Flexible Motion In-betweening with Diffusion Models

Aditya Kumar

Details about the paper

Introduction

Motion in-betweening is the task of generating motion sequences to interpolate user-provided keyframes constraints. The paper uses diffusion models to generate diverse human motions based on text as well as keyframe constraints. The authors have named this model **Conditional Motion Diffusion In-betweening (CondMDI)**.

Problem Defination

- **Given -** text prompt p, keyframes $c \in \mathbb{R}^{N \times J \times D}$ where N is the number of frames, J is the number of joints and D is the dimension of feature vector that is used to represent the joint.
- **Output -** In-between frames showcasing human trajectory $\mathbf{x} = \{\mathbf{x}^i\}_{i=1}^{N} \in \mathbb{R}^{N \times J \times D}$
- **Sparsity -** The keyframes within the input may be sparse i.e. they have temporal sparsity and they may only contain a subset of joints

Dataset

The model is evaluated on the **HumanML3D dataset**, which contains 14,646 text-annotated human motion sequences taken from the AMASS and HumanAct12 datasets. Motion sequences have variable lengths with an average of 7.1 seconds and are padded with 0's to have a fixed length of 196 frames with 20 fps. Motion in every frame is represented by a 263-dimensional feature vector, capturing the relative root joint translations, rotations and local pose with respect to the root joint.

Training Details

The training strategy adopted in the paper follows the framework of denoising diffusion probabilistic models (DDPMs). The key aspects of the training procedure are as follows:

- The model is trained using **randomly sampled keyframes** with random joints masked to allow flexibility in handling various keyframe configurations during inference.
- The diffusion process is parameterized over **1,000 timesteps**, where Gaussian noise is progressively added to the motion data during training.
- The model learns to reverse this process by predicting the original motion sequence from noisy samples using a denoising objective.

- **Classifier-free guidance** is used, where 10% of the training samples are unconditioned (i.e., no keyframe or text conditioning is applied) to allow for flexible conditioning at inference time.
- The objective function minimizes the L2 loss between the ground-truth motion data x_0 and the generated sample estimates from the noisy inputs x_t.

During inference, the model generates new motion sequences by reversing the noise-adding process, conditioned on the text and keyframe inputs.

Details of the new Experiment

Motivation

One of the problems that the authors have identified with the results that are generated by the model is that of minor footskates and jitters for highly dynamic motions. One of the reasons that is identified by the authors is the problem with outliers in the dataset. The authors have identified that the HumanML3D dataset used to train the model contains skating and swimming data which is the reason for the footskates and decreased smoothness in the model. I have tried to remove this outliers by sampling the dataset based on the similarity of the data with **swimming** and **skating** text prompts.

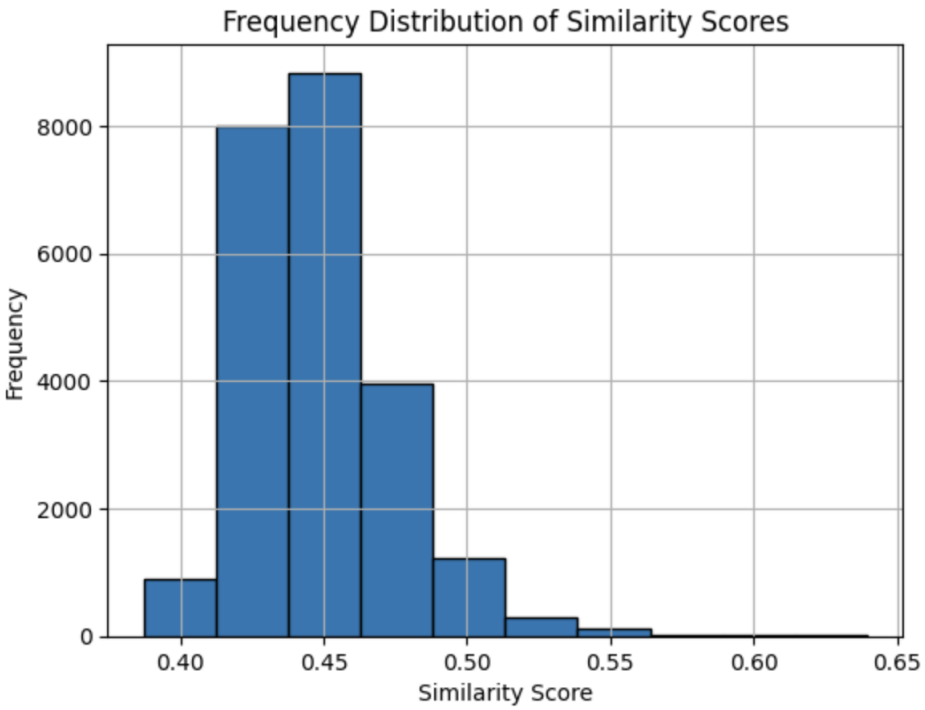
Goal

1. To remove the outliers from the dataset and see if the results improve in terms of footskates and smoothness.
2. To decrease the trajectory error and improve the R_precision of the model as compared to the previous model.

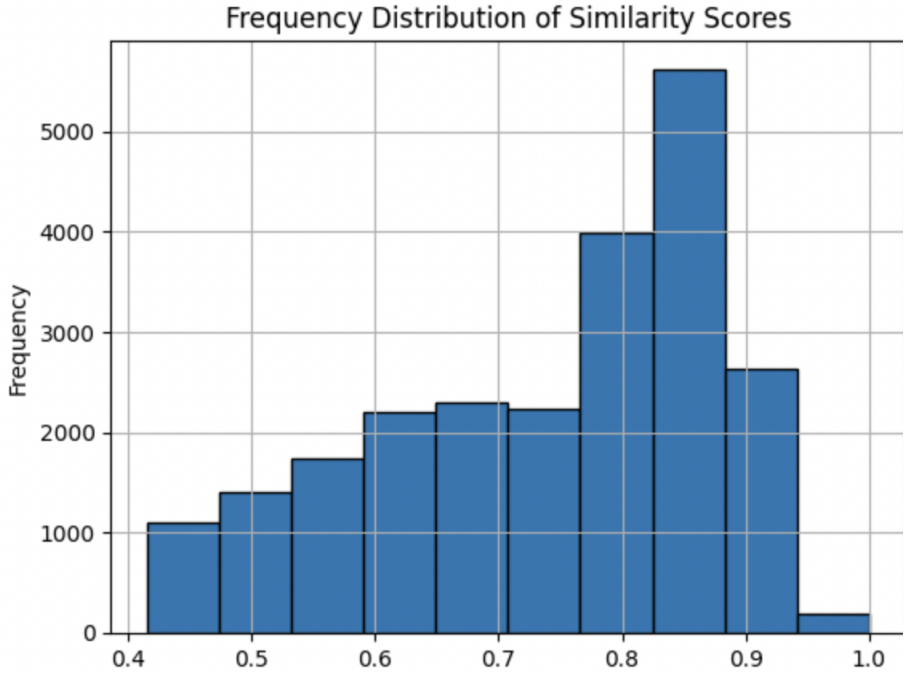
Method

My method for removing the outliers from the dataset can be divided into 2 parts:

1. **Scoring the data -** Before we can sample the data we first will have to score them based on some metrics do that we obtain an ordering in the dataset. For scoring the dataset we tried both **bag-of-words** and **word2vec** approaches to compute the similarity of text promts with the words **swimming** and **skating**. We found that the **word2vec** approach gave a much better distribution of scores that approximated a Gaussian distribution as compared to the **bag-of-words** approach. We even applied the **TFIDF** algorithm to give appropriate weights to the most improtant words when finding the similarity score. Below is the plot of the scores we obtained using the **word2vec** approach.



To compare here is the same plot for the scores obtained using the **bag-of-words** approach.



2. **Sampling the data -** Based on the distribution of score obtained above we sampled the data using a Gaussian sampling.
   Let $Z(\boldsymbol{X})$ = similarity score of the promt $\mathbf{X}$ obtained above.
   Let μ and σ be the mean and covariance matrix of the scores of the dataset.
   Then the probability of sampling $\mathbf{X}$ is given by the Gaussian distribution as:

$$p(\mathbf{X}|\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(Z(\boldsymbol{X})-\mu)^2}{2\sigma^2}}$$

We sampled 90% of the data using this Gaussian distribution and trained the model on this new dataset.

Results

The results of the new model are as follows:

```
---> [ground truth] R_precision: (top 1): 0.5147 (top 2): 0.7116 (top 3): 0.8068
Notebook editor cells
rajectory Error: (traj_fail_20cm): 0.9912 (traj_fail_50cm): 0.4922 (kps_fail_20c
m): 0.9184 (kps_fail_50cm): 0.3055 (kps_mean_err(m)): 0.6677
---> [vald] Keyframe Error: 0.7187
---> [vald] Skating Ratio: 0.1000
---> [vald] Matching Score: 9.8124
---> [vald] R_precision: (top 1): 0.0312 (top 2): 0.0742 (top 3): 0.1162
Time: 2024-11-03 19:52:31.495047
```

Observation and Conclusion

We observe that the **Trajectory error** and **Keyframe error** have increased but at the same time the **R-Precision** and the **Matching Score** have improved as expected.
The unexpected rise in **Trajectory error** and **Keyframe error** can have 2 plausible explanations:

1. The model was trained for only 5 epochs as comapred to 120000 epochs in the original paper. This could have led to the model not learning the correct motion sequences.

2. We have decreased the number of keyframes in the dataset to 0.9 times the original dataset (i.e. 13181 keyframes instead of 14646 keyframes). This could have led to the model not learning the correct motion sequences as the keyframes are the only source of information for the model to learn the motion sequences.