HumMUSS: Human Motion Understanding using State Space Models

Supplementary Material

Arnab Mondal
Mila & Apple
arnab.mondal@mila.quebec

Stefano Alletto
Apple
salletto@apple.com

Denis Tome
Apple
d_tome@apple.com

1. Implementation Details

Our pretraining approach builds on top of open-sourced implementation of MotionBERT [4] [1] . We implement our model using PyTorch and adapt the data-preprocessing pipeline from MotionBERT for both pretraining and finetuning phases. Detailed information about this pipeline can be found in [4]. The experiments were conducted on a linux machine equipped with 4 NVIDIA A100 80GB GPUs. However, for finetuning experiments, a single A100 80GB GPU proved to be sufficient.

HumMUSS is designed with 5 spatiotemporal blocks, that is $N = 5$. Given that we work with 2D joints and their confidences, the input dimensionality is set to $D_{in} = 3$. Our implementation utilizes a $16M$ parameter model for HumMUSS, with a model dimension (D_m) of 256.

In our approach, the hyperparameters for GDSSM Blocks are adjusted to make both the causal and unidirectional model consist of $16M$ parameters. Specifically, we use $k = 1$ for the spatial GDSSM block and $k = 2$ for the temporal GDSSM block. For the causal HumMUSS model, we set $m = 3$ for both the spatial and temporal blocks, while for the bidirectional model, $m = 2.5$ is used. Furthermore, the dimension of the state space (N) is set to 128 for both variants of HumMUSS, and the representation dimension (D_{rep}) is 512.

Apart from the learning rate, the other training hyperparameters for both the pretraining and finetuning tasks are kept consistent with those used in the MotionBERT model [4], ensuring a fair comparison with the transformer-based baseline. We use a learning rate of 0.0003 for the pretraining. We use similar learning rates as [4] for finetuning tasks.

2. Additional experiments

In this section, we delve deeper with additional experiments to explore the impact of architectural choices on HumMUSS. These investigations are specifically conducted within the domain of 3D Pose Estimation, utilizing the MPI-INF-3DHP dataset [3]. This task choice aligns well with our primary goal of pretraining objective to learn motion representations.

2.1. Generalization to longer context window

We show the ability of HumMUSS's architecture to generalize to longer sequence lengths in this experiment. The model, initially trained on a sequence length of F=27 frames, was tested on longer sequences of F=81, and F=243. As shown in Table 2, the HumMUSS model exhibits a consistency in performance when tested on longer sequences. This means HumMUSS can generalize to longer sequences than the one it is trained while being able to encode the longer context. This proves especially beneficial for deployment in tasks necessitating a longer contextual understanding than what was used during the model's training.

Training sequence length	Test sequence length		
F=27	F=27	F=81	F=243
HumMUSS	25.1	25.8	26.2

Table 1. 3D Pose Estimation performance (MPJPE in mm) of a HumMUSS model trained on 27 frame and tested on longer sequences.

2.2. Ablation study

An ablation study was conducted to evaluate the impact of different aggregation methods in the GDSSM (Generalized Dynamic Spatiotemporal Sequence Modeling) blocks of the HumMUSS model. In particular, we consider aggregation of the different information processing pathways in GDSSM blocks. In Section 4.1 and 4.2, we use Multiplicative gating to combine x_f, x_b and x_id. In this ablation, we also test other aggregation methods including average pooling and pooling with learnable weights. Notably, the pooling with learnable weights technique is the same as the one presented in Section 4.3 the results, as illustrated in

[1] https://github.com/Walter0807/MotionBERT

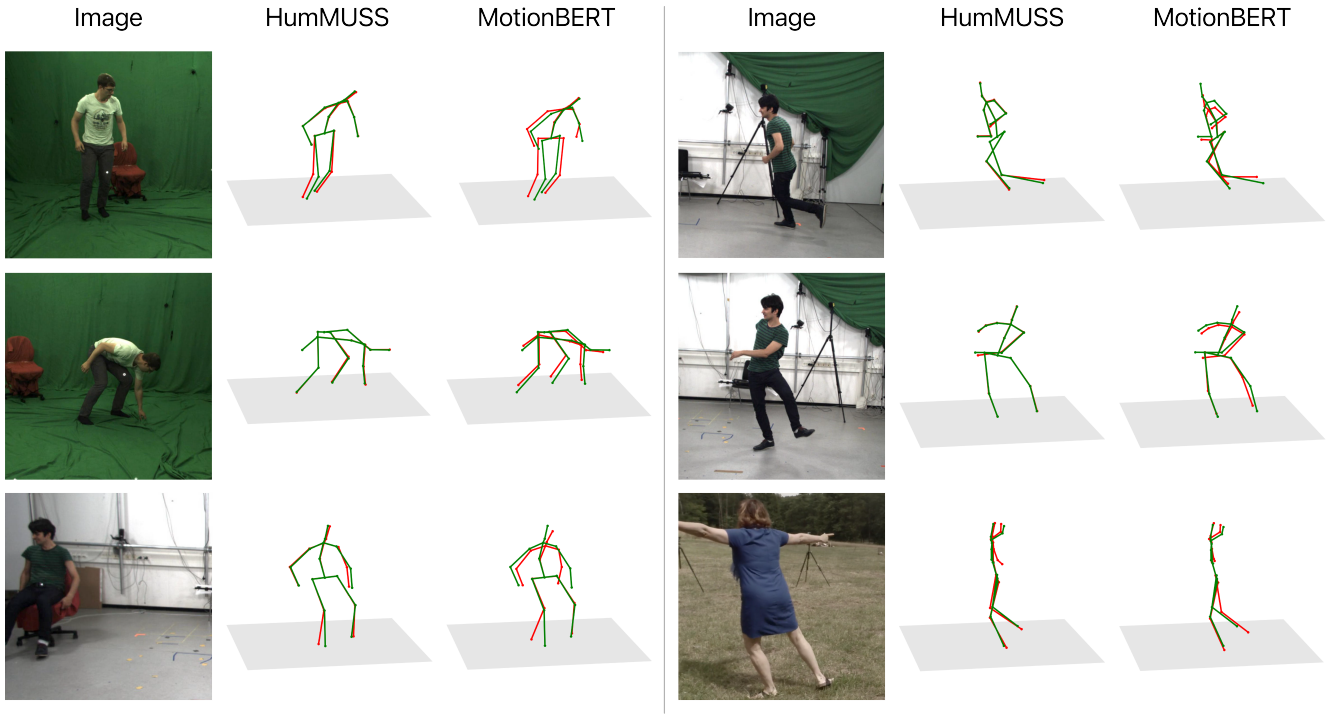

Figure 1. Example frames of 3d reconstruction for both HumMUSS and MotionBERT [4] when the input signal is sub-sampled by a factor of 8. Green and red poses represent *ground truth* and *prediction* respectively. Best viewed in color.

Table 3, show that Multiplicative Gating outperforms other methods in both causal and bidirectional model types, with the lowest MPJPE of 40.2 mm in the bidirectional setting and 45.1mm in the causal setting. This indicates the effectiveness of Multiplicative Gating in aggregating information within GDSSM blocks.

Type of aggregation in GDSSM Blocks	Model type	
	Causal	Bidirection
Average Pool	25.8	20.5
Pooling with learnable weights	25.3	20.1
Multiplicative Gating	24.6	18.7

Table 2. 3D Pose Estimation performance (MPJPE in mm) of HumMUSS using different aggregation methods to combine information in the GDSSM blocks.

Further, the performance of HumMUSS using different aggregation methods to combine two streams of Spatiotemporal GDSSMs was evaluated. The methods included Multiplicative Gating, Average Pooling, and Pooling with Learnable Weights. The findings, presented in Table 4, indicate that Pooling with Learnable Weights provides the best performance in both causal and bidirectional settings. This highlights the effectiveness of learnable weights in combining multiple streams of spatiotemporal data. Surprisingly, multiplicative gating performs poorly for combining two streams which warrants further investigation in the future.

Type of aggregation to combine two streams	Model type	
	Causal	Bidirection
Multiplicative Gating	26.6	21.1
Average Pool	25.1	19.5
Pooling with learnable weights	24.6	18.7

Table 3. 3D Pose Estimation performance (MPJPE in mm) of HumMUSS using different aggregation methods to combine the two streams of Spatiotemporal GDSSMs.

3. Initialization of DSSM parameters

A simple way to initialize the diagonal parameters is to use the linear initialization which sets the real part Λ_{re} to $-\frac{1}{2}\mathbb{1}_N$ and the imaginary part Λ_{im} to $(\pi j)_{j=1}^{N}$ [2]. Gu *et al.* [2] provides many other initialization techniques and empirically verify that they can approximate the initialization of the S4 [1] kernel which leads to stable training regime for longer sequences. In practice, we found all the techniques leads to similar performance and use the simple linear initialization scheme throughout our experiments. Finally, elements of C are samples from a normal distribution and Δ is initialized randomly between 0.001 and 0.1.

4. Generalization to unseen frame rate

Transformers, being discrete models, are inherently designed for sequences that have fixed uniform spacing in time and utilize positional encodings to encode temporal information. MotionBERT, in particular, exacerbates this issue by employing learnable positional encodings, which lack a clear sense of time. In contrast, HumMUSS being a continuous time model, does not require positional encodings and enables training and testing on data with different frame rates. Unlike Transformers, which struggle with down-sampled or up-sampled videos, HumMUSS seamlessly adapts. For instance, if trained on 30 FPS data, changing the discretization parameter Δ to 2Δ or $\frac{1}{2}\Delta$ allows the model to perform effectively on 15 or 60 FPS videos respectively, without the need for expensive re-training.

In practice, sampling rate changes can directly be measured by e.g. reading camera timestamps to detect frame drops or frame-rate changes due to external factors. As a continuous time model, HumMUSS interprets sequences as signal values at different timestamps. Adjustment the discretization parameter allows HumMUSS to process it as changes in the number of samples, rather than a different input signal. Furthermore, in sequential inference scenarios with varying sampling rate, we keep HumMUSS robust by dynamically scaling Δ based on the evolving sampling rate. This is possible because HumMUSS maintains a compressed memory of the past observed signal in its state that is independent of past sampling frequencies. Such adaptability positions HumMUSS as a potent model for real-time applications where input frame rates may vary due to factors like thermal throttling of capturing devices.

References

[1] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021. 2

[2] Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the parameterization and initialization of diagonal state space models. *Advances in Neural Information Processing Systems*, 35:35971–35983, 2022. 2

[3] Dushyant Mehta, Helge Rhodin, Dan Casas, Pascal Fua, Oleksandr Sotnychenko, Weipeng Xu, and Christian Theobalt. Monocular 3d human pose estimation in the wild using improved cnn supervision. In *2017 international conference on 3D vision (3DV)*, pages 506–516. IEEE, 2017. 1

[4] Wentao Zhu, Xiaoxuan Ma, Zhaoyang Liu, Libin Liu, Wayne Wu, and Yizhou Wang. Motionbert: A unified perspective on learning human motion representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15085–15099, 2023. 1, 2