Fine-Grained Semantic Segmentation of Motion Capture Data using Dilated Temporal Fully-Convolutional Networks

N. Cheema[1,2,3], S. Hosseini[1,2†], J. Sprenger[1,2], E. Herrmann[1,2], H. Du[1,2], K. Fischer[1,2] and P. Slusallek[1,2]

[1]German Research Center for Artificial Intelligence (DFKI) Saarbrücken, [2]Saarland University, [3]Max-Planck Institute for Informatics; Germany
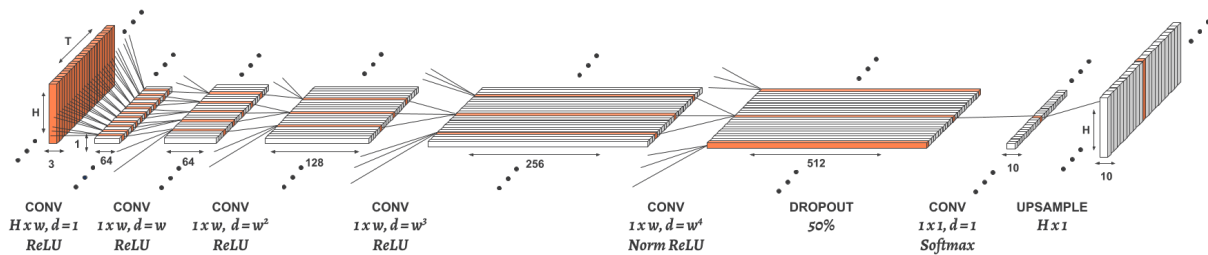

Figure 1: *Our dilated temporal fully-convolutional neural network (DT-FCN) for motion capture segmentation.*

Abstract

Human motion capture data has been widely used in data-driven character animation. In order to generate realistic, natural-looking motions, most data-driven approaches require considerable efforts of pre-processing, including motion segmentation and annotation. Existing (semi-) automatic solutions either require hand-crafted features for motion segmentation or do not produce the semantic annotations required for motion synthesis and building large-scale motion databases. In addition, human labeled annotation data suffers from inter- and intra-labeler inconsistencies by design. We propose a semi-automatic framework for semantic segmentation of motion capture data based on supervised machine learning techniques. It first transforms a motion capture sequence into a "motion image" and applies a convolutional neural network for image segmentation. Dilated temporal convolutions enable the extraction of temporal information from a large receptive field. Our model outperforms two state-of-the-art models for action segmentation, as well as a popular network for sequence modeling. Most of all, our method is very robust under noisy and inaccurate training labels and thus can handle human errors during the labeling process.

CCS Concepts
• Computing methodologies → Motion processing; Motion capture; Image processing;

1. Introduction

Data-driven motion synthesis for digital human models has been widely used to generate realistic and natural human motion. Many data-driven techniques require motion segmentation as a necessary pre-processing step, for instance, statistical modeling [MC12] and graph-based approaches [KGP08]. For semantic-embedded motion synthesis [MC12, DMHF16], recorded motions need to be split in structurally- and semantically-similar segments. An action can then be represented as a finite set of semantic-embedded states (motion primitives). For instance, walking can be decomposed as a combination of left and right steps. To our knowledge there is not extensive work done in motion primitive segmentation (i.e. segmenting

left step from *right step* for example) using recognition-based segmentation methods, therefore most of the mentioned related work focuses on action segmentation (i.e. segmenting *walking* from *picking*). We include work based on 3D motion capture data, as well as, 2D video based data.

Kinematic-based segmentation methods such as [MRC05, MC12] commonly compare hand-crafted, low-level kinematic time-series features (e.g. distance foot to floor [HSK16, MC12]) to find segment boundaries for such motion primitives. Since such features have to be designed separately for each new task, these methods are unfit for segmenting a multitude of different actions. I.e. the distance from foot to floor works for segmenting walking actions but to segment a picking action.

Data-analysis-based segmentation methods [ZDlTH13, VKK14] make use of unsupervised methods to automatically learn

---

† First two authors contributed equally; email: noshaba.cheema@dfki.de

high-level features for segmentation. They have been the conventional method for segmenting skeletal motion data. Although, these approaches produce more sophisticated results and can be used for unseen data, they generally lack control of the feature selection and are not able to produce semantic labels as they cannot make any insights on the content or semantics of the motion data.

Recognition-based segmentation, on the other hand, is based on supervised learning approaches. Typically, in supervised motion segmentation techniques a collection of motion capture data is manually segmented and labeled to train a classifier. Therefore, segments created by the classifier can be as complex as segments created by humans. In recent years, deep learning methods have gained popularity in recognition-based segmentation. Recurrent neural networks (RNN) have become the go-to method to model time-dependent sequences and have also been used for action segmentation using motion capture data [DWW15]. However, one of their major drawbacks is the exploding and vanishing gradient problem and the difficulty to parallelize their training. Recent studies [BKK18] suggest that certain architectures of Convolutional Neural Networks (CNNs) called Temporal Convolutional Networks (TCNs) can reach state-of-the-art results in typical sequence modeling tasks outperforming different types of RNNs. With the Copy Memory Task [BKK18], they show that TCNs exhibit longer memory than recurrent architectures with the same capacity. TCNs have also been used for action segmentation in videos [LFV*17]. The main advantage of recognition-based methods is the enhanced controllability; e.g. unsupervised methods like [ZDITH13] segment a walking sequence to *leftForward*, *leftStep*, *rightForward* and *rightStop* - essentially, dividing a single step into half when using four clusters. Many motion synthesis approaches [MC12] however, want to distinguish between a beginning/ending step and a step during locomotion. We therefore present a supervised segmentation method which is able to learn such semantic motion primitive labels. To account for inter- and intra-annotator disagreements, we further introduce noise into our training labels, as well as mask certain regions out. The presented method outperforms other state-of-the-art and popular recognition-based action segmentation methods.

2. Our Approach

In a preprocessing step, we first transform our motion capture data into an RGB image, much in the spirit of [LBTD17]. Each column of the image represents a frame in the motion sequence. The rows represent the joints and the RGB values are the scaled XYZ Euclidean coordinates of each corresponding joint. Such a *motion image* can be seen in Fig. 2. We then pass it to our network. The network architecture can be seen in Fig. 1. Akin to [LSD15] our model has a total of five convolutional layers. Each layer multiplies the number of filters by two starting from 64 to 512 filters. By setting the kernel height of a 2D convolution in the initial layer to the height of the input image, it performs convolutions only in time domain and is able to process RGB image data. The next four layers are 1D temporal acausal convolutions with dilation. Every layer has the same convolution width w with stride 1. Since motion features can span many frames, the filter needs to be able to look far into the future and into the past. Thus, our network requires a large receptive field. The receptive field of normal CNNs
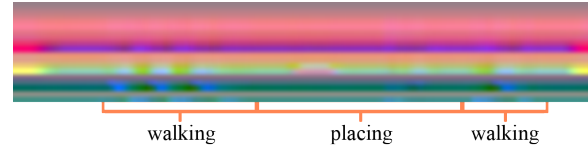

walking   placing   walking

Figure 2: *Sample generated motion image from local XYZ joint position coordinates.*

increase linearly to the depth of the network, thus increasing the amount of layers and parameters to train. Following the work of [VDODZ*16, BKK18], we apply dilated convolutions that enable exponentially large receptive field sizes. The dilation allows the filter to operate on a coarser scale than a normal convolution. For 1D sequence $\mathbf{x} \in \mathbb{R}^N$ and filter $f : \{0, ..., k-1\} \to \mathbb{R}$ such a convolution is formally defined as $(\mathbf{x} *_d f)(t) = \sum_{i=0}^{k-1} f(i) \cdot \mathbf{x}_{s-d \cdot i}$ where d is the dilation and k the filter length. A normal convolution thus can be seen as a special case of a dilated convolution with $d = 1$. Previous work [BKK18, LFV*17, VDODZ*16] suggests that the dilation rate should be set to $d = 2^{l-1}$. However, we find that a rate of $d = w^{l-1}$ (layer number $l = 1 \ldots L$, filter width w) is sufficient enough to ensure that there is at least one parameter hitting each input within the network's receptive field. Due to the increased dilation rate for $w > 2$ a sufficient receptive field size can be achieved with less layers. To ensure that the values created by the ReLU activation layers before the Softmax function, do not exceed reasonable input values, we normalize the output of the last ReLU activation using the following function: $NormReLU(x) = \frac{ReLU(x)}{\max(ReLU(x)) + \varepsilon}$ with $\max(x)$ being the maximum values of the input tensor x. We use a value of $\varepsilon = 1e^{-5}$ and found that this greatly improves accuracy. Finally, we upsample the output to the full image height for visualization purposes. Since our model is fully-convolutional [LSD15] it is able to handle input sequences of variable length.

3. Evaluation

Our model's parameters are learned using the categorical cross entropy loss with Stochastic Gradient Descent and the Adam optimizer. We implemented the network using Keras with a Tensorflow back-end. In all of our experiments, we use non-randomized 7-fold cross-validation and 100 training epochs. Our motion capture data set for evaluation consists of 70 sequences in total, of which 60 are used for training and 10 for testing. The data set contains standing, walking, picking, placing, and turning actions. 19 joints were used. Table 1 shows the 10 motion primitive labels used for segmentation and their total number of frames in the data set. The sequences are between 13-22 seconds long at 72 FPS, captured with OptiTrack™. The segmentation was done by human experts.

In order to determine the optimal receptive field size (RFS), we test our Dilated Temporal Fully-Convolutional Network (DT-FCN) on different convolution kernel widths w. Fig. 3 shows that even though a width $w = 5$ (RFS: 3125 frames) covers the longest sequence in the dataset, the test accuracy is worse on all noise levels compared to using $w = 3$ (RFS: 243 frames). This is in contrast to popular claims [BKK18, VDODZ*16] in sequence modeling tasks that the receptive field of the network should at least cover the longest sequence in the data set, and supports a more intuitive suggestion that local structures are enough to be able to clas-

Motion Primitive	Description	Color	#Frames
standing	standing in T or I pose	■	29990
begin left step	left step from standing	■	2173
begin right step	right step from standing	■	6509
left step	left step in locomotion	■	8958
right step	right step in locomotion	■	7201
end left step	left step to standing	■	4763
end right step	right step to standing	■	4797
reach	reach out with arm	■	10659
retrieve	retrieve with arm	■	6813
turn (in standing)	turn body direction	■	6630

Table 1: *Motion primitive labels with their corresponding label color in the segmented images and number of frames in the dataset.*

sify important structural and temporal information. Furthermore, the increase in parameter space for $w = 5$ make it also more prone to over-fitting, as compared to $w = 3$. Since our model has to be robust against human error due to wrongly-classified labels, we further train our model on noisy labels by setting a certain percentage of labels to a random label and test it on the true labels. Fig. 4 shows such noisy labels for the training. Fig. 3 shows despite adding 80% noisy labels in the training data, an accuracy of almost 90% is reached on the true test labels for $w = 3$. Qualitative results for $w = 3$ can be seen in Fig. 4. We compare our best model ($w = 3$) against two other state-of-the-art TCN-based models [LVRH16, LFV*17] for action segmentation and a commonly used RNN [GS05] for sequence modeling using our data set without noisy labels. All TCN-based methods are able to train magnitudes faster compared to the RNN-based method, due to the "embarrassingly parallel" nature of convolutions. Training takes ∼1 minute for the TCN-based methods compared to ∼40 minutes for Bi-LSTM for 100 epochs on a 4GB GTX970 and 16GB Intel-i7. Segmentation takes less than ∼1 second for all methods. Qualitatively, our model shows a better performance than the state-of-the-art (Tab. 2, 3, Fig. 6).

Discrepancies between human raters are mostly in the boundary regions between two adjacent segments. Hence, we conduct an experiment where a window of width $w_{transition}$ at a boundary frame contains randomly set, and hence wrong, training labels, as seen in Fig. 7 (top of each row). Additionally, we conduct an experiment which is less "harsh" in nature: Instead of giving wrong information at the boundaries, we provide no information by setting the loss to zero within these regions. Essentially, leaving the networks to their own interpretation. The results of both experiments can be seen in Tab. 4 and Fig. 7. Expectedly, results are better without any information than with wrong information. Most interestingly, masking the boundary frames (e.g. 11 frames, Tab. 4) improves the overall model performance compared to the original data (Tab. 2). This might be due to inter- and intra-annotator disagreements: When there is no information given in these regions, the network is able to learn the relevant features from the geometric information alone. On average, this can be more accurate on unseen examples than learning from "human generated" annotations, which tend to be different from annotator to annotator (or even from using the same annotator) in boundary regions.
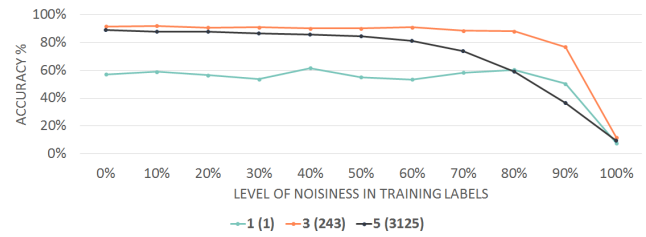

Figure 3: *Test accuracies for different receptive field sizes of our proposed architecture depending on training label noise level.*
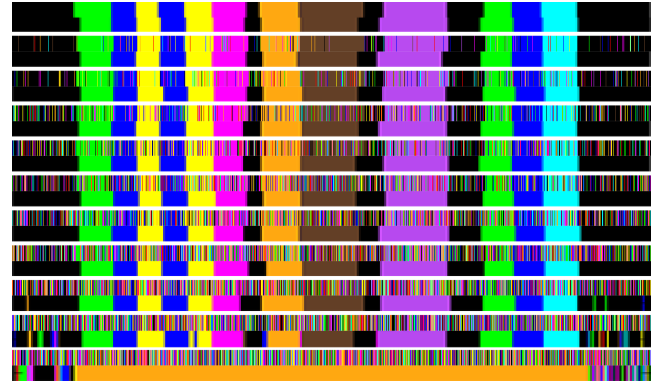

Figure 4: *Segmentation results of example training sequence of our $w = 3$ model. Noisy training label is on top of corresponding prediction. Top row without any noise. Noise added in +10% intervals.*

	Bi-LSTM [GS05]	ED-TCN [LVRH16]	DilatedTCN [LFV*17]	Ours ($w = 3$)
Train	86.32%	90.05%	90.64%	**95.42%**
Test	81.95%	88.69%	88.47%	**90.81%**
# Param.	378,634	1,613,770	388,874	663,562
RFS	∞	49	254	243
Causality	Acausal	Acausal	Causal	Acausal

Table 2: *Per-frame accuracy for benchmark models and our best model ($w = 3$) on the dataset without any noise in the train labels.*

Figure 5: *Segmentation results of benchmark models, as well as our best model ($w = 3$) on an example test sequence. Each row shows ground truth (top half) and the corresponding prediction (bottom half).*

Figure 6: *Segmentation results for training on 90% label noise.*

Noise in Training	Bi-LSTM [GS05]	ED-TCN [LVRH16]	DilatedTCN [LFV*17]	Ours (w = 3)
0%	81.95%	88.69%	88.47%	**90.81%**
10%	83.70%	89.16%	89.37%	**90.99%**
20%	79.94%	89.46%	88.86%	**90.75%**
30%	85.02%	89.40%	88.90%	**91.20%**
40%	84.29%	89.02%	88.64%	**91.13%**
50%	81.72%	87.40%	87.93%	**91.04%**
60%	83.08%	88.65%	86.14%	**92.99%**
70%	82.16%	87.66%	79.46%	**90.11%**
80%	81.91%	80.46%	61.82%	**89.55%**
90%	71.74%	54.91%	34.99%	**81.18%**
100%	8.76%	10.39%	11.85%	11.24%

Table 3: *Per-frame test accuracy for benchmark models and our best model (w = 3) with noise added into the training labels.*

$w_{transition}$	Bi-LSTM [GS05]	ED-TCN [LVRH16]	DilatedTCN [LFV*17]	Ours (w = 3)
11 frames	84.57%	88.15%	87.50%	**91.45%**
21 frames	81.47%	83.82%	85.48%	**89.03%**
31 frames	78.94%	75.75%	80.62%	**83.59%**
11 frames	81.02%	88.25%	89.04%	**92.42%**
21 frames	80.05%	85.88%	86.80%	**91.93%**
31 frames	76.71%	84.88%	86.01%	**90.48%**

Table 4: *Per-frame test accuracies for benchmark models and our best model (w = 3) with noise in the transition regions (top three rows) and with masked boundary regions (bottom three rows).*

4. Conclusion

In this paper we present a first dilated temporal FCN-based method for fine-grained semantic segmentation of motion capture sequences. Compared to commonly used unsupervised methods [ZDlTH13] our approach is able to learn complex labels such as "begin left step" or "end right step", while robustly handling labeling errors. While the key-ingredients for the success were already present in prior work [LBTD17, LSD15, BKK18], we believe it is the combination of these methods which accounts for the improved accuracy compared to other TCN-based methods. The combination of a VGG/FCN32-inspired model with acausal dilated convolutions and an increased dilation rate compared to other methods [LFV*17, VDODZ*16] enables a large receptive field with few parameters. Thus training time is reduced while the performance of the model is above state-of-the art competitors. It can even distinguish very similar motion primitives like *start left step* and *left*
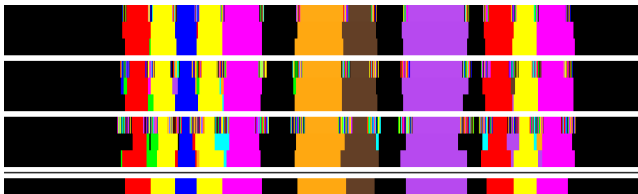

Figure 7: *Predicted segmentations by our model (w = 3). Blocks from top to bottom: $w_{transition} = 11$, $w_{transition} = 21$, $w_{transition} = 31$. Each block consists of the noisy training label (top), the corresponding prediction without masking out the noise (middle) and with masking out (bottom).*

step. Most of all, the model is very robust under label noise. Hence even cheaply labelled data can be used for training. Even more, we found that a semi-supervised approach by removal of the boundary labels between classes can improve the performance further. With this work, we have shown that our model provides a fruitful segmentation tool for motion capture segmentation.

References

[BKK18] BAI S., KOLTER J. Z., KOLTUN V.: An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271* (2018). 2, 4

[DMHF16] DU H., MANNS M., HERRMANN E., FISCHER K.: Joint angle data representation for data driven human motion synthesis. *Procedia CIRP 41* (2016), 746–751. 1

[DWW15] DU Y., WANG W., WANG L.: Hierarchical recurrent neural network for skeleton based action recognition. In *Proceedings of the IEEE CVPR* (2015), pp. 1110–1118. 2

[GS05] GRAVES A., SCHMIDHUBER J.: Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks 18*, 5-6 (2005), 602–610. 3, 4

[HSK16] HOLDEN D., SAITO J., KOMURA T.: A deep learning framework for character motion synthesis and editing. *ACM Transactions on Graphics (TOG) 35*, 4 (2016), 138. 1

[KGP08] KOVAR L., GLEICHER M., PIGHIN F.: Motion graphs. In *ACM SIGGRAPH 2008 classes* (2008), ACM, p. 51. 1

[LBTD17] 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). 2, 4

[LFV*17] LEA C., FLYNN M. D., VIDAL R., REITER A., HAGER G. D.: Temporal convolutional networks for action segmentation and detection. *2017 IEEE CVPR* (2017), 1003–1012. 2, 3, 4

[LSD15] LONG J., SHELHAMER E., DARRELL T.: Fully convolutional networks for semantic segmentation. In *Proceedings of IEEE CVPR* (2015), pp. 3431–3440. 2, 4

[LVRH16] LEA C., VIDAL R., REITER A., HAGER G. D.: Temporal convolutional networks: A unified approach to action segmentation. In *ECCV* (2016), Springer, pp. 47–54. 3, 4

[MC12] MIN J., CHAI J.: Motion graphs++: A compact generative model for semantic motion analysis and synthesis. *In ACM Transactions on Graphics 31*, 6 (2012), 153:1–153:12. 1, 2

[MRC05] MÜLLER M., RÖDER T., CLAUSEN M.: Efficient content-based retrieval of motion capture data. In *ACM Transactions on Graphics (ToG)* (2005), vol. 24, ACM, pp. 677–685. 1

[VDODZ*16] VAN DEN OORD A., DIELEMAN S., ZEN H., SIMONYAN K., ET AL.: Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499* (2016). 2, 4

[VKK14] VÖGELE A., KRÜGER B., KLEIN R.: Efficient unsupervised temporal segmentation of human motion. In *Proceedings of ACM SIGGRAPH/Eurographics SCA* (2014), pp. 167–176. 1

[ZDlTH13] ZHOU F., DE LA TORRE F., HODGINS J. K.: Hierarchical aligned cluster analysis for temporal clustering of human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence 35*, 3 (2013), 582–596. 1, 2, 4