SALAD: Skeleton-aware Latent Diffusion for Text-driven Motion Generation and Editing

Supplementary Material

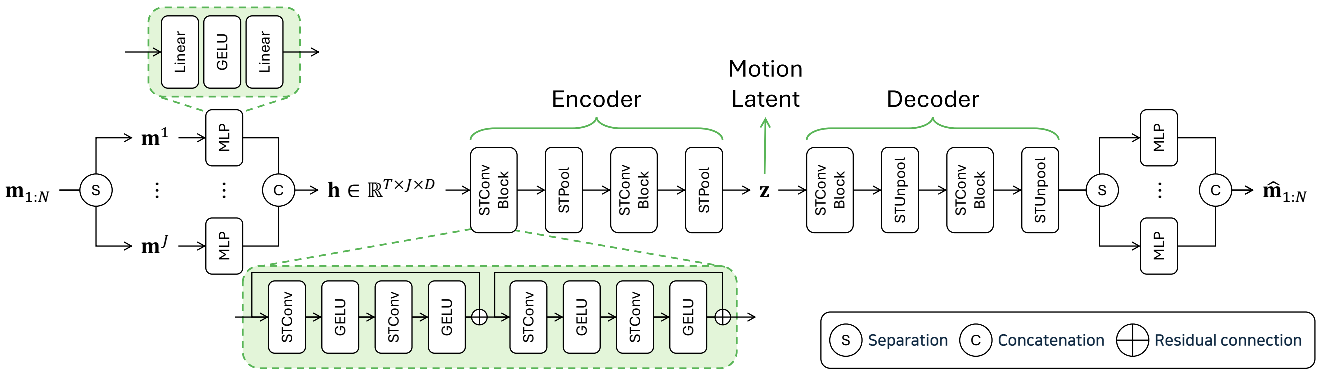

Figure 1. Detailed architecture of the VAE network. Separation and concatenation of tensors are performed with respect to the joint dimension.

A. Network Architectures

In this section, we provide a detailed explanation of the network architecture. The code is available at github.

A.1. Skeleto-temporal VAE

We visualize the detailed architecture of the VAE network in Figure 1. Given a motion sequence $\mathbf{m}$, we first decompose it into joint-wise features:

$$\mathbf{m} = \{\mathbf{m}^1, \dots, \mathbf{m}^J\},$$

where $\mathbf{m}^j \in \mathbb{R}^{N \times D_j}$. Here, N denotes the number of frames and D_j represents the number of features of joint j, which varies depending on the joint. Specifically, $D_j = 7$ for the root joint, as it includes 1-dimensional height, 2-dimensional translational velocity on the horizontal plane, 1-dimensional angular velocity around the up-axis, and 3-dimensional velocity. For the foot and toe joints, $D_j = 13$, comprising 3-dimensional local positions and velocities, 6-dimensional local rotations, and a contact label. For all the other joints, $D_j = 12$, excluding the contact label. To match the dimension of each joint within the skeleton-aware latent space, we apply a joint-wise multi-layer perceptron (MLP) to each $\mathbf{m}^j$, consisting of 2 linear layers and the Gaussian error linear unit (GELU) activation function [1] in the middle, yielding the joint-wise hidden latent variables $\mathbf{h}^j$.

As mentioned in the main paper, the skeleto-temporal convolution (STConv) layers use a combination of a skeletal convolution (SkelConv) and a temporal convolution (TempConv). The SkelConv layer is a graph convolution over the

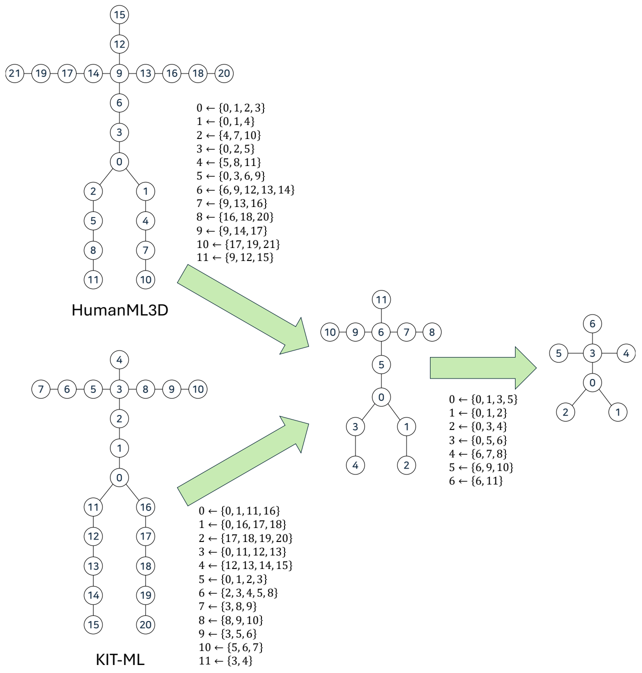

Figure 2. Illustration of the skeletal pooling process for the HumanML3D and KIT-ML datasets. The original skeleton (left) is progressively abstracted by pooling adjacent joints (middle and right). The notation $i \leftarrow \{\}$ indicates the abstracted joint index and the set of original joints that are grouped together. The unpooling layers operate in the reverse order to restore the skeletal resolution.

joint dimension, which is defined as follows:

$$\mathrm{SkelConv}(\mathbf{h}^j) := \mathbf{\Theta}_1(\mathbf{h}^j) + \frac{1}{|\mathcal{N}(j)|} \sum_{n \in \mathcal{N}(j)} \mathbf{\Theta}_2(\mathbf{h}^n),$$

where $\mathbf{\Theta}_{\{1,2\}}$ represents a linear feed-forward layer, and $\mathcal{N}(j)$ denotes the indices of joints neighboring to joint j. The TempConv is a 1D convolution layer with a kernel of size 3 and a stride of 1, and it is shared across all joints. Additionally, we use residual connections within a stack of STConv layers. For the activation function, we use the GELU function at the end of each STConv layer.

For the skeleto-temporal pooling (STPool) layers, we apply the average pooling across both skeletal and temporal dimensions. Specifically, temporal features are pooled with a kernel of size 2 and a stride of 2, while skeletal pooling reduces the number of joints by summarizing adjacent joints. In contrast, the skeleto-temporal unpooling (STUnpool) layers perform the inverse operation of STPool. To increase the skeletal resolution, we recreate unpooled joints by summing the features from the corresponding pooled joints. The temporal resolution is increased by upsampling the features along the temporal dimension using linear interpolation. The downsampling and upsampling for each dataset is visualized in Figure 2.

A.2. Skeleto-temporal Denoiser

In this section, we provide the implementation details for each component of the skeleto-temporal denoiser. **Positional Embedding.** We incorporate order information of $\mathbf{z}_t^l$ for both temporal and skeletal dimensions by applying positional embedding. Specifically, we first compute the positional embedding $\mathbf{e} \in \mathbb{R}^{TJ \times D}$ using the sinusoidal positional embedding method [5]. We then reshape this tensor to $\mathbb{R}^{T \times J \times D}$, and add it to the motion latent $\mathbf{z}_t^l$.
FiLM. To feed the diffusion timestep information to the denoiser, we employ the FiLM operator combined with an MLP layer, which has shown impressive performance in motion diffusion models [4, 6]. Specifically, the diffusion timestep t is passed through a sinusoidal positional embedding, followed by an MLP layer, resulting in a scale factor $\boldsymbol{\gamma}_t$ and shift factor $\boldsymbol{\beta}_t$, both D-dimensional vectors used to modulate the output of network modules:

$$\mathrm{FiLM}(\mathbf{z}, t) = \boldsymbol{\gamma}_t \odot \mathbf{z} + \boldsymbol{\beta}_t, \tag{1}$$

where γ_t and β_t are produced for each of attention block and feed-forward network in each layer. For brevity, we omit t for the FiLM in the following sections.
Feed-forward Network. To enhance the non-linearity capacity of the model, we employ an FFN module combined with FiLM:

$$\mathrm{FFN}(\mathbf{z}_t^l) \coloneqq \mathrm{MLP}(\mathrm{GELU}(\mathrm{MLP}(\mathrm{LN}(\mathbf{z}_t^l)))), \tag{2}$$
$$\mathbf{z}_t^{l+1} \leftarrow \mathbf{z}_t^l + \mathrm{FiLM}(\mathrm{FFN}(\mathbf{z}_t^l)), \tag{3}$$

where GELU represents the Gaussian error linear unit activation function [1]. Notably, FFN module produces the output of the l-th transformer layer, which we denote as $\mathbf{z}_t^{l+1}$, and it is used as an input of the $(l+1)$-th layer.

A.3. Hyperparameters

For training the VAE, we set $\lambda_{\mathrm{pos}} = 0.5$, $\lambda_{\mathrm{vel}} = 0.5$, and $\lambda_{\mathrm{kl}} = 0.02$. We also used a learning rate scheduler for training both the VAE and denoiser, which linearly increased the learning rate from 0 to 0.0002 over the first 2000 steps, then decayed it by a factor of 0.1 at 150,000 and 250,000 iterations for the VAE, and at 50,000 iterations for the denoiser. The latent dimensions for modules of the skeleto-temporal VAE was set to 32, while the denoiser used a latent dimension of 256.

We used a scaled linear scheduler for the noise schedule during denoiser training, similar to the approach used in Stable Diffusion [3]. Specifically, β_t at diffusion timestep t is defined as follows:

$$\beta_t = \left(\sqrt{\beta_1} + \frac{t-1}{T-1} \cdot \left(\sqrt{\beta_T} - \sqrt{\beta_1}\right)\right)^2, \tag{4}$$

where we set $\beta_1 = 0.00085$, $\beta_T = 0.012$, and $T = 1000$ following the default setting of Stable Diffusion. This scheduler allows for a smooth transition between noise levels, ensuring effective denoising throughout the entire process.

B. Zero-shot Editing via Attention Modulation

To enable zero-shot editing by modulating cross-attention maps between motion and text, we adopt the editing procedure introduced by Prompt-to-Prompt [2], which is depicted in Algorithm 1. Let $\mathrm{Denoise}(\mathbf{z}_t, t, c)$ represent a single denoising step at diffusion timestep t using a pre-trained SALAD model conditioned on text prompt c, producing the denoised latent $\mathbf{z}_{t-1}$ and cross-attention map $\mathbf{M}_t$. Additionally, $\mathrm{Denoise}(\mathbf{z}_t, t, c)\{\mathbf{M} \leftarrow \hat{\mathbf{M}}\}$ denotes a denoising step where the original attention map $\mathbf{M}$ is replaced by a modified attention map $\hat{\mathbf{M}}$, while the value $\mathbf{V}$ is computed using the target prompt c^*. The function $\mathrm{Edit}(\mathbf{M}_t, \mathbf{M}_t^*, t)$ is a general editing function that modulates two attention maps based on the diffusion timestep t, which will be elaborated in the following sections.
Word Swap. The objective of word swap is to incorporate descriptions from the target prompt while maintaining movements from the source prompt. To this end, the source prompt is used during early denoising steps, and the target prompt is used in the later steps:

$$\mathrm{Edit}(\mathbf{M}_t, \mathbf{M}_t^*, t) \coloneqq \begin{cases} \mathbf{M}_t^* & \text{if } t < \tau \\ \mathbf{M}_t & \text{otherwise,} \end{cases}$$

where τ is a hyperparameter that determines when to switch from the source prompt to the target prompt during the denoising process. Because the overall composition is established in the early steps, we guide the overall structure by injecting the attention maps from the source prompt during

Algorithm 1 Zero-shot editing by cross-attention modulation

```
1:  Input: A source prompt $c$ and a target prompt $c^*$.
2:  Output: A source motion $\mathbf{z}_0$ and an edited motion $\mathbf{z}_0^*$.
3:  $\mathbf{z}_T \sim \mathcal{N}(0, I)$ a unit Gaussian random variable;
4:  $\mathbf{z}_T^* \leftarrow \mathbf{z}_T$;
5:  for $t = T, T-1, \ldots, 1$ do
6:      $\mathbf{z}_{t-1}, \mathbf{M}_t \leftarrow \mathrm{Denoise}(\mathbf{z}_t, t, c)$;
7:      $\mathbf{z}_{t-1}^*, \mathbf{M}_t^* \leftarrow \mathrm{Denoise}(\mathbf{z}_t^*, t, c^*)$;
8:      $\hat{\mathbf{M}}_t \leftarrow \mathrm{Edit}(\mathbf{M}_t, \mathbf{M}_t^*, t)$;
9:      $\mathbf{z}_{t-1}^* \leftarrow \mathrm{Denoise}(\mathbf{z}_t^*, t, c^*)\{\mathbf{M} \leftarrow \hat{\mathbf{M}}_t\}$;
10: end for
11: return $(\mathbf{z}_0, \mathbf{z}_0^*)$
```

the early steps, and we use the attention maps of the target prompt to refine details later. Although the optimal value of τ can vary depending on the specific motion and text inputs, we found that $\tau = 0.8T$ generally yields good results.

Prompt Refinement. In this case, the target prompt is created by appending new tokens to the source text to add additional details. We first obtain the cross-attention map of the target text prompt $\mathbf{M}^*$, and then overwrite the attention values for existing tokens with those from the original cross-attention map $\mathbf{M}$ to preserve the common information between the two prompts:

$$(\mathrm{Edit}(\mathbf{M}_t, \mathbf{M}_t^*, t))_{i,j,k} := \begin{cases} (\mathbf{M}_t^*)_{i,j,k} & \text{if } \mathcal{A}(j) = -1 \\ (\mathbf{M}_t)_{i,j,\mathcal{A}(k)} & \text{otherwise,} \end{cases}$$

where i, j, and k denote the indices of the skeletal, temporal, and text tokens, respectively. The function $\mathcal{A}(\cdot)$ is an alignment function that maps a token index from the target prompt c^* to its corresponding index in c if one exists, or returns -1 otherwise.

Attention Re-weighting. This editing case involves amplifying or reducing the attention weights associated with specific word tokens to influence the resulting motions. Given a word token k^* and a scaling parameter s, we adjust the attention values as follows:

$$(\mathrm{Edit}(\mathbf{M}_t, \mathbf{M}_t^*, t))_{i,j,k} := \begin{cases} s \cdot (\mathbf{M}_t)_{i,j,k} & \text{if } k = k^* \\ (\mathbf{M}_t)_{i,j,k} & \text{otherwise.} \end{cases}$$

We empirically found that setting s within the range of $[-3, 3]$ yields reasonable editing results. Notably, using negative weights can generate semantically opposite outcomes, indicating that the cross-attention maps in SALAD capture a sophisticated understanding of the high-level relationships between text and motion.

Attention Mirroring. In this case, cross-attention values between counterpart body parts, such as the left and right arms, are swapped:

$$(\mathrm{Edit}(\mathbf{M}_t, \mathbf{M}_t^*, t))_{i,j,k} := (\mathbf{M}_t)_{i,\mathcal{C}(j),k}$$

where $\mathcal{C}(j)$ outputs the joint index of the counterpart if it exists, or j otherwise. This approach effectively mirrors the motion without needing to compute a new cross-attention map $\hat{\mathbf{M}}$ corresponding to an editing text prompt.

C. Additional Experiment Results

C.1. Cross-attention Maps

We present additional visualizations of cross-attention maps between the input text and generated motions in Figure 3. Overall, the attention maps consistently captured the relationships between the text and motion as reflected in the generated results. In Figure 3-(a), attention peaks for *jumping* and *high* occurred twice along the temporal dimension, corresponding to the character jumping twice, indicating that the attention map effectively captures the relationship between frames and words. Additionally, as shown in Figure 3-(b), the words *walking* and *treadmill* were associated with the lower body parts, showing consistently high weights along the frames, demonstrating that the attention map captures the relationship between body parts and words. Figure 3-(c), (d), and (e) further show that the cross-attention maps capture the relationship between skeleto-temporal features and each word, conveying which body part should to be activated, at which timing, and where to act, such as *sits ... ground*, *stretches arms ... forward*, and *puts both hands ... air*. Overall, these results demonstrate that cross-attention maps effectively capture the relationship between skeleto-temporal features and each word in the textual descriptions.

C.2. Classifier-free Guidance Weights

We present the quantitative results for different classifier-free guidance (CFG) weight values in Table 1. As mentioned in the main paper, the performance of SALAD improved across both metrics as the weight values increased, but excessively high weight values resulted in a decline in performance for both metrics. Notably, R-precision exhibited marginal differences for CFG weight values greater than or equal to 7.5, remaining within statistically equivalent ranges. In contrast, for FID and MM-Dist on the HumanML3D dataset, the default setting $w = 7.5$ yielded the best results. Similarly, in the KIT-ML dataset, $w = 7.5$ provided a balance between the quality and text-motion alignment. For Diversity, $w = 1.5$ produced results significantly inferior to the ground truth in both datasets, but the performance progressively improved as the CFG weight values increased. However, the optimal point at which the best Diversity is achieved was different between datasets: $w = 11.5$ for HumanML3D and $w = 7.5$ for KIT-ML.

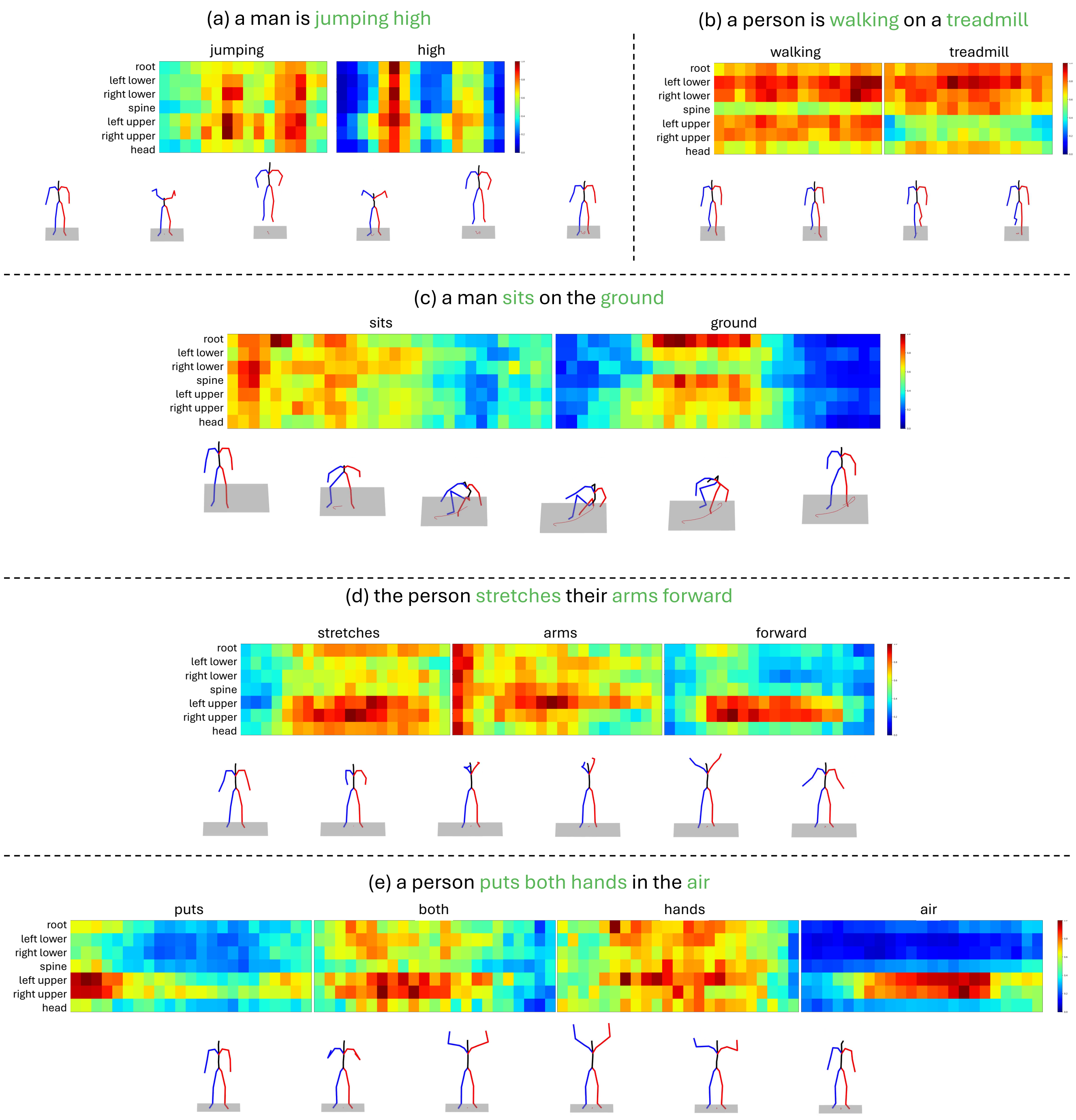

Figure 3. Additional visualizations of cross-attention maps between text and motion. Each row corresponds to a specific body part, and each column represents temporal frames.

C.3. Diffusion Parametrization

To demonstrate the effectiveness of $\mathbf{v}$-prediction parametrization, we compared the results of SALAD models trained with different parametrizations, as shown in the last 3 rows of Table **??**. While any variation of SALAD produced comparable or outperforming results compared to previous methods, $\mathbf{v}$-prediction consistently produced more stable results than the other alternatives across different datasets. Specifically, on the HumanML3D dataset, $\mathbf{x}$- and $\mathbf{v}$-prediction yielded statistically equivalent results for

w	R-Precision ↑ Top-1	Top-2	Top-3	FID ↓	MM-Dist ↓	Diversity →
Real motion	$0.511^{\pm.003}$	$0.703^{\pm.003}$	$0.797^{\pm.002}$	$0.002^{\pm.000}$	$2.974^{\pm.008}$	$9.503^{\pm.065}$
1.5	$0.438^{\pm.003}$	$0.622^{\pm.002}$	$0.727^{\pm.002}$	$1.291^{\pm.020}$	$3.411^{\pm.011}$	$9.298^{\pm.085}$
2.5	$0.517^{\pm.003}$	$0.705^{\pm.002}$	$0.801^{\pm.002}$	$0.457^{\pm.011}$	$2.945^{\pm.011}$	$9.616^{\pm.089}$
3.5	$0.548^{\pm.003}$	$0.738^{\pm.002}$	$0.829^{\pm.002}$	$0.223^{\pm.006}$	$2.782^{\pm.009}$	$9.711^{\pm.092}$
4.5	$0.563^{\pm.003}$	$0.755^{\pm.002}$	$0.842^{\pm.002}$	$0.132^{\pm.004}$	$2.711^{\pm.008}$	$9.737^{\pm.090}$
5.5	$0.573^{\pm.003}$	$0.763^{\pm.002}$	$0.850^{\pm.002}$	$0.093^{\pm.003}$	$2.674^{\pm.008}$	$9.741^{\pm.094}$
6.5	$0.578^{\pm.003}$	$0.767^{\pm.002}$	$0.854^{\pm.002}$	$0.078^{\pm.003}$	$2.656^{\pm.009}$	$9.722^{\pm.095}$
7.5 (default)	$0.581^{\pm.003}$	$0.769^{\pm.003}$	$0.857^{\pm.002}$	$0.076^{\pm.002}$	$2.649^{\pm.009}$	$9.696^{\pm.096}$
8.5	$0.581^{\pm.003}$	$0.770^{\pm.002}$	$0.857^{\pm.002}$	$0.083^{\pm.002}$	$2.650^{\pm.009}$	$9.669^{\pm.094}$
9.5	$0.581^{\pm.003}$	$0.771^{\pm.002}$	$0.858^{\pm.002}$	$0.097^{\pm.003}$	$2.655^{\pm.009}$	$9.638^{\pm.093}$
10.5	$0.581^{\pm.003}$	$0.770^{\pm.002}$	$0.857^{\pm.001}$	$0.116^{\pm.003}$	$2.663^{\pm.008}$	$9.608^{\pm.094}$
11.5	$0.581^{\pm.002}$	$0.769^{\pm.002}$	$0.857^{\pm.002}$	$0.138^{\pm.003}$	$2.673^{\pm.008}$	$9.577^{\pm.093}$
12.5	$0.578^{\pm.002}$	$0.768^{\pm.002}$	$0.856^{\pm.002}$	$0.164^{\pm.003}$	$2.686^{\pm.008}$	$9.657^{\pm.091}$

CFG Weights	R-Precision ↑ Top-1	Top-2	Top-3	FID ↓	MM-Dist ↓	Diversity →
Real motion	$0.424^{\pm.005}$	$0.649^{\pm.006}$	$0.779^{\pm.006}$	$0.031^{\pm.004}$	$2.788^{\pm.012}$	$11.08^{\pm.097}$
1.5	$0.439^{\pm.005}$	$0.656^{\pm.006}$	$0.774^{\pm.007}$	$0.837^{\pm.033}$	$2.755^{\pm.018}$	$11.033^{\pm.115}$
2.5	$0.471^{\pm.006}$	$0.691^{\pm.005}$	$0.809^{\pm.005}$	$0.489^{\pm.017}$	$2.593^{\pm.013}$	$11.085^{\pm.118}$
3.5	$0.480^{\pm.006}$	$0.706^{\pm.005}$	$0.818^{\pm.006}$	$0.396^{\pm.013}$	$2.562^{\pm.013}$	$11.086^{\pm.120}$
4.5	$0.484^{\pm.006}$	$0.709^{\pm.005}$	$0.823^{\pm.006}$	$0.352^{\pm.011}$	$2.559^{\pm.015}$	$11.083^{\pm.122}$
5.5	$0.484^{\pm.006}$	$0.710^{\pm.005}$	$0.824^{\pm.006}$	$0.324^{\pm.009}$	$2.571^{\pm.016}$	$11.078^{\pm.122}$
6.5	$0.481^{\pm.007}$	$0.708^{\pm.006}$	$0.824^{\pm.006}$	$0.308^{\pm.009}$	$2.588^{\pm.016}$	$11.077^{\pm.119}$
7.5 (default)	$0.480^{\pm.007}$	$0.707^{\pm.006}$	$0.826^{\pm.006}$	$0.298^{\pm.009}$	$2.608^{\pm.015}$	$11.079^{\pm.119}$
8.5	$0.476^{\pm.008}$	$0.705^{\pm.005}$	$0.823^{\pm.006}$	$0.294^{\pm.008}$	$2.630^{\pm.015}$	$11.083^{\pm.120}$
9.5	$0.477^{\pm.007}$	$0.704^{\pm.005}$	$0.822^{\pm.005}$	$0.294^{\pm.008}$	$2.651^{\pm.015}$	$11.089^{\pm.120}$
10.5	$0.474^{\pm.006}$	$0.703^{\pm.005}$	$0.820^{\pm.006}$	$0.297^{\pm.009}$	$2.674^{\pm.015}$	$11.096^{\pm.121}$
11.5	$0.472^{\pm.006}$	$0.702^{\pm.006}$	$0.818^{\pm.006}$	$0.303^{\pm.009}$	$2.697^{\pm.015}$	$11.102^{\pm.121}$
12.5	$0.471^{\pm.006}$	$0.697^{\pm.006}$	$0.815^{\pm.006}$	$0.311^{\pm.010}$	$2.719^{\pm.015}$	$11.108^{\pm.122}$

Table 1. Quantitative evaluation results with different CFG weight values on the test sets of HumanML3D (top) and KIT-ML (bottom). ↑ and ↓ denote that higher and lower values are better, respectively, while → denotes that the values closer to the real motion are better. Red and blue colors indicate the best and the second best results, respectively.

R-precision within the confidence range, but $\mathbf{v}$-prediction achieved superior results on FID and MM-Dist compared to $\mathbf{x}$-prediction, while ϵ-prediction consistently produced inferior results. On the KIT-ML dataset, ϵ-prediction scored better on FID compared to $\mathbf{x}$-prediction, while both achieved similar results in terms of the text-motion alignment. Also on this dataset, $\mathbf{v}$-prediction significantly outperformed both alternatives. These results demonstrate the effectiveness of $\mathbf{v}$-prediction in enhancing stability and robustness in text-to-motion generation using SALAD.

C.4. FiLM Layers

While FiLM has been adopted in several motion diffusion models [4, 6], its effectiveness in motion generation has yet to be fully evaluated. Therefore, we compared the performance of SALAD with and without FiLM, as shown in Table 3. For the model without FiLM layers, we injected diffusion timestep information to $\mathbf{z}_t^l$ by directly adding the positional embedding of diffusion timestep t to it. Across both datasets, FiLM layers improved both text-motion alignment and generation quality, indicating that explicit modulation of intermediate representations effectively produces high-quality and text-faithful results.

Parametrization	R-Precision (Top-3) ↑	FID ↓
$\mathbf{x}$-prediction	$0.860^{\pm.002}$	$0.111^{\pm.003}$
ϵ-prediction	$0.803^{\pm.002}$	$0.257^{\pm.008}$
$\mathbf{v}$-prediction	$0.857^{\pm.002}$	$0.076^{\pm.002}$

Table 2. Quantitative results on different diffusion parametrizations.

Method	R-Precision (Top-3) $\uparrow$	FID $\downarrow$
with FiLM	$0.857^{\pm.002}$	$0.076^{\pm.002}$
without FiLM	$0.843^{\pm.002}$	$0.087^{\pm.003}$

Method	R-Precision (Top-3) $\uparrow$	FID $\downarrow$
with FiLM	$0.828^{\pm.005}$	$0.296^{\pm.012}$
without FiLM	$0.803^{\pm.006}$	$0.311^{\pm.013}$

Table 3. Ablation results showing the effect of FiLM layers on the test sets of HumanML3D (top) and KIT-ML (bottom).

Method	FID $\downarrow$	MPJPE $\downarrow$
Full model	$0.003^{\pm.000}$	$0.016^{\pm.000}$
without $\mathcal{L}_{\text{pos}}$	$0.005^{\pm.000}$	$0.023^{\pm.000}$
without $\mathcal{L}_{\text{vel}}$	$0.009^{\pm.000}$	$0.016^{\pm.000}$
without both	$0.012^{\pm.000}$	$0.024^{\pm.000}$

Table 4. Ablation results showing the effect of FiLM layers on the test sets of HumanML3D (top) and KIT-ML (bottom).

C.5. Loss Terms of VAE

To validate the effectiveness of the auxiliary loss terms in the VAE, including $\mathcal{L}_{\text{pos}}$ and $\mathcal{L}_{\text{vel}}$, we evaluated FID and MPJPE while ablating these loss terms, as shown in Tab. 4. Removing $\mathcal{L}_{\text{pos}}$ had a negative effect in both metrics, while $\mathcal{L}_{\text{vel}}$ had no effect on MPJPE but led to a more significant degradation in FID compared to ablating $\mathcal{L}_{\text{pos}}$. Furthermore, removing both loss terms resulted in the worst performance across all metrics. These results demonstrate the importance of auxiliary losses on joint positions and velocities in refining the latent space, contributing to improving motion quality and reliability.

References

[1] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415, 2016. 1, 2

[2] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. arXiv preprint arXiv:2208.01626, 2022. 2

[3] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 10684–10695, 2022. 2

[4] Jonathan Tseng, Rodrigo Castellon, and Karen Liu. Edge: Editable dance generation from music. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 448–458, 2023. 2, 5

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in Neural Information Processing Systems. Curran Associates, Inc., 2017. 2

[6] Mingyuan Zhang, Zhongang Cai, Liang Pan, Fangzhou Hong, Xinying Guo, Lei Yang, and Ziwei Liu. Motiondiffuse: Text-driven human motion generation with diffusion model. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2024. 2, 5