

ScaMo: Exploring the Scaling Law in Autoregressive Motion Generation Model

Supplementary Material

Contents

A Flops calculation	2
B Implmentation Details	2
C Tokenizer Results	3
D More Results on HumnaML3D Benchmark	3
E Dataset Visualization	5
F. FSQ settings	5
G More Generation Visualizations	6
H Limitations	7

A. Flops calculation

In this paper, we follow the criteria proposed by OpenAI [5] to calculate the floating point operations (FLOPs). The detailed breakdown of these computations is provided in Tab. 1. Additionally, to facilitate reproducibility, we present the complete code for this calculation process in Code 1.

By utilizing the provided code, we can quickly compute the FLOPs for their specific model configurations, facilitating performance analysis and design optimization.

Operation	Parameters	FLOPs per Token
Embed	$(n_{\text{vocab}} + n_{\text{ctx}})d_{\text{model}}$	$4d_{\text{model}}$
Attention: Q, K, V	$n_{\text{layer}} \times d_{\text{model}} \times 3d_{\text{attn}}$	$2n_{\text{layer}} \times d_{\text{model}} \times 3d_{\text{attn}}$
Attention: Mask	–	$2n_{\text{layer}} \times n_{\text{ctx}} \times d_{\text{attn}}$
Attention: Projection	$n_{\text{layer}} \times d_{\text{attn}} \times d_{\text{model}}$	$2n_{\text{layer}} \times d_{\text{attn}} \times d_{\text{model}}$
Feedforward	$n_{\text{layer}} \times 2d_{\text{model}} \times d_{\text{ff}}$	$2n_{\text{layer}} \times 2d_{\text{ff}}$
De-embed	–	$2d_{\text{model}} \times n_{\text{vocab}}$
Total (Non-Embedding)	$N = 2d_{\text{model}} \times n_{\text{layer}}(2d_{\text{attn}} + d_{\text{ff}})$	$C_{\text{forward}} = 2N + 2n_{\text{layer}} \times n_{\text{ctx}} \times d_{\text{attn}}$

Table 1. Details of FLOPs calculation criteria by OpenAI [5].

```

1 def openai_flops_per_token(n_layers, n_heads, d_model, n_ctx, n_vocab, ff_ratio=4):
2     """Open AI method for forward pass FLOPs counting of decoder-only Transformer
3     """
4     d_attn = d_model // n_heads
5     d_ff = d_model * ff_ratio
6
7     embeddings = 4 * d_model
8     attn_qkv = 2 * n_layers * d_model * 3 * (d_attn * n_heads)
9     attn_mask = 2 * n_layers * n_ctx * (d_attn * n_heads)
10    attn_project = 2 * n_layers * (d_attn * n_heads) * d_model
11    ff = 2 * n_layers * 2 * d_model * d_ff
12    logits = 2 * d_model * n_vocab
13
14    return embeddings + attn_qkv + attn_mask + attn_project + ff + logits

```

Code 1. Open AI method for forward pass FLOPs counting of decoder-only Transformer.

B. Implementation Details

The framework is implemented using PyTorch. For the Motion FSQ-VAE, both the encoders and decoders are designed as convolutional residual blocks, utilizing a downsampling factor of 4. The transformer architecture closely aligns with that of LLaMA. Specifically, each block incorporates RMSNorm prior to both the prefix attention layer and the feed-forward network (FFN) layer. We train the transformers using bf16 to reduce the memory. We do not use the masking strategy in [8]. The optimization details are shown in Tab. 2.

Config	Tokenizer	Transformer
optimizer	AdamW	AdamW
optimizer momentum	0.9	0.9
weight decay	0.0	1e-06
learning rate schedule	MultiStepLR	Warmup and Cosine decay
milestone_ratio	0.6	-
warmup ratio	0.003	0.1

Table 2. The optimizer details.

For motion representation, we follow HumanML3D [8]. **HumanML3D Format** proposes a motion representation $x^{1:L}$ inspired by motion features in character control. This redundant representation is quite suited to neural models, particularly variational autoencoders. Specifically, the i -th pose x^i is defined by a tuple of root angular velocity $r^a \in \mathbb{R}$ along Y-axis, root linear velocities ($r^x, r^z \in \mathbb{R}$) on XZ-plane, root height $r^y \in \mathbb{R}$, local joints positions $j^p \in \mathbb{R}^{3N_j}$, velocities $j^v \in \mathbb{R}^{3N_j}$ and rotations $j^r \in \mathbb{R}^{6N_j}$ in root space, and binary foot-ground contact features $c^f \in \mathbb{R}^4$ by thresholding the heel and toe joint velocities, where N_j denotes the joint number, giving:

$$x^i = \{r^a, r^x, r^z, r^y, j^p, j^v, j^r, c^f\}.$$

C. Tokenizer Results

We show the numerical results of different tokenizers here. The superior performance of FSQ in terms of reconstruction accuracy, codebook utilization, and code distribution uniformity positions it as a more robust and scalable alternative than VQ. This advantage is particularly beneficial in scenarios that require high-capacity encoding, such as large-scale motion data, where effective codebook utilization and precise reconstruction are paramount.

VQ	L1 loss	FID	MPJPE	Activate	Entropy
256	0.071	0.12	0.05	1.00	170.58
512	0.065	0.10	0.05	1.0	364.45
1024	0.062	0.09	0.05	0.99	704.70
2048	0.060	0.08	0.05	0.97	1145.77
4096	0.078	0.78	0.09	0.69	147.96
8192	0.083	21.8	0.17	0.81	82.20
32768	0.076	5.05	0.11	0.63	1029.20

(a) VQ results on HumanML3D.

VQ	L1 loss	MPJPE	Activate	Entropy
256	0.050	49.70	1.00	177.65
512	0.047	47.90	1.00	404.09
1024	0.045	45.90	0.998	752.028
2048	0.044	60.10	0.996	1202.23
4096	0.044	42.40	0.994	3373.35
8192	0.045	43.53	0.998	6714.25
16384	0.077	62.15	0.993	12732.29
32768	0.054	48.50	0.962	22286.32

(b) VQ results on MotionUnion.

FSQ	L1 loss	FID	MPJPE	Activate	Entropy
256	0.081	0.159	0.057	1.00	213.20
512	0.075	0.129	0.053	1.00	446.10
1024	0.0713	0.106	0.052	1.00	723.80
4096	0.064	0.088	0.049	0.998	2759.52
16384	0.053	0.052	0.044	0.976	10119.25
65536	0.051	0.049	0.042	0.764	33818.21

(c) FSQ results on HumanML3D.

FSQ	L1 loss	MPJPE	Activate	Entropy
256	0.049	47.30	1.00	220.26
512	0.046	44.66	1.00	441.05
1024	0.045	43.40	1.00	853.64
2048	0.042	41.57	1.00	1572.82
4096	0.041	40.66	1.00	3561.95
16384	0.037	37.94	0.999	10974.16
65536	0.034	36.60	0.999	40818.21

(d) FSQ results on MotionUnion.

Table 3. Tokenizer numerical results. The Entropy is Exponential Entropy.

D. More Results on HumnaML3D Benchmark

We train all models on our proprietary dataset, MotionUnion, and evaluate their performance on the HumanML3D benchmark. The numerical results are presented in Tab. 4. Notably, we observe that the model configured with the largest codebook size and model capacity achieves the best overall performance, consistent with the lowest normalized test loss. However, when examining cases of overfitting—such as the combination of a small codebook size (e.g., 256) and a large model size (44M parameters)—the automatic metrics continue to improve, despite being inconsistent with the normalized loss. A similar phenomenon is observed when training T2M-GPT [8]. We hypothesize that this discrepancy arises from the suboptimal performance of the pretrained feature extractor. Additionally, our findings suggest that larger codebook sizes necessitate proportionally larger model capacities to fully leverage their potential.

Furthermore, we conduct a comparative analysis against other frameworks that directly fine-tune large language models (LLMs), such as those proposed in [2, 6, 7, 9]. Our approach demonstrates competitive results on semantic alignment metrics, including R@1, R@3, and Matching Score. Notably, our model achieves superior performance in terms of the FID, highlighting the advantages of our motion tokenizer and architectural design. These results indicate that training a native motion generation model from scratch offers substantial benefits compared to fine-tuning an LLM. Specifically, this approach not only improves performance but also achieves significant parameter efficiency.

Model	Model Size	FID↓	R@1↑	R@2↑	R@3↑	Matching Score↓	Diversity
MotionGPT [2]*	Llama-1-13B	0.592	0.363	-	0.633	4.029	-
MotionGPT [2]*	Llama-2-13B	0.571	0.367	-	0.654	3.981	-
MotionLLM [7]	Gemma-2b	0.491	0.482	-	0.770	3.138	-
AvatarGPT [1]	Llama-1-13B	0.567	0.389	-	0.623	-	-
LargeMotionModel [6]	Llama-2-13B	0.166	0.519	-	0.803	2.964	-
Codebook size	Model Size	FID↓	R@1↑	R@2↑	R@3↑	Matching Score↓	Diversity
256	44M	3.184	0.302	0.45	0.547	4.557	8.317
256	111M	1.197	0.398	0.565	0.667	3.726	8.968
256	343M	0.730	0.432	0.618	0.719	3.466	8.972
256	775M	0.704	0.434	0.617	0.722	3.428	9.393
256	1B	0.709	0.441	0.626	0.723	3.424	9.123
256	3B	0.670	0.443	0.627	0.726	3.410	8.738
512	44M	3.971	0.271	0.402	0.498	4.981	8.792
512	111M	1.338	0.373	0.550	0.660	3.741	8.567
512	343M	0.851	0.415	0.590	0.695	3.514	9.226
512	775M	0.664	0.441	0.619	0.727	3.361	9.187
512	1B	0.624	0.447	0.631	0.734	3.330	8.948
512	3B	0.617	0.443	0.627	0.734	3.340	9.217
1024	44M	8.111	0.216	0.332	0.415	5.766	7.614
1024	111M	1.331	0.371	0.535	0.647	3.865	9.118
1024	343M	0.815	0.422	0.601	0.705	3.525	9.404
1024	775M	0.583	0.447	0.635	0.735	3.300	9.489
1024	1B	0.488	0.453	0.650	0.745	3.290	9.136
1024	3B	0.496	0.453	0.643	0.741	3.296	9.376
2048	44M	13.964	0.192	0.298	0.372	6.295	6.548
2048	111M	1.553	0.361	0.528	0.640	3.857	9.11
2048	343M	0.794	0.418	0.604	0.707	3.490	9.136
2048	775M	0.465	0.450	0.636	0.736	3.300	9.241
2048	1B	0.320	0.454	0.640	0.740	3.264	9.836
2048	3B	0.346	0.465	0.656	0.752	3.216	9.277
4096	44M	18.311	0.131	0.217	0.276	7.077	6.043
4096	111M	1.465	0.327	0.492	0.599	4.134	8.542
4096	343M	0.568	0.422	0.587	0.689	3.467	9.174
4096	775M	0.240	0.464	0.650	0.750	3.250	9.393
4096	1B	0.208	0.486	0.672	0.771	3.120	9.564
4096	3B	0.214	0.483	0.674	0.764	3.128	9.455
16384	44M	44.240	0.056	0.103	0.153	8.019	2.842
16384	111M	4.714	0.254	0.395	0.496	4.891	8.030
16384	343M	1.217	0.380	0.556	0.661	3.711	8.838
16384	775M	0.501	0.443	0.625	0.723	3.370	9.342
16384	1B	0.347	0.477	0.657	0.758	3.206	9.727
16384	3B	0.331	0.469	0.670	0.761	3.192	9.310
65536	44M	50.796	0.041	0.0791	0.1185	8.203	1.490
65536	111M	2.178	0.311	0.461	0.566	4.286	5.311
65536	343M	0.104	0.510	0.692	0.781	3.021	9.540
65536	775M	0.150	0.495	0.685	0.785	3.080	9.558
65536	1B	0.131	0.503	0.687	0.779	3.070	9.580
65536	3B	0.101	0.512	0.695	0.796	2.990	9.590

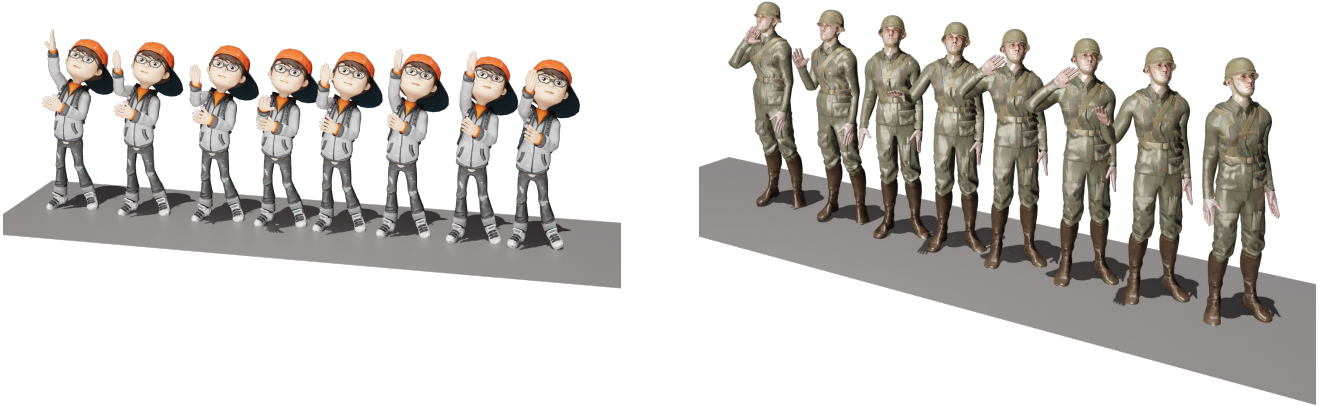
Table 4. Test results of different models on HumanML3D Benchmark. We take the results of MotionGPT* from Wang *et al.* [6].

E. Dataset Visualization

We show motion visualizations and text annotations of MotionUnion in Fig. 1. Render videos can be found in the supplementary materials. The specific frames and sequences are shown in the Tab. 5. PhysHumanML3D subset is the physics-optimized version of HumanML3D using HPC [3].

	Frames	Seqs
PhysHumanML3D	5770156	22628
Animation	55282	559
Combatmotion	3368986	26097
EgoBody	437976	980
Fitness	106537	262
Game Motion	797824	3296
Haa500	438733	6944
HumanML3D	4117392	29228
Humman	187580	971
Idea400	2108727	12042
Kungfu	311507	1032
Music	914642	3394
Perform	327903	923
100 Style	4018110	16074
Internal Data	3905243	23067

Table 5. The detailed quantities of frames and sequences within the MotionUnion dataset.



The person is simulating kite flying indoors. They use their arms to mimic holding and controlling a kite string, periodically adjusting their grip and pulling on the imaginary string, while shifting their weight and standing in place to maintain balance.

The person is standing and saluting. The motion involves raising their right arm to the forehead in a salute gesture and then lowering it back to the starting position by the side of the body.

Figure 1. MotionUnion visualization

F. FSQ settings

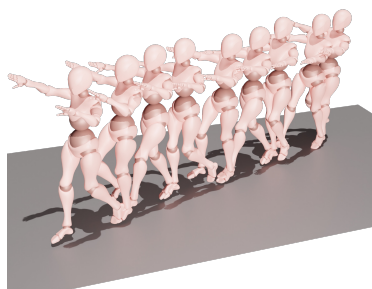
We follow Mentzer *et al.* [4] to set the L in Tab. 6. The codebook size can be calculated as $|\mathcal{C}| = \prod_{i=1}^d L_i$. For example, $2^{10} \approx 8 * 5 * 5 * 5 = 1000$.

Target size $ \mathcal{C} $	2^4	2^6	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{14}	2^{16}
Quantized integer layers L	[5, 3]	[8, 8]	[8, 6, 5]	[8, 8, 8]	[8, 5, 5, 5]	[8, 8, 6, 5]	[7, 5, 5, 5, 5]	[8, 8, 8, 6, 5]	[8, 8, 8, 5, 5, 5]

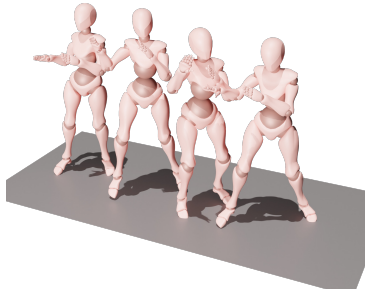
Table 6. The choices of L in FSQ.

G. More Generation Visualizations

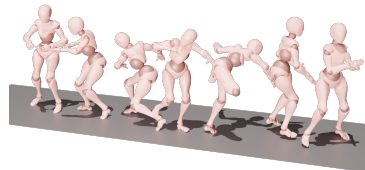
We show some of the generation results in Fig. 2. The visualization shows our model could handle various types of texts. More generation visualizations and comparisons between different model sizes and codebook sizes can be found in the supplementary materials.



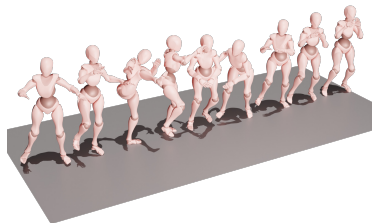
The person walks like the mummy.



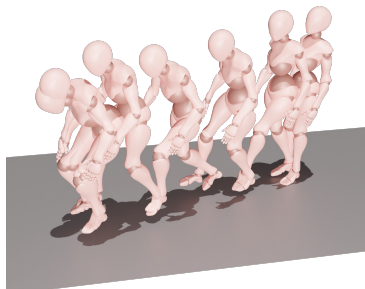
A person is perparing to battle.



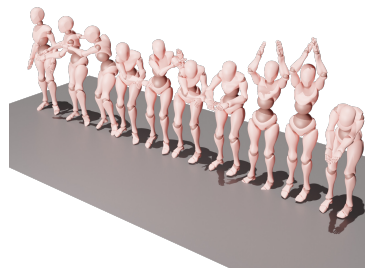
Side kick.



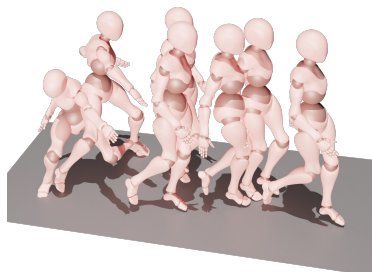
The person steps forward and kicks.



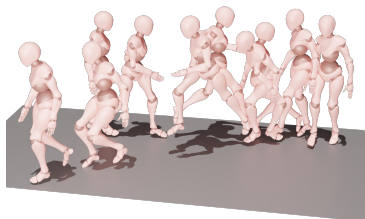
The person takes 4 steps forward, then shakes the legs.



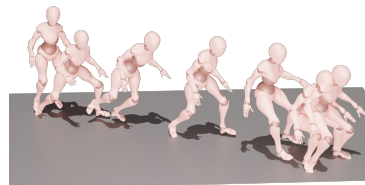
A person is acting like a human elephant.



A man Get Hit ,root motion get Forward,Heavy Stagger and Slow Recovery.



The character dashes forward, then leans to the left side and gathers strength, before explosively smashing the right and left sides forward. Finally, they stand upright.



The character bends down and goes through the obstacle, making a direct, light, and agile movement, flowing smoothly.

Figure 2. Human motion generation results on test set of ScaMo.

H. Limitations

The main limitation of this paper is the limited data. Unfortunately, we still have not observed the emerging abilities, based on these limited data. We are still working on collecting larger text-motion datasets and leaving it as our future work. Additionally, some of the data were sourced from video motion capture, which has posed quality constraints that, in turn, impact the generation quality.

References

- [1] Fangzhou Hong, Mingyuan Zhang, Liang Pan, Zhongang Cai, Lei Yang, and Ziwei Liu. Avatarclip: Zero-shot text-driven generation and animation of 3d avatars. *ACM SIGGRAPH*, 2022. [4](#)
- [2] Biao Jiang, Xin Chen, Wen Liu, Jingyi Yu, Gang Yu, and Tao Chen. Motiongpt: Human motion as a foreign language. *NeurIPS*, 2024. [3](#), [4](#)
- [3] Zhengyi Luo, Jinkun Cao, Kris Kitani, Weipeng Xu, et al. Perpetual humanoid control for real-time simulated avatars. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10895–10904, 2023. [5](#)
- [4] Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: Vq-vae made simple. *arXiv preprint arXiv:2309.15505*, 2023. [5](#)
- [5] OpenAI GPT-4 Team. Gpt-4 technical report. 2023. [2](#)
- [6] Ye Wang, Sipeng Zheng, Bin Cao, Qianshan Wei, Qin Jin, and Zongqing Lu. Quo vadis, motion generation? from large language models to large motion models. *arXiv preprint arXiv:2410.03311*, 2024. [3](#), [4](#)
- [7] Qi Wu, Yubo Zhao, Yifan Wang, Yu-Wing Tai, and Chi-Keung Tang. Motionllm: Multimodal motion-language learning with large language models. *arXiv preprint arXiv:2405.17013*, 2024. [3](#), [4](#)
- [8] Jianrong Zhang, Yangsong Zhang, Xiaodong Cun, Yong Zhang, Hongwei Zhao, Hongtao Lu, Xi Shen, and Ying Shan. Generating human motion from textual descriptions with discrete representations. In *CVPR*, pages 14730–14740, 2023. [2](#), [3](#)
- [9] Zixiang Zhou, Yu Wan, and Baoyuan Wang. Avatargpt: All-in-one framework for motion understanding planning generation and beyond. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1357–1366, 2024. [3](#)