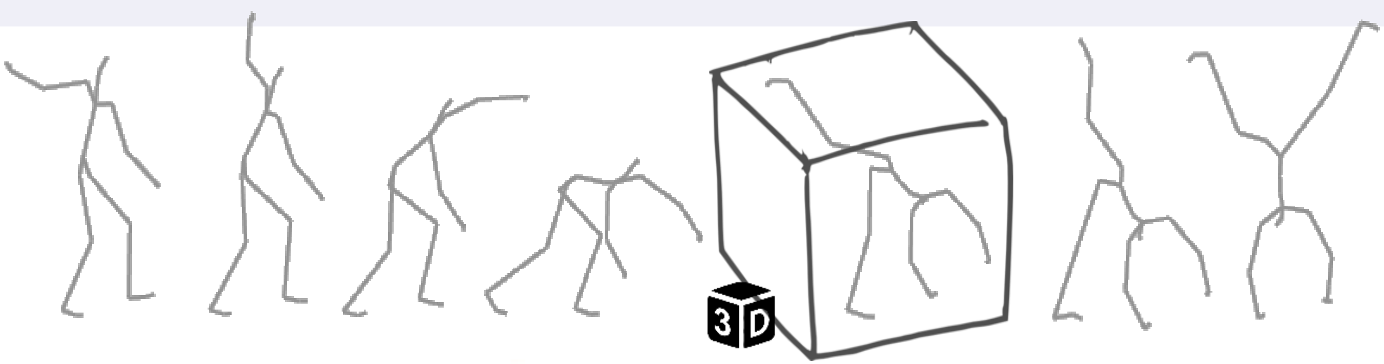
Learning Features for 3D Human-Skeleton Sequences

Jan Sedmidubsky
xsedmid@fi.muni.cz

Pavel Zezula
zezula@fi.muni.cz



Laboratory of Data Intensive Systems and Applications
disa.fi.muni.cz

Faculty of Informatics, Masaryk University, Czech Republic
fi.muni.cz

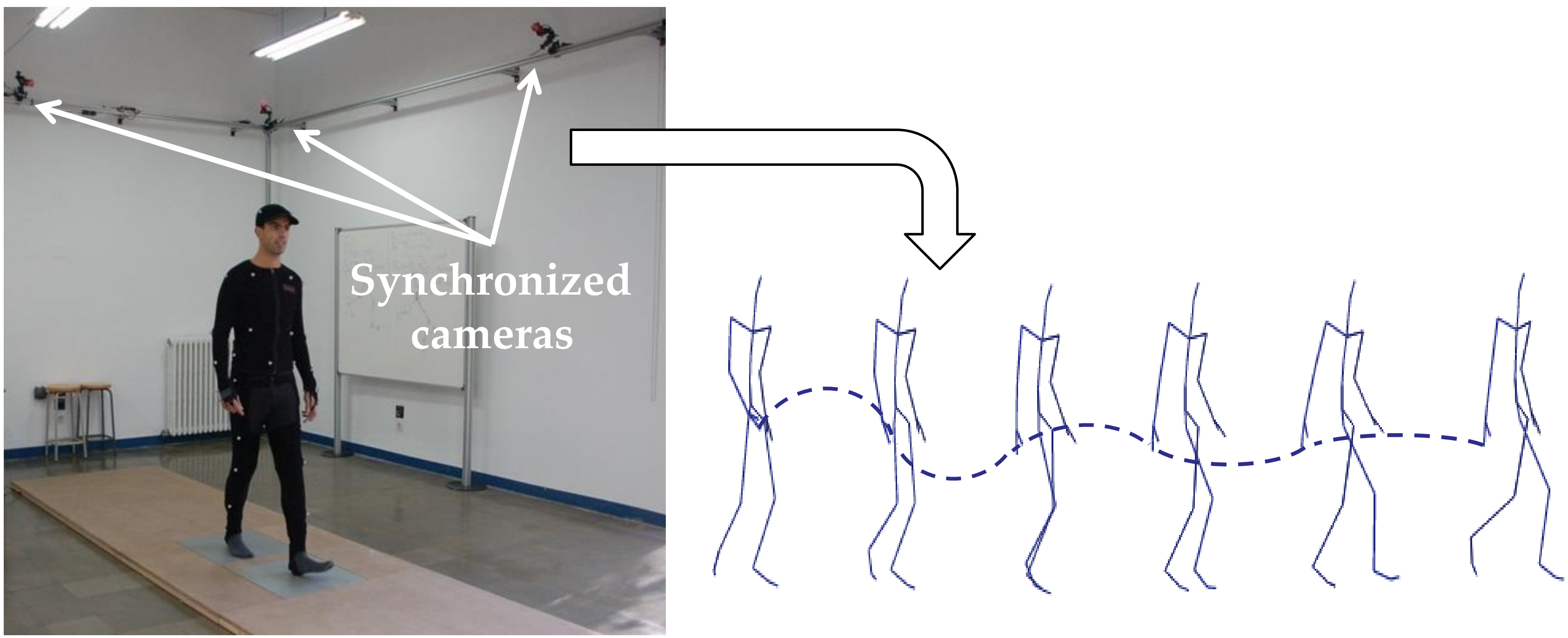
Outline

1) Motion Data: Representation, Applications, Operations
2) Similarity of Motion Sequences
3) Learning Motion Features for Similarity Comparison
   - CNN Features
   - LSTM Features
4) Applying Learned Features for *k*NN Classification
5) Enhancing Feature Learning by Data Augmentation
   1) Cropping/Extending/Shifting the Motion Content
   2) Adding Noise to 3D Joint Coordinates

1 Motion Data

3D Skeleton Sequences ~ Motion Capture Data ~ MoCap Data ~ Motion Data

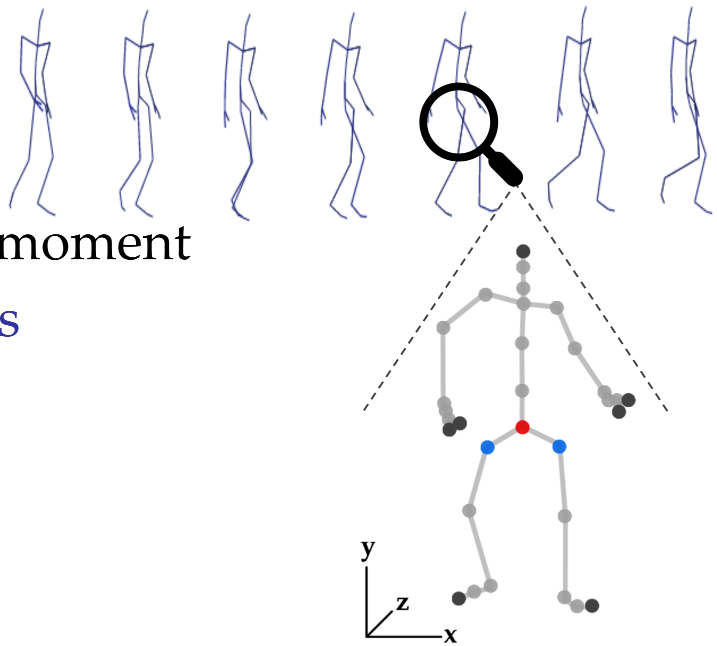
- Continuous spatio-temporal characteristics of a human motion simplified into a discrete sequence of 3D skeletons



3D skeleton sequences

- Skeleton pose:
  - Skeleton configuration in a given time moment
  - 3D positions of body landmarks ~ joints

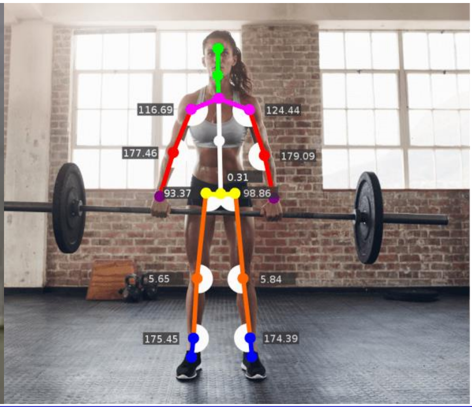
- Different views on motion data:
  - A sequence of 3D skeleton poses
  - A set of 3D trajectories of joints



Pose captured in a given time moment

Applications

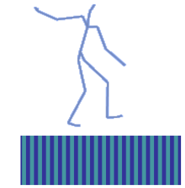
- Many application domains where motion data have a great potential to be utilized and automatically processed
  - Computer animation – virtual and augmented reality
  - Medicine – rehabilitation, detection of movement disorders
  - Sports – assessment of performance, digital referees
  - Smart-homes – detection of falls of elderly people
  - Military – simulation of conflict resolving situations



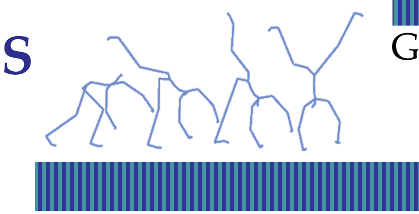
1 Data – Types of Motion Sequences

Motion data types

- **Short** motions:
  - Semantically-**indivisible** motions ~ **ACTIONS**
  - Length – typically in order of seconds
  - Database – usually a large number of actions
- **Long** motions:
  - Semantically-**divisible** motions ~ sequences of actions
  - Length – in order of minutes, hours, days, or even unlimited
  - Database – typically a single long motion processed either as a whole, or in the stream-based nature



Gait cycle (0.6 s)



Cartwheel (2.1 s)

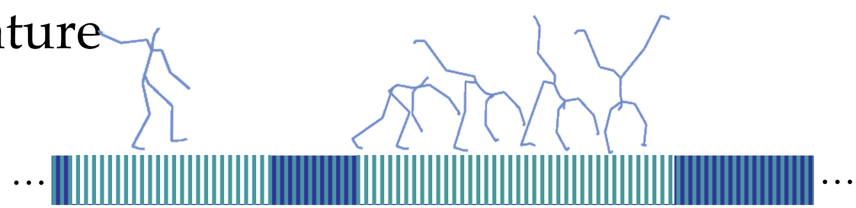
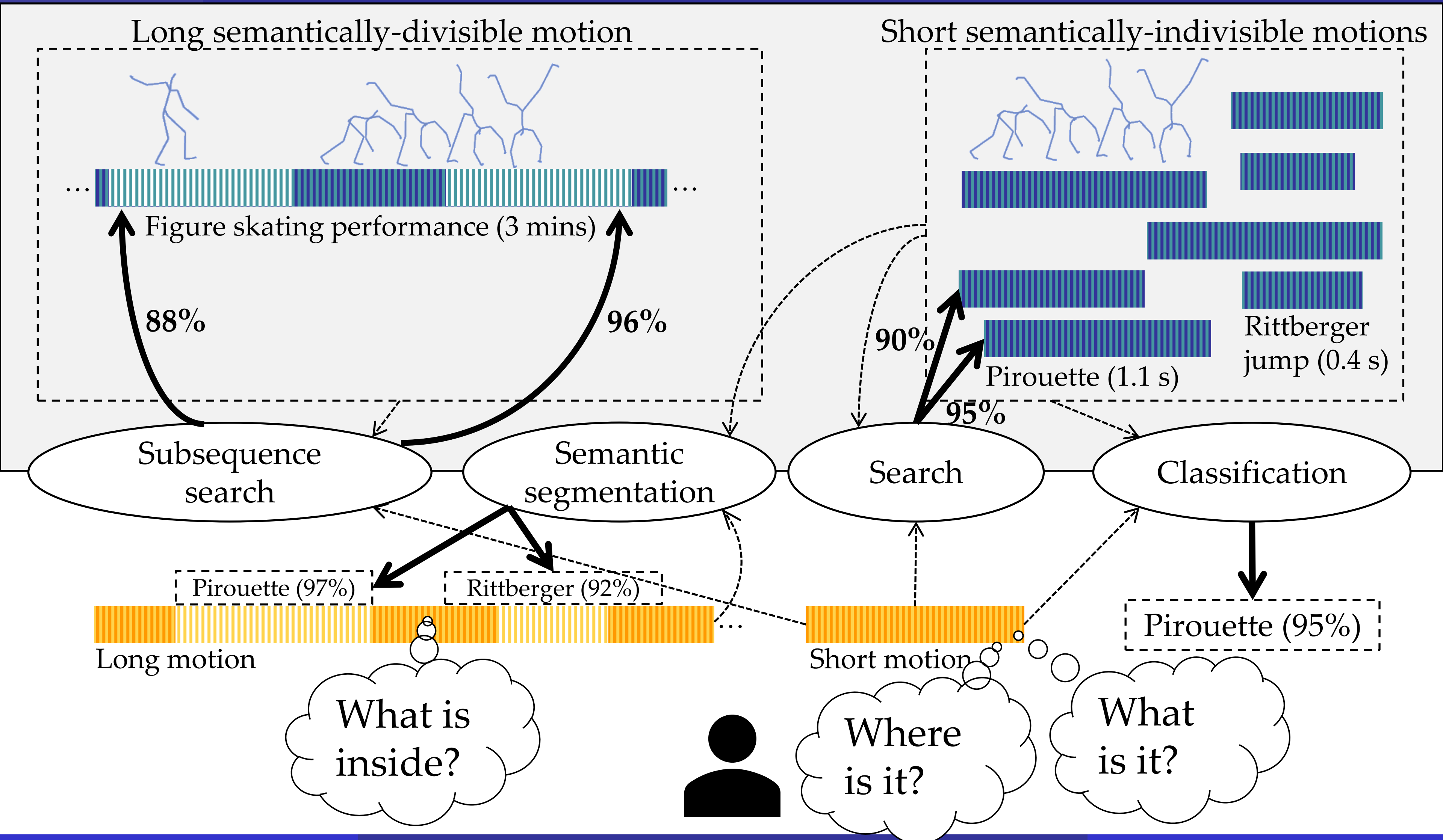


Figure skating performance (3 mins)

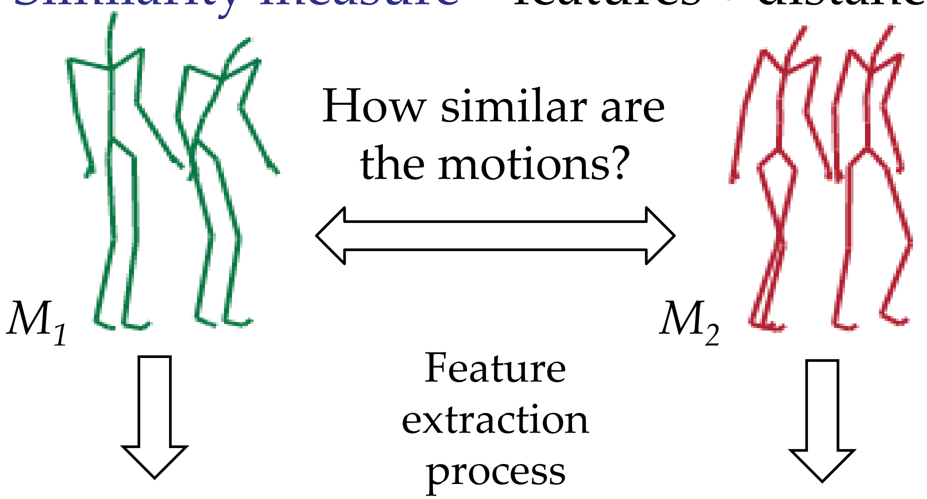
1 Motion-Analysis Operations



2 Similarity of Motion Sequences

Similarity of actions (short motions)

- Determining similarity is needed everywhere, e.g., for classification, searching, semantic segmentation, synthesis
  - Similarity measure = features + distance function



<0, 0, 5.2, 8.1, 0, 2.3, -1.1, 0, …> F_{M_1}

<0, 0.1, 6.2, 7.1, 0, 2.9, -2.1, 0, …> F_{M_2}

$$dist(F_{M_1}, F_{M_2}) = 8.56$$

2 Challenges of Similarity Measures

Objective

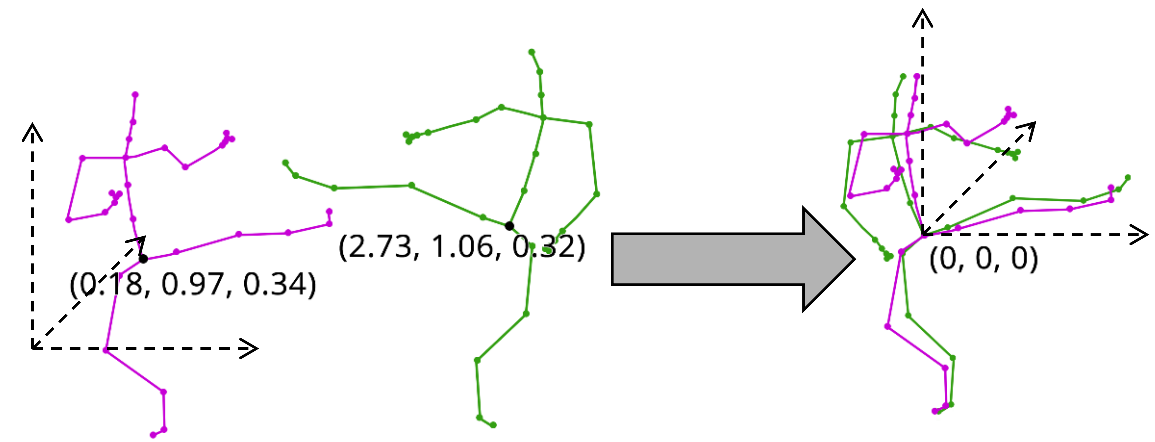
- To propose an effective and efficient similarity measure, i.e., features + distance function

Problems

- Similarity is application-dependent (*e.g., recognizing daily actions vs. recognizing people based on their style of walking*)
- Subjects have different bodies *(e.g., child vs. adult)*
- Spatial and temporal deformations – the same action (*e.g., kick*) can be performed at different:
  - Styles *(e.g., frontal kick vs. side kick)* and
  - Speeds *(e.g., faster vs. slower)*

Preprocessing step – data normalization

- Optional step depending on a target application
- Types – skeleton size and joint position and orientation
- Normalizing each pose independently vs. conditionally



Features

- Hand-crafted features – manual feature engineering
  - Low descriptive power – outperformed by ML approaches
  - E.g., time series of joint angle rotations compared by DTW
- Machine-learned features – learning features automatically
  - Large amount of training data needed
  - E.g., CNN, RNN, LSTM features:
    - 16–256D float vectors compared by the Euclidean distance

      [Coskun et al.: Human Motion Analysis with Deep Metric Learning. ECCV, 2018]
    - ?D float vectors compared by the Euclidean distance

      [Aristidou et al.: Deep Motifs and Motion Signatures, ACM Trans. Graph., 2018]
    - 4,096D float vectors compared by the Euclidean distance

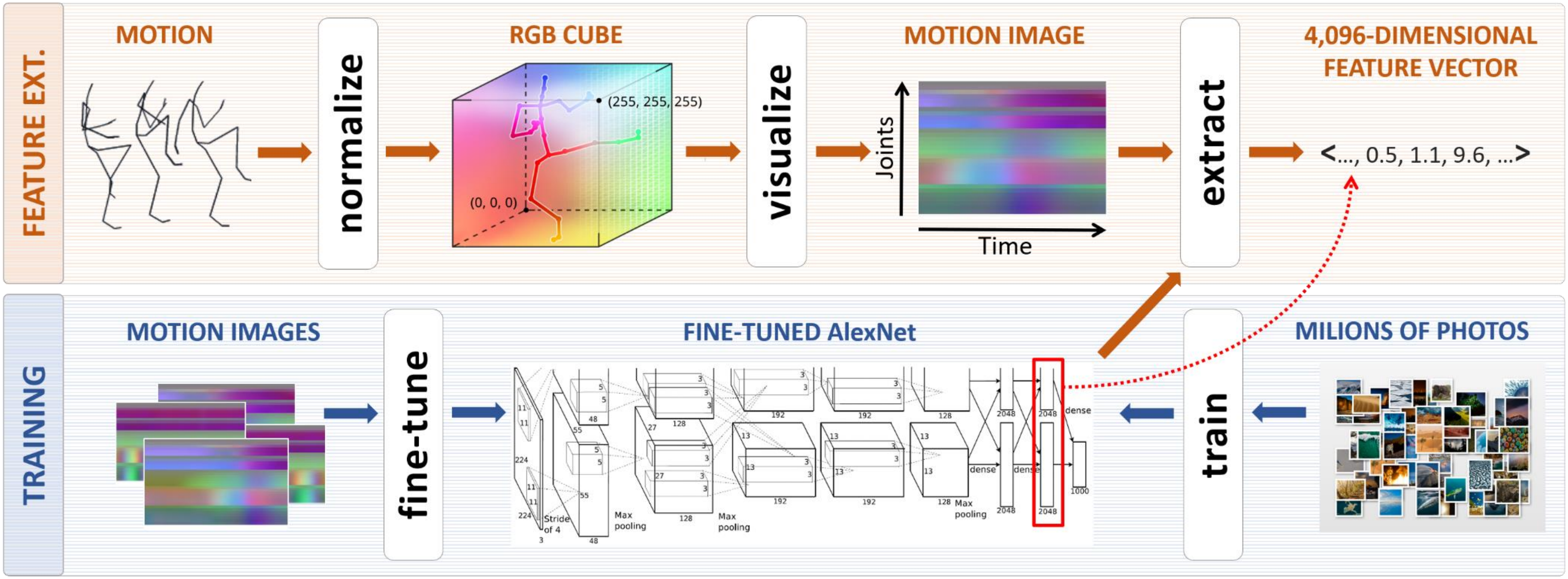
      [Sedmidubsky et al.: Effective and efficient similarity searching in motion capture data. MTAP, 2018]
    - 160D bit vectors compared by the Hamming distance

      [Wang et al.: Deep signatures for indexing and retrieval in large motion databases. Motion in Games, 2015]

Motion-image similarity measure (CNN features)

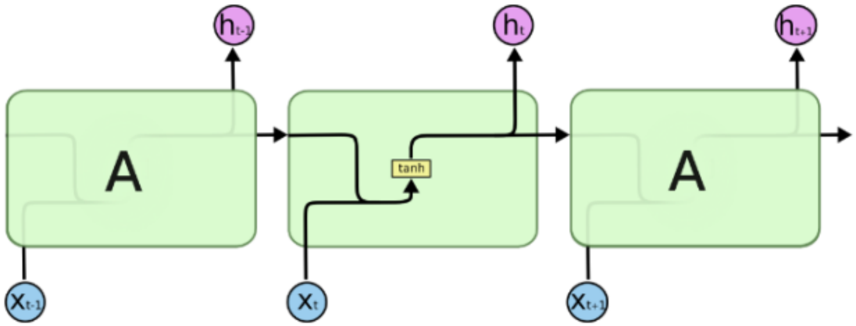
[Sedmidubsky et al.: Effective and efficient similarity searching in motion capture data. Multimedia Tools and Apps, 2018]

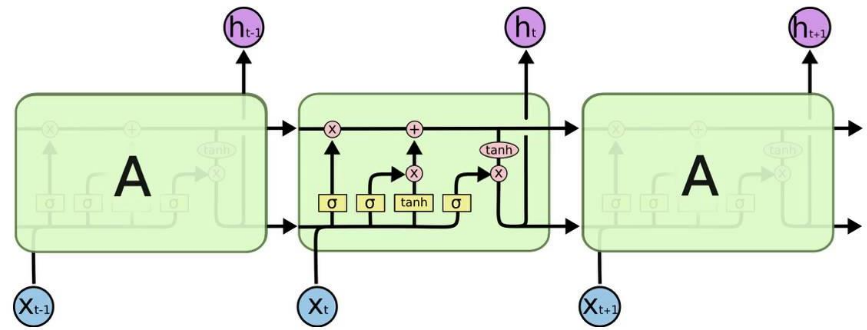
- Deep 4,096D features compared by the Euclidean distance
- Suitable for short motions in order of seconds (~ actions)



Recurrent Neural Networks (RNN)

- Output contents are influenced by the history of inputs

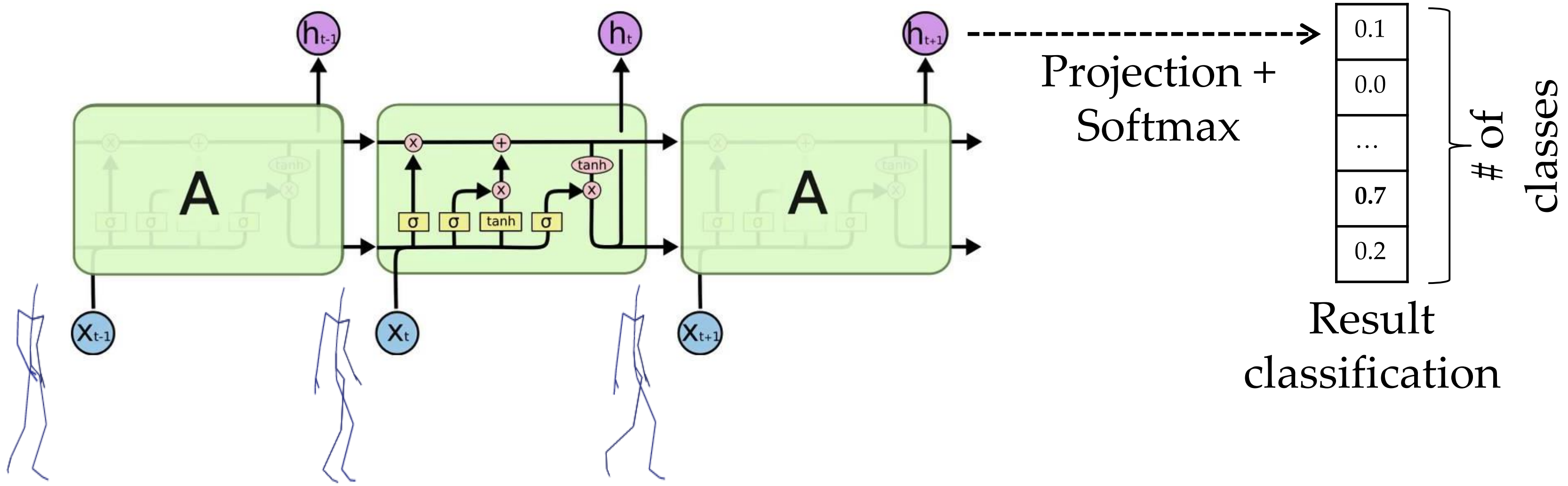




- **Long-Short Term Memory (LSTM)** network:
  - A special kind of RNN, capable of learning long-term dependencies
  - It learns when data should be remembered and when they should be thrown away

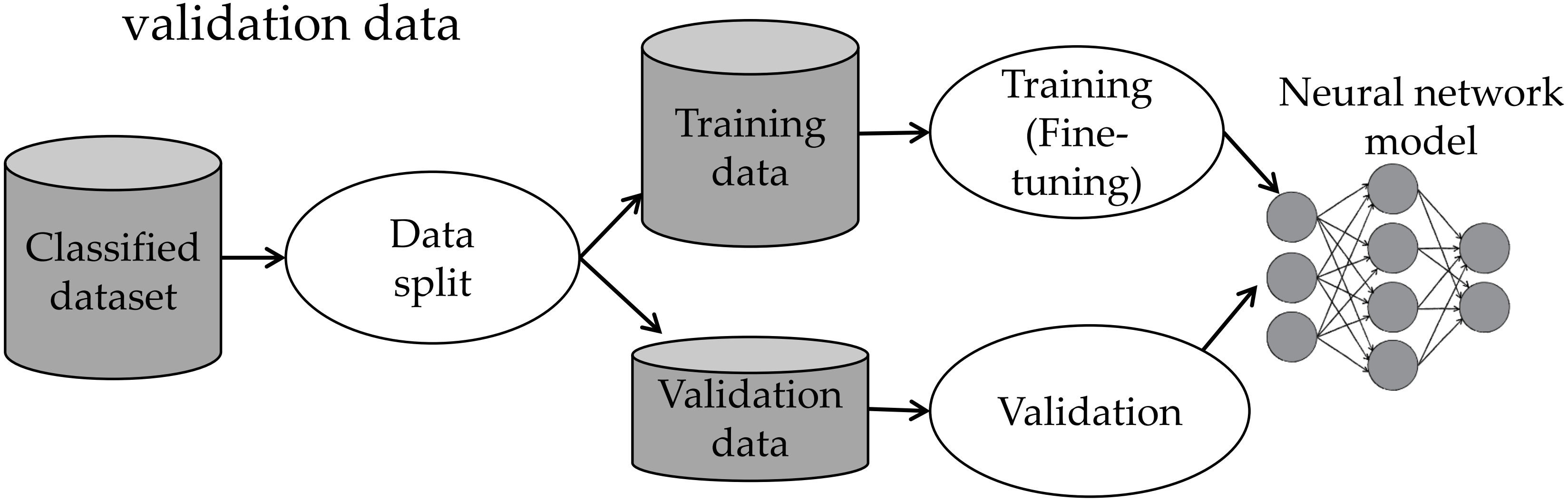
LSTM-based similarity measure (LSTM features)

- Number of states/cells corresponds to the number of poses
- The last state h_{t+1} can be used as a feature
- Size of each state h_i is a user-defined parameter
  - Suitable state size of 512 / 1,024 / 2,048 dimensions



Training CNN/LSTM models for classification and/or feature extraction

- Training a neural network model (CNN/LSTM)
  - Labelled training and validation data ~ actions categorized in classes
  - Training in epochs (usually hundreds of epochs)
  - Result model taken from the epoch achieving the highest accuracy on validation data



3 Classification Dataset

DISA 2019

HDM05 dataset (120 Hz sampling, 31 body joints)

- Ground truth – **2,328**/2,345 actions in **122**/130 classes
  - Shortest and longest actions: 13 frames (0.1s) and 900 frames (7.5s)
  - Action classes corresponding to daily/exercising activities, e.g.:
    - "Clap with hands 5 times", "Walk two steps, starting with left leg", "Turn left", "Frontal kick by left leg two times", "Cartwheel, starting with left hand"

Learning & validation

- *n*-fold cross validation procedure – a standard statistical method to estimate the skill of machine learning models
- We adopt 2-fold cross validation – dataset split into 2 folds, either randomly or in a balanced way:
  1) 1^{st} fold used for training, 2^{nd} fold used for validation
  2) 2^{nd} fold used for training, 1^{st} fold used for validation

  make average

Training CNN/LSTM models for feature extraction

- Training times:
  - Training time linearly depends on the number of training actions
  - CNN (through motion images):
    - 1,164 actions ~ 2 hours
  - LSTM (influenced by parameters, e.g., feature size, action length, fps):
    - 1,164 actions ~ 1 hour
    - 10,476 actions ~ 9 hours
    - 20,952 actions ~ 18 hours
- Learned features:
  - CNN: 4,096D features + Euclidean distance
  - LSTM: 1,024D features + Manhattan distance

4 Applying Learned Features for Action Recognition

Reference collection

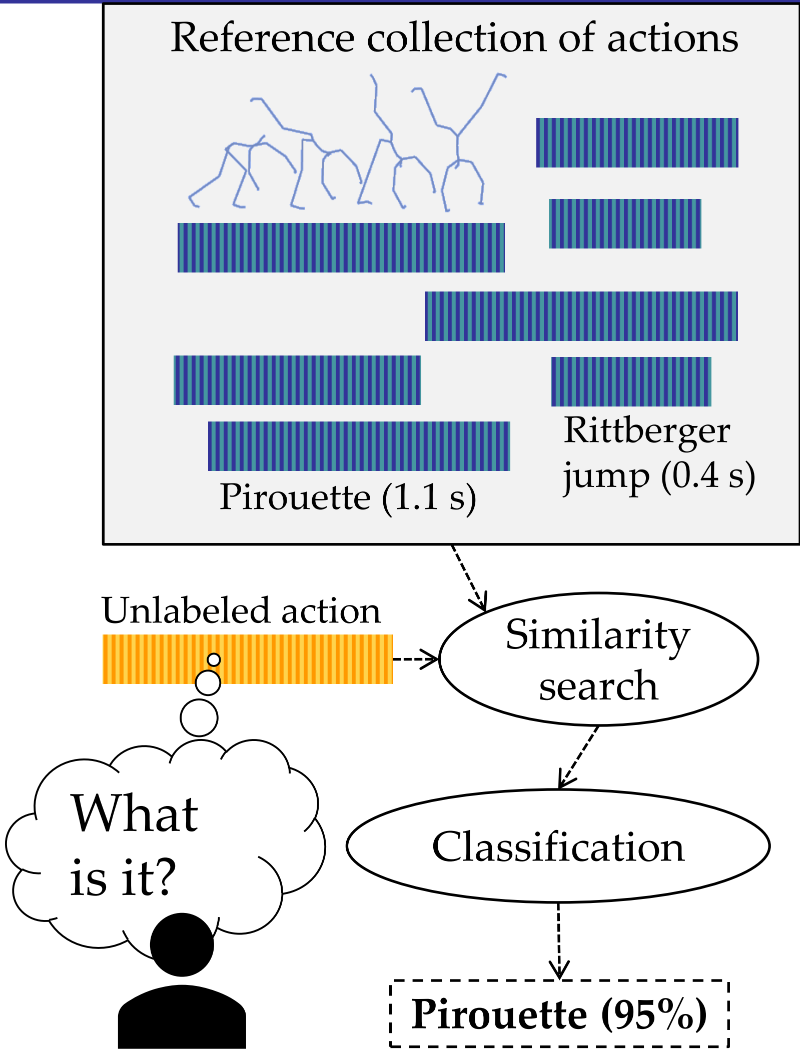
- Categorized features of training actions, obtained using the CNN/ LSTM neural network model

Similarity search

- Searching for the *k*-nearest actions to the unlabeled action, simply using the sequential scan

Classification

- Recognizing the unlabeled action class using the 1NN or Weighted-distance *k*NN classifier



1NN classification

- Searching for the nearest neighbor to the unlabeled action
- Class of the nearest neighbor considered as class of the query

Unlabeled action
feature vector
<…, 0.93, 10.1, 2.43, …>

1. **8.7 JUMP**
2. 10.9 KICK
3. 13.2 KICK
4. 14.3 KICK

⇒ JUMP (100%)

JUMP class
feature vectors
<…, 0.53, 10.8, 4.64, …>
<…, 0.12, 8.60, 1.99, …>

KICK class
feature vectors
<…, 8.93, 10.1, 2.43, …>
<…, 7.42, 7.14, 2.27, …>
<…, 3.93, 6.26, 3.41, …>

Weighted-distance *k*NN classifier

[Sedmidubsky et al.: Probabilistic Classification of Skeleton Sequences. DEXA, 2018]

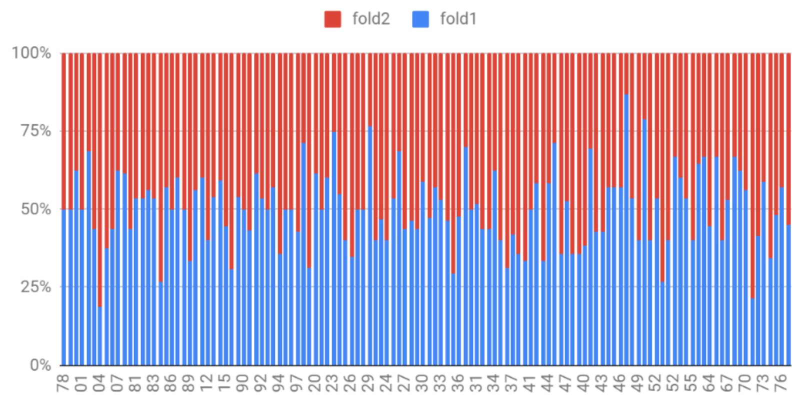
- Considering not only the number of votes but also the similarity of neighbors
  - Normalizing the neighbor distance with respect to the *k*-th neighbor
    - Effective when distances of nearest neighbors vary across classes
  - Computing class relevance by summing relevance of class neighbors (1 – normalized distance)

Original distances	Normalized distances	Relevance of neighbors	Relevance of classes	
1. 8.7 JUMP	1. 0.55 JUMP	1. 0.45 JUMP	0.45 JUMP	⇒ JUMP (45%)
2. 10.9 KICK	2. 0.69 KICK	2. 0.31 KICK	0.56 KICK	⇒ KICK (55%)
3. 13.2 KICK	3. 0.84 KICK	3. 0.16 KICK		
4. 14.3 KICK	4. 0.91 KICK	4. 0.09 KICK		

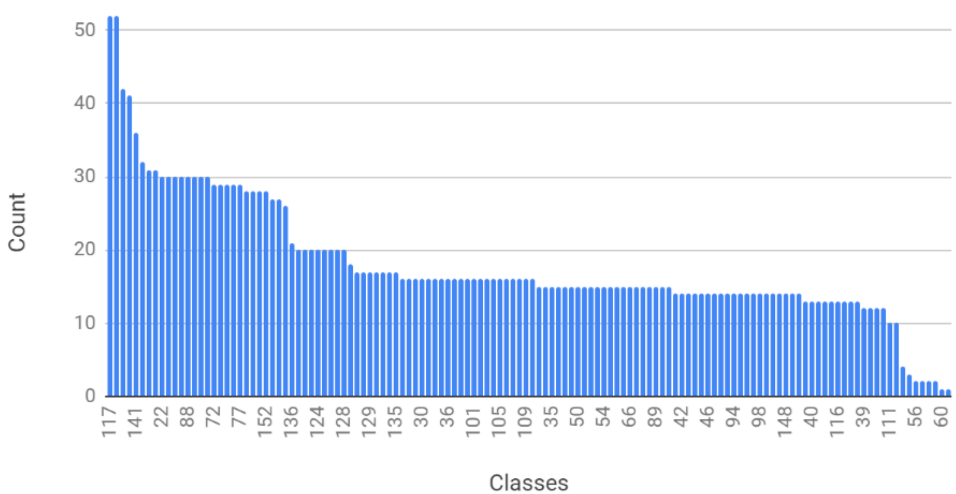
4 CNN/LSTM Recognition Results – Influence of Data Splitting Strategy

- Data: 2,328 actions in 122 classes
  - 1,164 training and 1,164 test actions
  - Fold-splitting strategies:
    - Random
    - Balanced

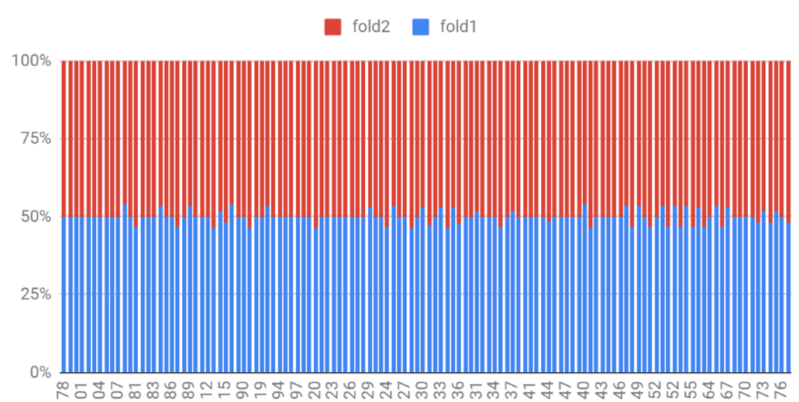
Random splits - ratio of same-class action count between fold1 and fold2



Number of actions in classes



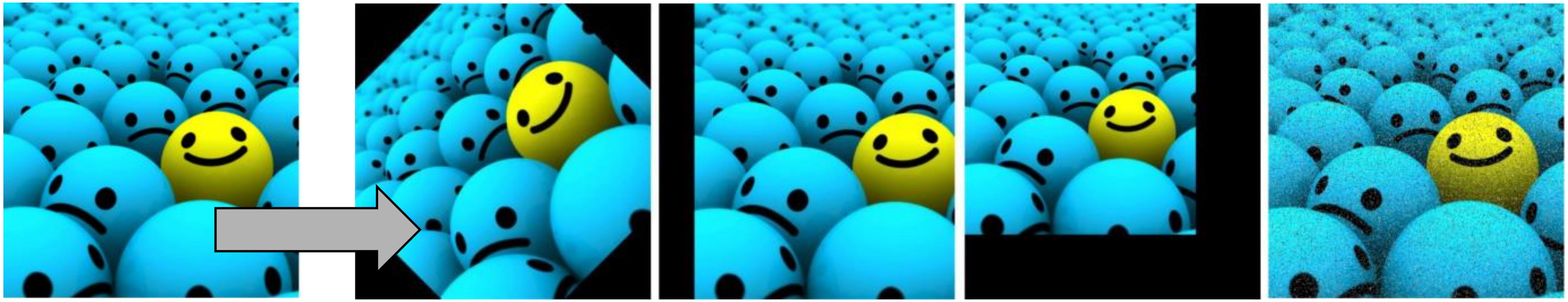
Balanced splits - ratio of same-class action count between fold1 and fold2



k

Augmentation of training data

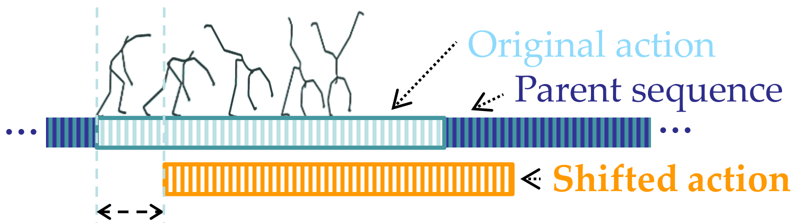
- Increasing the number of training actions artificially
  - Inspiration in the image domain – image flips, blurry images



- Proposed action augmentation techniques:
  1) Shifting/cropping/extending original actions
  2) Adding noise to 3D joint coordinates of original actions
  3) Combination of both the above techniques

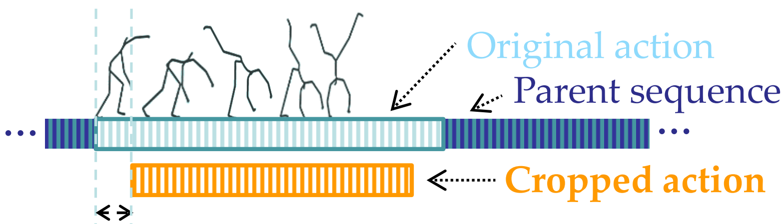
Augmentation (1)

- Shift – shifting boundaries of original actions within parent sequences



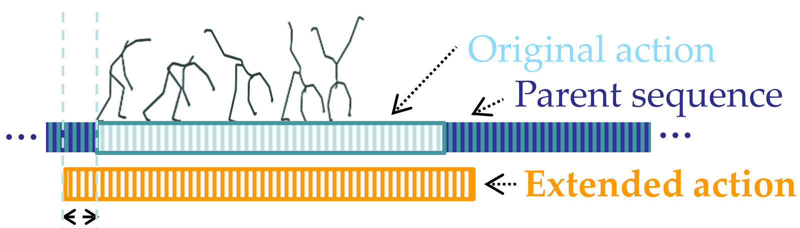
shift of 20% w.r.t. action length

- Crop – cropping original actions



crop of 20% w.r.t. action length (10% on left and 10% on right side)

- Extension – extending boundaries of original actions within parent sequences

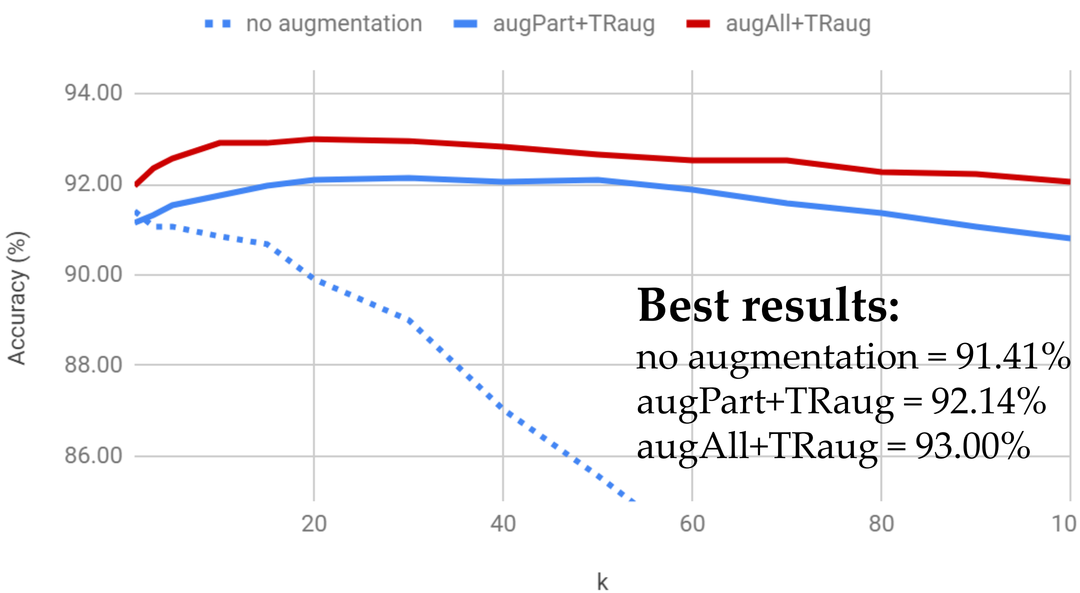


extension of 20% w.r.t. action length (10% on left and 10% on right side)

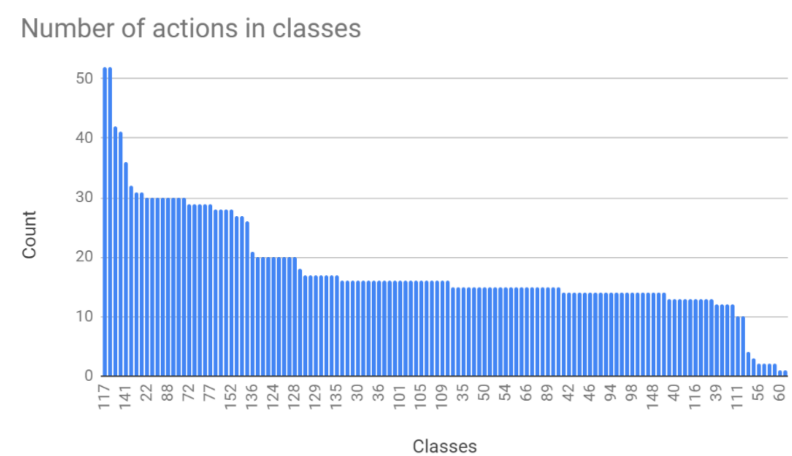
5 Shifting/Cropping/Extending Original Actions – TRaug Results

Training data augmentation (1) – shift/crop/extension

- no augmentation – **1,164** actions for learning; **1,164** reference actions
- augPart+TRaug – **5,820** actions for learning; **5,820** reference actions
  - Each action in 5 variants: original; 10/20% shift (left + right)
- augAll+TRaug – **10,476** actions for learning; **10,476** reference actions
  - Each action in 9 variants: original; 10/20% shift (left + right); 10/20% crop and extension

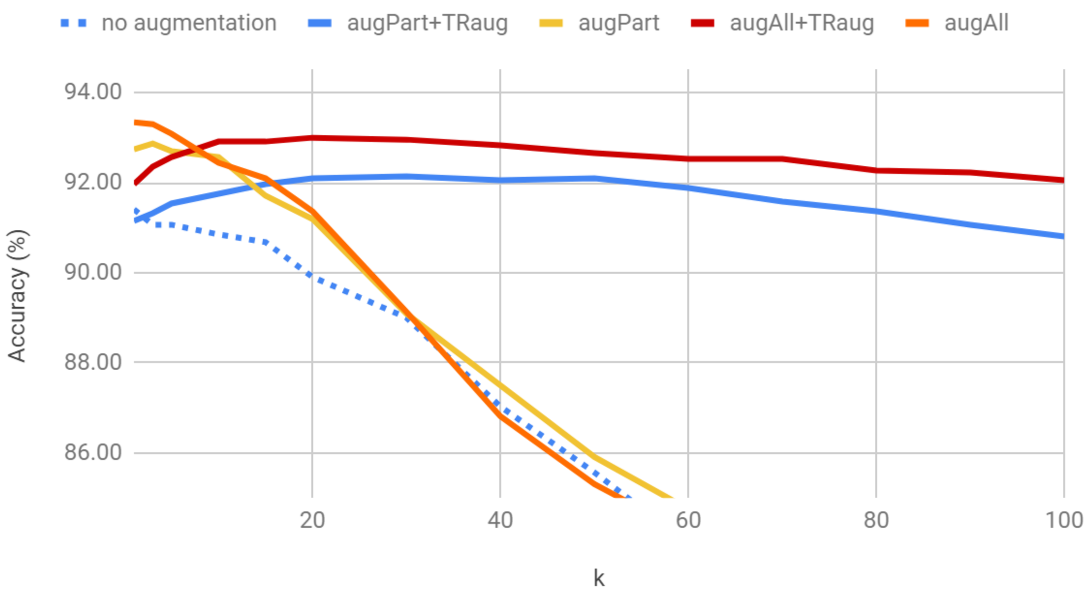


Best results:
no augmentation = 91.41%
augPart+TRaug = 92.14%
augAll+TRaug = 93.00%



Training data augmentation (1) – shift/crop/extension

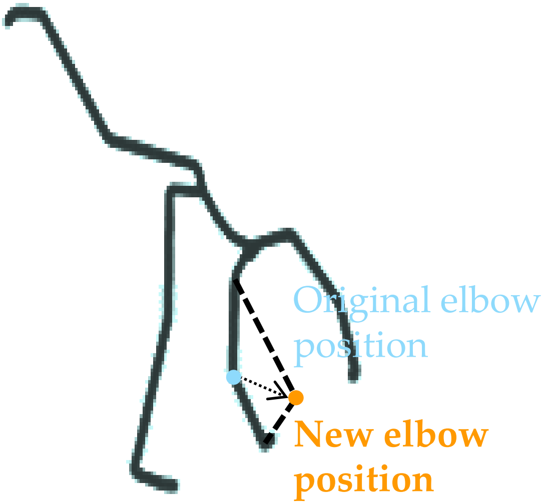
- no augmentation – **1,164** actions for learning; **1,164** reference actions
- augPart – **5,820** actions for learning; **1,164** reference actions
  - Each action in 5 variants: original; 10/20% shift (left + right)
- augAll – **10,476** actions for learning; **1,164** reference actions
  - Each action in 9 variants: original; 10/20% shift (left + right); 10/20% crop and extension



Best results:
no augmentation = 91.41%
augPart+TRaug = 92.14%
augAll+TRaug = 93.00%
augPart = 92.87%
augAll = 93.34%

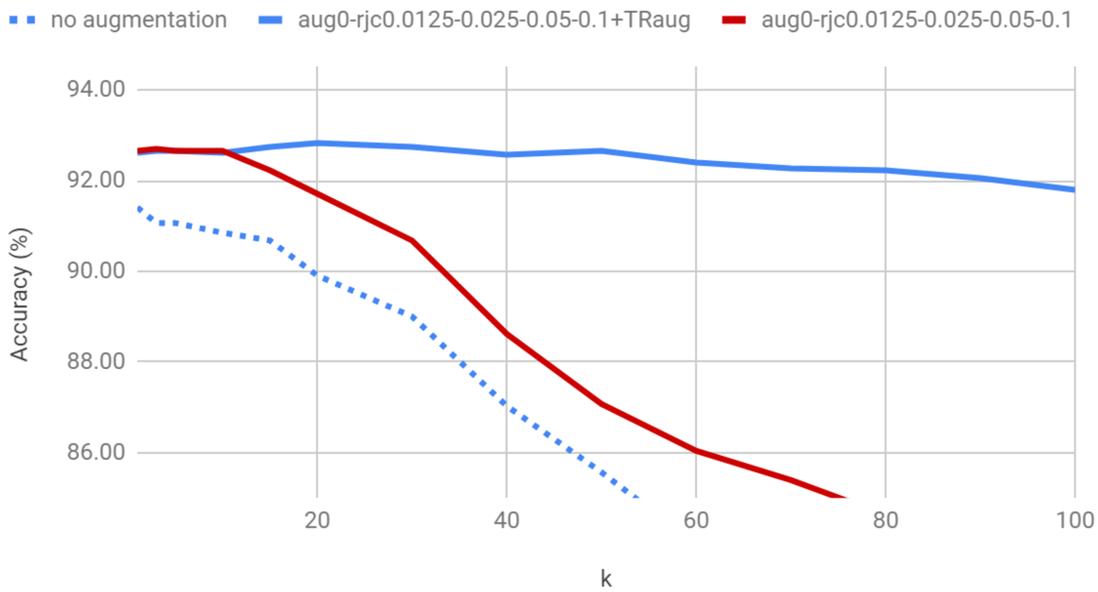
Augmentation (2)

- Random joint coordinate noise – moving each joint coordinate to a new 3D position
  - *MRT* – max relative move threshold (e.g., 5cm)
  - Joint coordinate moved in each of $x/y/z$ axis by a random value from $[0, MRT]$



Training data augment. (2) – noise in joint coords

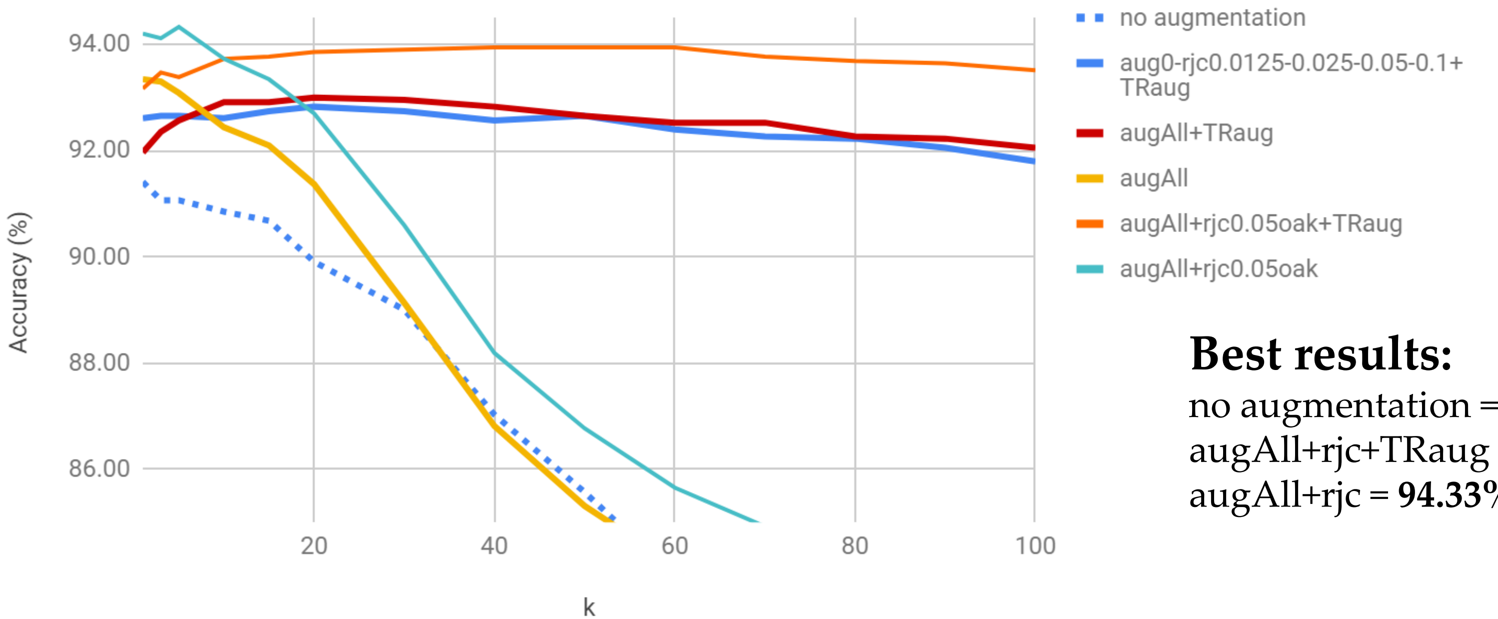
- no augmentation – **1,164** actions for learning; **1,164** reference actions
- aug0-rjc – **5,820** actions for learning; **1,164** reference actions
  - Each action in 5 variants: original; *MRT* of ~ 0.6, 1.3, 2.5 and 5 cm
- aug0-rjc+TRaug – **5,820** actions for learning; **5,820** reference actions
  - Each action in 5 variants: original; *MRT* of ~ 0.6, 1.3, 2.5 and 5 cm



Best results:
no augmentation = 91.41%
aug0-rjc+TRaug = 92.83%
aug0-rjc = 92.70%

Training data augmentation (3) – combination

- no augmentation – **1,164** actions for learning; **1,164** reference actions
- augAll+TRaug – **10,476** actions for learning; **10,476** reference actions
  - Each action in 9 variants: original; 10/20% shift/crop/extension with *MRT* of 2.5 cm
- augAll – **10,476** actions for learning; **1,164** reference actions
  - Each action in 9 variants: original; 10/20% shift/crop/extension with *MRT* of 2.5 cm



Best results:
no augmentation = 91.41%
augAll+rjc+TRaug = 93.94%
augAll+rjc = **94.33%**

State-of-the-art comparison

- HDM05 dataset 2,328/2,345 samples in 122/130 classes
- 2-fold cross validation (50% of training data)

	Method	Accuracy (%)	
		HDM-122	HDM-130
Related approaches	Huang et al. (2016)	N/A	75.78
	Laraba et al. (2017)	N/A	83.33
	Li et al. (2018)	N/A	86.17
	1NN on 4kMI (2017)	87.24	86.79
	1NN on 4kMIE (2017)	87.84	87.38
	Confusion-based 15NN_TCS on 4kMIE (2018)	89.09	88.78
	LSTM features on balanced splits (2019)	**91.41**	
	LSTM features on balanced splits + augmented tr. data (2019)	**94.33**	

Conclusions

Observations

- LSTM features outperform CNN features in both effectiveness and efficiency
  - LSTM features can be parametrized in many ways (e.g., size of feature, size of embedding)
- Splitting training data in a **balanced way** increases the recognition accuracy a lot
  - 88.66% => **91.41%** (decrease in classification error of **24%**)
- **Augmenting** less-populated classes of training data increases the recognition accuracy a lot
  - 91.41% => **94.33%** (decrease in error of **34%** vs. no augmentation)
  - 89.09% => 94.33% (decrease in error of **48%** vs. state-of-the-art result)