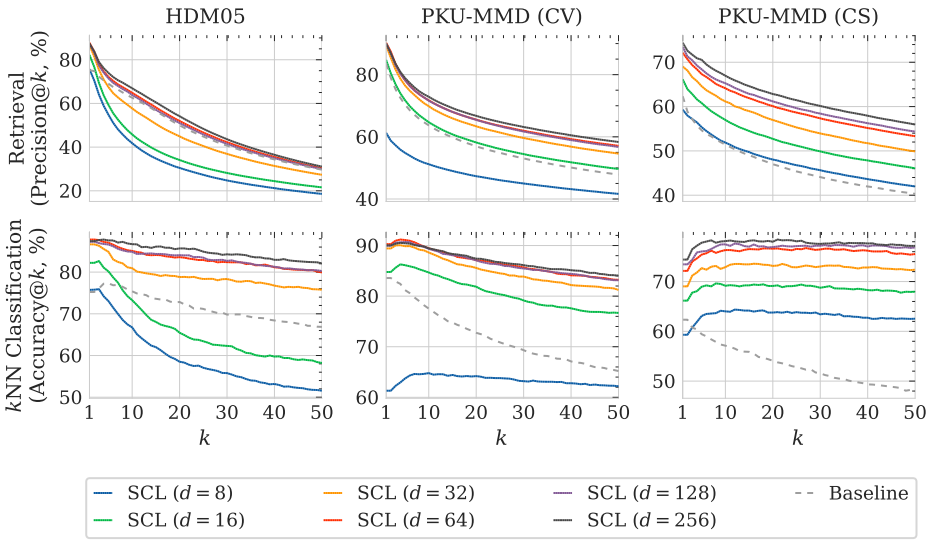

Fig. 3: kNN effectiveness of SCL for both retrieval and classification scenarios, when varying the number k of nearest neighbors during query processing. Dashed lines represent the baseline effectiveness.

Table 1: Precision@1 (%) of SCL representations for different settings of latent code dimensionality d and hyperparameter β controlling the trade-off between segment-reconstruction accuracy and latent-code disentanglement.

(a) HDM05

	β				
d	0	.01	.1	1	10
8	75.1	77.0	74.4	75.8	**77.4**
16	80.6	81.2	81.1	**82.2**	82.2
32	85.1	84.4	85.5	**86.7**	82.1
64	87.2	87.5	87.6	**87.8**	80.9
128	86.8	87.6	**87.9**	87.3	83.1
256	87.5	87.2	**87.9**	87.4	82.8

(b) PKU-MMD (CV)

	β				
0	.01	.1	1	10
61.6	62.5	63.2	61.3	**63.7**
74.1	75.6	78.4	**84.8**	74.5
78.7	78.5	80.8	**89.4**	74.7
82.7	81.3	83.9	**90.3**	73.0
85.5	85.2	88.0	**90.0**	69.7
84.5	84.2	88.6	**90.0**	79.0

(c) PKU-MMD (CS)

	β				
0	.01	.1	1	10
55.0	58.6	**60.3**	59.3	55.4
63.1	62.8	65.1	**66.2**	56.0
65.4	66.6	68.3	**69.0**	53.9
67.9	68.1	70.0	**72.1**	51.5
71.7	70.9	72.3	**73.5**	56.6
70.4	69.4	72.4	**74.4**	58.5

Regarding the training phase of SCL representations, a coarse grid search over the parameter β (see Table 1) showed that $\beta = 1$ delivers an optimal or comparable effectiveness most of the time. There is an exception for small SCL dimensionalities ($d = 8$) where tuning β led to slight improvements. Besides evaluation of effectiveness and efficiency, we also employed the trained VAE decoder to reconstruct original skeleton data of a segment purely from its latent code representation. In Fig. 4, we illustrate examples of reconstructed segment data for two specific HDM05 segments.

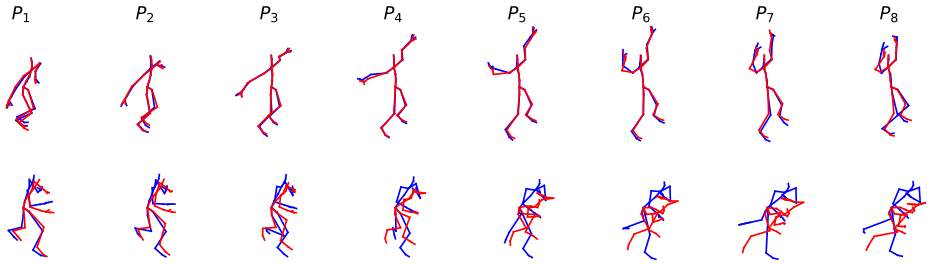

Fig. 4: Examples of original (blue) and reconstructed (red) segment poses from the HDM05 dataset. The top (bottom) row depicts a success (failure) case of reconstruction using our VAE model with $d = 256$ and $\beta = 1$.

4.4 State-of-the-Art Comparison

The state-of-the-art skeleton-data processing primarily focuses on classifying pre-segmented and labeled actions. For this reason, we compare the results of our classification approach to the results of existing classifiers evaluated on the HDM05 and PKU-MMD datasets. Specifically, we adopt our kNN retrieval approach by fixing k to 4 as this setting reaches high classification accuracy. In

Table 2: Comparison of our approach with the existing supervised/unsupervised classifiers trained on the pre-segmented actions. The values of classification accuracy are taken from the referenced papers.

	HDM05	PKU (CV)	PKU (CS)
Supervised Approaches			
Activity images + CNN [25]	-	92.00	85.00
DSwarm-Net [2]	**90.67**	-	-
BiLSTM [5]	89.26	**92.11**	**84.73**
Unsupervised Approaches			
baseline: raw skeleton data + DTW	75.22	83.58	62.34
Motion words + DTW [22]	80.30	-	-
LSTM + triplet-loss [10]	83.76	-	-
MS^2L [12]	-	-	64.86
Our approach ($\beta = 1, d = 256, k = 4$)	**87.80**	**90.53**	**77.20**

Table 2, we demonstrate that our approach achieves superior accuracy on both datasets compared to existing unsupervised classifiers. Let us also emphasize that our solution is approaching the accuracy of supervised classifiers, even if no information about the pre-segmentation nor labels of actions was available in the training phase.

5 Conclusions

We have proposed a new skeleton-data representation that is learned using a unique combination of the β-VAE approach and a lightweight convolutional neural network. Such representation has several advantages in contrast to related approaches. First, the representation can be extracted for skeleton sequences of any length on the level of short segments. Second, the segment feature is learned from continuous skeleton sequences completely in an unsupervised way, without the requirement of knowledge of pre-segmented actions or their labels. Third, the learned segment feature preserves semantic information of the underlying skeleton data, which is confirmed by reaching much higher effectiveness in retrieval-based scenarios compared to the baseline approach. Fourth, the segment feature is very compact and efficiently comparable with the cosine distance, which supports indexing possibilities for the future. In addition, the universality of the proposed approach enables its high applicability in many tasks, e.g., not only for action recognition or detection but also for sub-sequence search.

Acknowledgements

This research is supported by anonymous grant No. XXX.

References

1. Aristidou, A., Cohen-Or, D., Hodgins, J.K., Chrysanthou, Y., Shamir, A.: Deep motifs and motion signatures. ACM Transactions on Graphics **37**(6), 187:1–187:13 (2018)
2. Basak, H., Kundu, R., Singh, P.K., Ijaz, M.F., Wozniak, M., Sarkar, R.: A union of deep learning and swarm-based optimization for 3d human action recognition. Scientific Reports **12**(5494), 1–17 (2022)
3. Cheng, Y.B., Chen, X., Chen, J., Wei, P., Zhang, D., Lin, L.: Hierarchical transformer: Unsupervised representation learning for skeleton-based human action recognition. In: IEEE International Conference on Multimedia and Expo (ICME). pp. 1–6 (2021)
4. Dubey, S., Dixit, M.: A comprehensive survey on human pose estimation approaches. Multimedia Systems pp. 1–29 (2022)
5. Elias, P., Sedmidubsky, J., Zezula, P.: Understanding the limits of 2d skeletons for action recognition. Multimedia Systems **27**(3), 547–561 (2021)
6. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: European Conference on Computer Vision (ECCV). pp. 630–645. Springer (2016)
7. Higgins, I., Matthey, L., Pal, A., Burgess, C.P., Glorot, X., Botvinick, M.M., Mohamed, S., Lerchner, A.: beta-vae: Learning basic visual concepts with a constrained variational framework. In: 5th International Conference on Learning Representations (ICLR). pp. 1–22. OpenReview.net (2017)
8. Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., Suleyman, M., Zisserman, A.: The kinetics human action video dataset. arXiv (2017)
9. Kendall, A., Gal, Y., Cipolla, R.: Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7482–7491 (2018)
10. Kico, I., Sedmidubsky, J., Zezula, P.: Towards Efficient Human Action Retrieval based on Triplet-Loss Metric Learning. In: 33rd International Conference on Database and Expert Systems Applications (DEXA). pp. 234–247. Springer International Publishing, Cham (2022)
11. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
12. Lin, L., Song, S., Yang, W., Liu, J.: MS2L: Multi-Task Self-Supervised Learning for Skeleton Based Action Recognition. In: 28th ACM International Conference on Multimedia (MM). pp. 2490–2498. ACM, New York, NY, USA (2020)
13. Liu, C., Hu, Y., Li, Y., Song, S., Liu, J.: PKU-MMD: A large scale benchmark for skeleton-based human action understanding. In: Workshop on Visual Analysis in Smart and Connected Communities (VSCC@MM). pp. 1–8. ACM (2017)
14. Liu, J., Song, S., Liu, C., Li, Y., Hu, Y.: A Benchmark Dataset and Comparison Study for Multi-Modal Human Action Analytics. ACM Transactions on Multimedia Computing, Communications, and Applications **16**(2) (2020)
15. Liu, J., Shahroudy, A., Perez, M., Wang, G., Duan, L., Kot, A.C.: NTU RGB+D 120: A Large-Scale Benchmark for 3D Human Activity Understanding. IEEE Transactions on Pattern Analysis and Machine Intelligence (2019)
16. Liu, X., He, G., Peng, S., Cheung, Y., Tang, Y.Y.: Efficient human motion retrieval via temporal adjacent bag of words and discriminative neighborhood preserving dictionary learning. IEEE Transactions on Human-Machine Systems **47**(6), 763–776 (2017)

17. Lv, N., Wang, Y., Feng, Z., Peng, J.: Deep hashing for motion capture data retrieval. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 2215–2219. IEEE (2021)
18. Müller, M., Röder, T., Clausen, M., Eberhardt, B., Krüger, B., Weber, A.: Documentation Mocap Database HDM05. Tech. Rep. CG-2007-2, Universität Bonn (2007)
19. Papadopoulos, K., Ghorbel, E., Baptista, R., Aouada, D., Ottersten, B.E.: Two-stage rgb-based action detection using augmented 3d poses. In: 18th International Conference on Computer Analysis of Images and Patterns (CAIP). vol. 11678, pp. 26–35. Springer (2019)
20. Peng, W., Hong, X., Zhao, G.: Tripool: Graph triplet pooling for 3d skeleton-based action recognition. Pattern Recognition **115**, 107921 (2021)
21. Rakthanmanon, T., Campana, B.J.L., Mueen, A., Batista, G.E.A.P.A., Westover, M.B., Zhu, Q., Zakaria, J., Keogh, E.J.: Searching and mining trillions of time series subsequences under dynamic time warping. In: 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). pp. 262–270. ACM (2012)
22. Sedmidubsky, J., Budikova, P., Dohnal, V., Zezula, P.: Motion words: A text-like representation of 3d skeleton sequences. In: 42nd European Conference on Information Retrieval (ECIR). pp. 527–541. Springer (2020)
23. Sedmidubsky, J., Elias, P., Zezula, P.: Searching for variable-speed motions in long sequences of motion capture data. Information Systems **80**, 148–158 (2019)
24. 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)
25. Vernikos, I., Koutrintzes, D., Mathe, E., Spyrou, E., Mylonas, P.: Early fusion of visual representations of skeletal data for human activity recognition. In: 12th Hellenic Conference on Artificial Intelligence (SETN). ACM (2022)
26. Yang, Y., Liu, G., Gao, X.: Motion Guided Attention Learning for Self-Supervised 3D Human Action Recognition. IEEE Transactions on Circuits and Systems for Video Technology pp. 1–13 (2022)