Supplementary Material for the paper: Action-conditioned On-demand Motion Generation

1 OVERVIEW

Our supplementary material includes two parts:

- Supplementary video which includes gif examples for mode interpolation and trajectory customization.
- Supplementary document which introduces more implementations details as mentioned in the main text.

In this document, we first introduce the model architectures and more implementation details in Section 2, then demonstrate more details for our mode preserving VAE in Section 3. Dataset details can also be found in Section 4.

2 IMPLEMENTATION DETAILS

In this section, we introduce the implementation details of ODMO and other baseline algorithms, mode-preserving sampling, and metrics for evaluating the generated motion sequences.

2.1 ODMO Implementation Detail

We implemented the framework of ODMO using PyTorch and trained the models with TITAN RTX GPU. The detailed architecture for each component is shown in Figure 2 in main text and Table 2. Note that the latent space dimension (the output of the LMP encoder) is 20 and the decoder only relied on this small code space and the one-hot encoded class information to reconstruct the motion. During training, the contrastive margin α is set to be 5, and the target variance of posteriors is set to 0.05. In the training of the network, we adopted the Adam optimizer with the learning rate 10^{-3} for both dataset. Additionally, teacher forcing (TF) is applied for enforcing the stable and efficiency of the training with the TF rate linear decayed from 1 to 0.3.

2.2 Mode-preserving sampling and silhouette score

We use sihouette score to find the number of modes in activity category adaptively. To be specific, we fit a GMM with k mixture components and compute the silhouette score S_k. We vary k from 3 to 11 to get $S_3 \cdots S_{11}$. Starting from S_3, we determine an i such that $S_i > 0.95 * S_{i+1}$ for the first time. Then the number of modes is selected as i. We dropped mixture components whose size is smaller than 10 for denoising purpose.

2.3 Finding the optimal σ^2

As we show in Section 3, contrastive mechanism applied on the encoder's output mean $\mu(\mathbf{x})$ together with the prior regularization of encoder's output covariance vector $\sigma^2(\mathbf{x})$ helps to form our data-driven mode-aware latent space. To find the most suitable range of prior scale σ^2 for the motion generation task, we sweep the parameter and list the result in Table 1. As we can see, as σ^2 increases, the performance drops especially for Acc and FID since the estimated posterior $q_\phi(\mathbf{z} \mid \mathbf{x})$ becomes too flat losing essential information of each motion $\mathbf{x}$. At the same time, when σ^2 becomes too small, the generalization ability of the learnt model decreases thus Diversity is low in general. Taking all the metrics different models obtained and the effect of σ^2 on model generalization ability into consideration, we use $\sigma^2 = 0.05$ to train all our final models (listed in the main paper).

Table 1: σ^2 selection in three datasets.

HumanAct12				
Methods	Acc↑	FID↓	Diversity	MModality
Real motions	$99.70^{\pm.06}$	$0.02^{\pm.01}$	$7.08^{\pm.06}$	$2.53^{\pm.05}$
$\sigma^2 = 6.25$	$94.06^{\pm.30}$	$0.30^{\pm.01}$	$6.98^{\pm.15}$	$2.57^{\pm.05}$
$\sigma^2 = 1.25$	$95.84^{\pm.36}$	$0.31^{\pm.01}$	$7.00^{\pm.15}$	$2.53^{\pm.07}$
$\sigma^2 = 0.25$	$97.52^{\pm.22}$	$0.15^{\pm.01}$	$7.08^{\pm.14}$	$2.42^{\pm.05}$
$\sigma^2 = 0.05$	$97.81^{\pm.21}$	$0.12^{\pm.01}$	$7.05^{\pm.15}$	$2.57^{\pm.04}$
$\sigma^2 = 0.01$	$97.81^{\pm.21}$	$0.09^{\pm.02}$	$7.03^{\pm.13}$	$2.53^{\pm.09}$

Mocap				
Methods	Acc↑	FID↓	Diversity	MModality
Real motions	$98.54^{\pm.14}$	$0.02^{\pm.00}$	$6.57^{\pm.11}$	$2.17^{\pm.08}$
$\sigma^2 = 6.25$	$86.95^{\pm.56}$	$1.14^{\pm.05}$	$6.32^{\pm.09}$	$2.85^{\pm.10}$
$\sigma^2 = 1.25$	$89.05^{\pm.37}$	$0.89^{\pm.05}$	$6.40^{\pm.08}$	$2.63^{\pm.08}$
$\sigma^2 = 0.25$	$91.69^{\pm.38}$	$0.60^{\pm.02}$	$6.51^{\pm.07}$	$2.63^{\pm.07}$
$\sigma^2 = 0.05$	$93.51^{\pm.39}$	$0.34^{\pm.03}$	$6.56^{\pm.07}$	$2.49^{\pm.06}$
$\sigma^2 = 0.01$	$89.82^{\pm.26}$	$0.80^{\pm.03}$	$6.40^{\pm.07}$	$2.83^{\pm.08}$

UESTC					
Methods	Acc↑	FID_{tr}↓	FID_{test}↓	Diversity	MModality
Real motions	$99.79^{\pm.05}$	$0.01^{\pm.00}$	$0.05^{\pm.00}$	$7.20^{\pm.06}$	$1.61^{\pm.02}$
$\sigma^2 = 6.25$	$59.19^{\pm.22}$	$3.46^{\pm.03}$	$3.45^{\pm.03}$	$6.51^{\pm.07}$	$2.26^{\pm.03}$
$\sigma^2 = 1.25$	$77.18^{\pm.28}$	$1.25^{\pm.02}$	$1.27^{\pm.02}$	$6.80^{\pm.04}$	$2.09^{\pm.03}$
$\sigma^2 = 0.25$	$89.83^{\pm.20}$	$0.44^{\pm.01}$	$0.40^{\pm.01}$	$6.99^{\pm.07}$	$1.87^{\pm.03}$
$\sigma^2 = 0.05$	$93.67^{\pm.18}$	$0.15^{\pm.00}$	$0.17^{\pm.00}$	$7.11^{\pm.07}$	$1.61^{\pm.03}$
$\sigma^2 = 0.01$	$93.70^{\pm.17}$	$0.10^{\pm.00}$	$0.12^{\pm.00}$	$7.08^{\pm.06}$	$1.76^{\pm.05}$

2.4 Baseline Algorithms

Action2Motion (A2M): We use A2M as one of our major baselines for the three public datasets. We trained the A2M with joints' xyz coordinates representation (A2M xyz) and A2M using Lie group representation (A2M Lie) on MoCap dataset for seven motion categories with their default parameters. On HumanAct12 dataset, we directly employed their released model as baseline. On UESTC dataset, we also directly trained A2M xyz and A2M Lie with their default parameters.

ACTOR: We treat ACTOR as the other strong baselines for UESTC and HumanAct12 dataset. We used their released model for final evaluation.

Other baselines: For implementation of Act-MoCoGAN, we followed their default architecture while replacing the image generator with pose generator and modified the image, video discriminators accordingly, as introduced in A2M paper. We trained the Act-MoCoGAN by tweaking the hyper-parameters so that it achieves the best generative results in the three dataset. As for the Gen-DLow, we adapted it to motion generation task by replacing the motion

Table 2: Architecture of ODMO

Module	Layers	details
LMP encoder	RNN layer	1 LSTM layer with 64 hidden units
	Feature layer	2 fully connected layers followed by BN and PReLU activation
	Output layer	1 fully connected layer
Trajectory generator	Embedding layer	2 fully conencted layers with PReLU activation in between
	RNN layer	2 LSTM layers with 128 hidden units
	Feature layer	3 fully conencted layers with PReLU activation
	Output layer	1 fully connected layer
Motion generator (Trajectory encoder)	Embedding layer	1 fully conencted layer with PReLU activation
	RNN layer	2 LSTM layers with 128 hidden units
Motion generator (Motion decoder)	Embedding layer	1 fully conencted layer with PReLU activation
	RNN layer	2 LSTM layers with 128 hidden units
	Output layer	1 fully connected layer

Table 3: Architecture of MoCap and HumanAct12 motion classifiers (same as Action2Motion)

Layers	Details
RNN layer	2 GRU layers with 128 hidden units
Feature layer	fully connected layer with Tanh activation
Output layer	1 fully connected layer

Table 4: Architecture of our UESTC classifier

Layers	Details
RNN layer	3 GRU layers with 256 hidden units
Feature layer	fully connected layer with Tanh activation
Output layer	1 fully connected layer

history condition as the class one-hot encoding. We trained the Gen-DLow (both VAE and DLow) in much more epochs (5000,600) than the default parameter (500, 500) so that it can be better to achieve the generative task.

2.5 Metrics Computation Details

Our metric computation follows the same procedure as Action2-Motion (A2M) which includes sampling of ground truth and generated motion sequences, as well as computing fidelity and diversity (multimodality) metrics based on them. However, since A2M and ACTOR used slightly different sampling strategies on generating the final evaluation metrics, for fair comparison, we recomputed their metrics with our sampling setup instead of using the numbers reported in their papers.

2.5.1 Metrics based on classifier. Frechet Inception Distance(FID), Multimodality and Diversity are computed based on the extracted features, i.e the output of the feature layer (see Tables 3 and 4) in the classifier.

FID: We randomly sample (400/250/200) samples for (MoCap / HumanAct12 / UESTC) samples from ground truth motion and generated motion across all categories indepedently. Then we calculate the sample mean μ and covariance matrix Σ of the activation of the extracted features $x \in \mathbb{R}^{30}$ (the feature dimension in classifier according to Action2Motion) from the set of real motion sequences as (μ_1, Σ_1) and the set of generated motion sequences as (μ_2, Σ_2) respectively. Lastly, FID is computed as $\|\mu_1 - \mu_2\|^2 + \mathrm{Tr}(\Sigma_1 + \Sigma_2 - 2\sqrt{\Sigma_1\Sigma_2})$.

Multimodality: To measure diversity inside one category, we randomly sample 20 pairs of motion sequences inside that category and extract their features as set C_1, then compute the average pairwise Euclidean distance among these pairs, i.e, $\frac{1}{20}\sum_{(x_i,x_j)\in C_1}\left\|x_i - x_j\right\|$.

Diversity: To measure the diversity of motion sequences across motion categories, we randomly sample 200 pairs of motion across all categories and extract their features as set C_2, then calculate the average pairwise Euclidean distance, i.e. $\frac{1}{200}\sum_{(x_i,x_j)\in C_2}\left\|x_i - x_j\right\|$.

2.5.2 Motion Classifiers. The metrics based on motion classifiers trained on real motion have been introduced in the A2M paper and we applied the same framework to compare performance of different models. For HumanAct12 dataset, we used the released classifier from A2M to make the results we benchmarked comparable. For MoCap dataset, we used 7 action categories instead of 8 action categories as in the A2M paper (Wash activity is dropped due to the noisy data), and thus we need to train our own classifier for our MoCap dataset. The architecture is shown in Table 3. Similarly, for UESTC dataset, a xyz representaion based classifier is not avalable from previous works, so we trained our own classifier with architecture shown in Table 4.

3 MODE PRESERVING VAE

3.1 VAE: From variational perspective to prior regularization

From a Variational Bayesian perspective, the evidence low bound (ELBO) can be derived from data log likelihood $\log p_\theta(x)$ minus KL divergence between the estimated posterior $q_\phi(\mathbf{z} \mid \mathbf{x})$ and $p_\theta(\mathbf{z} \mid \mathbf{x})$ as shown below:

$$\begin{aligned}\log p_\theta(\mathbf{x}) &= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x})\right] \\ &= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left(\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}\mid\mathbf{x})}\cdot\frac{q_\phi(\mathbf{z}\mid\mathbf{x})}{p_\theta(\mathbf{z}\mid\mathbf{x})}\right)\right] \\ &= \underbrace{\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left(\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}\mid\mathbf{x})}\right)\right]}_{ELBO} + D_{KL}(q_\phi(\mathbf{z}\mid\mathbf{x})\|p_\theta(\mathbf{z}\mid\mathbf{x})).\end{aligned}$$

To train a VAE for a given dataset D, the loss function, which needs to be minimized, is the negative of the ELBO, and can be rewritten as

$$\begin{aligned}-l_{\theta,\phi}(D) &= \sum_{\mathbf{x}\in D} \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left(p_\theta(\mathbf{x},\mathbf{z})\right) - \log\left(q_\phi(\mathbf{z}\mid\mathbf{x})\right)\right] \\ &= \sum_{\mathbf{x}\in D} \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log(p_\theta(\mathbf{x}\mid\mathbf{z}))\right] + D_{KL}(q_\phi(\mathbf{z}\mid\mathbf{x}) \parallel p_\theta(\mathbf{z})).\end{aligned}$$

For the standard VAE, we assume the prior $p(\mathbf{z})$ is a Normal distribution with zero mean and unit variance, and the estimated posterior $q_\phi(\mathbf{z}\mid\mathbf{x})$ is assumed to be a normal distribution with mean $\mu(\mathbf{x}) \in \mathbb{R}^d$ and covariance $\text{diag}(\sigma^2(\mathbf{x})) \in \mathbb{R}^{d\times d}$. Taking the powerful transformation capacity of neural networks into consideration, this assumption generally works. With this choice of $q_\phi(\mathbf{z}\mid\mathbf{x})$, we have:

$$\begin{aligned}-l_{\theta,\phi}(D) &= \sum_{\mathbf{x}\in D} \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log(p_\theta(\mathbf{x}\mid\mathbf{z}))\right] + D_{KL}(q_\Phi(\mathbf{z}\mid\mathbf{x}) \parallel p_\theta(\mathbf{z})) \\ &= \sum_{\mathbf{x}\in D} (\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log(p_\theta(\mathbf{x}\mid\mathbf{z}))\right] + \\ &\quad \frac{1}{2}\left[-\sum_i^d\left(\log\sigma_i^2(\mathbf{x}) + 1\right) + \sum_i^d \sigma_i^2(\mathbf{x}) + \sum_i^d \mu_i^2(\mathbf{x})\right]).\end{aligned} \tag{1}$$

The encoder and decoder networks with parameters, ϕ and θ, respectively can be trained jointly with the loss function in Equation (1).

If we leave the variational framework aside, ELBO can also be viewed from a quite different optimization perspective: VAE is more like an autoencoder with additional sampling process in the latent space. We want the encoding decoding process have good reconstruction results, so a reconstruction loss term should be imposed. At the same time, we want some regularization on the encoder's output $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ such that there is less overfitting and more generalization. Without regularization of $\mu(\mathbf{x})$, the latent space can potentially be infinite and too sparse and specific for each datapoint. Without regularization of $\sigma(\mathbf{x})$, it can safely go to 0 which reduces to an autoencoder without any sampling power. So we need regularization for both terms. Thus we can rewrite ELBO as a combination of a reconstruction loss and a regularization term on $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$, i.e. the negative of log of prior distribution of these parameters.

We assume that for each data point $\mathbf{x}$, the prior for encoded vector μ is independent over all components. For μ, that term would correspond to a prior of μ under normal distribution, i.e.

$$\frac{-1}{2}\mu_i^2(\mathbf{x}) \longrightarrow \mu_i(\mathbf{x}) \sim \mathcal{N}\left(\mu_i(\mathbf{x}) \mid 0, 1\right).$$

Similarly, for $\sigma(\mathbf{x})$, its related regularization term would correspond to a prior of gamma distribution,

$$\frac{1}{2}\left(\log\left(\sigma_i^2(\mathbf{x})\right) - \sigma_i^2(\mathbf{x})\right) \longrightarrow \sigma_i^2(\mathbf{x}) \sim \Gamma\left(\sigma_i^2(\mathbf{x}) \mid \alpha = \frac{3}{2}, \beta = \frac{1}{2}\right).$$

3.2 Contrastive Mechanism

Following this point of view, more complex prior of μ's should be constructed to provide more modeling capacity in the latent space. To automatically form such a structure in the latent space, we deducted some general principles for the algorithm to follow. First, when the motion sequences are similar, their latent codes should be nearby, it leads to the assumptions that the latent codes corresponding to the same mode should cluster in the latent space. At the same time, the low-dimensional representation of different motion sequences are far apart in the latent space, which means that each category should its own cluster.

Instead of using standard distributions, we form a distribution using self contrasting mechanism. We want to regularize μ jointly for n datapoints. The most convenient way is to view it as a spring ensemble system. The probability of one configuration $X : (\mu_1, \cdots, \mu_n)$ is modeled as Boltzmann distribution, which is proportional to the exponential of the configurations energy, i.e. $p(x) \propto e^{-\beta E}$, where $E = \sum_{i=1}^n \sum_{j=1}^n \frac{1}{2}k(c_i, c_j)(\mu(\mu_i, \mu_j) - \mu_s(c_i, c_j))^2$. In different cases, we specify stiffness function k and stress free length μ_s accordingly.

For encoded two data points corresponding to the same activity, μ_i and μ_j, then we assign a spring whose stress free length is $\mu_s = 0$ and the dynamic length is $\mu = \left\|\mu_i - \mu_j\right\|$. We assume the stiffness is 1, then its energy is $\frac{1}{2}(\mu - \mu_s)^2$. Higher energy is contained by the pair of points with further distance. A spring like this is assigned to every pair of μ_i and μ_j belonging to the same category for all categories. Thus if we look at energy from one category c, minimizing this part of the energy would be equivalent to minimizing the distance of each $\mu_{c,i}(1 \le i \le n_c)$ (encoded data belonging to category c) to the category center $\mu_c = \frac{\sum_{i=1}^{n_c}\mu_{c,i}}{n_c}$, as shown in Equation (2).

$$\sum_{i=1}^{n_c}\sum_{j=1}^{n_c}\frac{1}{2}\left\|\mu_{c,i} - \mu_{c,j}\right\|^2 = 2n_c\sum_{i=1}^{n_c}\frac{1}{2}\left\|\mu_{c,i} - \mu_c\right\|^2 \tag{2}$$

Considering the encoded two data points μ_i and μ_j corresponding to different activities, we define a compression-only spring with stress free length $\mu_s = \alpha$ (margin size), and the spring's dynamic length as $\mu = \left\|\mu_i - \mu_j\right\|$, similarly as in the previous case. For this compression only spring, the energy is $\frac{1}{2}(\mu - \mu_s)^2$ only if $\mu < \alpha$. This leads to a hinge loss function in the optimization.

4 DATASETS

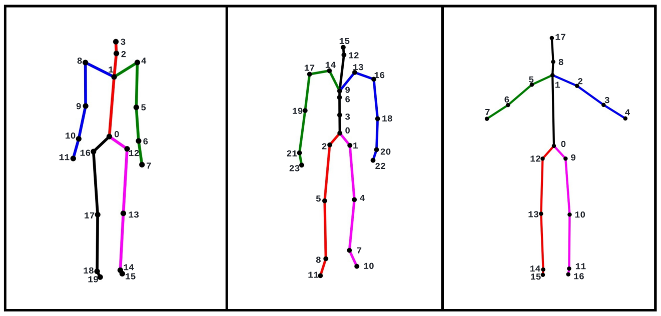

Figure 1: Human skeleton in the MoCap dataset (left), HumanAct12 dataset (middle), and UESTC dataset (right) with the joint number annotated in black. Root node (pelvis) is annotated as 0.

We used CMU MoCap dataset, HumanAct 12 dataset, and UESTC dataset, which contain diversified 3D motion sequences with varied lengths for seven, twelve, and forty daily activities respectively. MoCap dataset represents 3D human skeleton as 21 joints and 20 bones; while HumanAct 12 dataset uses 24 joints and 23 bones; For UESTC dataset, there are 18 joints and 17 bones as shown in Figure 1. The root node for these three datasets is 0. The detailed categories and number of motions belonging to each dataset are shown in Tables 5 to 7.

During the training phase, in order to balance the training samples from different motion categories, we applied the stratified sampling strategy. Furthermore, since each real motion sequence in the training data has different time duration, in each training epoch, we randomly sample a consecutive motion segment with the desired length from each real motion sequence: 100 frames for MoCap and 60 frames for HumanAct12, and 60 frames for UESTC. If one motion sequence is shorter than the desired length, we pad it with the last frame until the desired length is reached. After obtaining the fixed-length sub-sequences from the original data source, we shift it to ensure that the initial frame's root joint is at the origin.

Table 5: Statistics of our MoCap Dataset

Action type	Number of sequences
Walk	474
Run	109
Dance	77
Jump	108
Animal Behavior	101
Step	68
Climb	33
Total	970

Table 6: Statistics of HumanAct12 Dataset

Action type	Number of sequences
Warm up	215
Walk	47
Run	50
Jump	94
Drink	88
Lift dumbbell	218
Sit	54
Eat	77
Turn steering wheel	56
Phone	61
Boxing	140
Throw	91
Total	1191

Table 7: Statistics of UESTC Dataset

Action type	Number of sequences
standing-gastrocnemius-calf	345
single-leg-lateral-hopping	345
high-knees-running	345
rope-skipping	343
standing-toe-touches	339
front-raising	285
straight-forward-flexion	285
standing-opposite-elbow-to-knee-crunch	285
dumbbell-side-bend	285
shoulder-raising	285
single-dumbbell-raising	285
wrist-circling	285
punching	285
pulling-chest-expanders	284
shoulder-abduction	281
overhead-stretching	270
head-anticlockwise-circling	270
deltoid-muscle-stretching	270
upper-back-stretching	270
spinal-stretching	268
alternate-knee-lifting	255
knee-to-chest	255
knee-circling	255
bent-over-twist	255
standing-rotation	255
pinching-back	240
dumbbell-shrugging	240
dumbbell-one-arm-shoulder-pressing	240
elbow-circling	240
arm-circling	235
raising-hand-and-jumping	225
forward-lunging	225
left-kicking	225
left-lunging	225
punching-and-knee-lifting	225
squatting	225
jumping-jack	225
marking-time-and-knee-lifting	225
rotation-clapping	225
left-stretching	224
Total	10629