Journal of Electronics and Information Science (2023)
Clausius Scientific Press, Canada

DOI: 10.23977/jeis.2023.080309
ISSN 2371-9524 Vol. 8 Num. 3

MoCapDT: Temporal Guided Motion Capture Solving with Diffusion Transfer

Butian Xiong

Chinese University of Hong Kong (Shenzhen), Shenzhen, 518172, China
120090584@link.cuhk.edu.cn

Keywords: MoCapDT, 3D space, NC-SOFT

Abstract: We present an approach to reconstruct the joint location from noisy marker position in 4D data. The 4D data means the 3D location of different markers and time sequence. At the core of our approach, we apply a modified diffusion model architecture to transfer and denoise the raw marker information in the latent space under the guidance of other temporal data. Then we decode the latent space to not real skeleton 3D space. This enable us not only utilize the temporal guidance, we further utilize the iterative denoising technique to exploit the potential in the diffusion network. Furthermore, we demonstrate that our work outperform auto- encoder based deep learning model by large margin during our experiment on CMU-Synthesized data set and some real-world dataset provided by NC-SOFT.

1. Introduction

In the game development and movie producing, optical marker based motion capture is always the essential method to produce a high fidelity human motion. The procedure of optical motion capture divided into 2 steps. We first place the marker which is small sticker that can reflect the light on human body. Then we use the camera to receive the emission of the marker to calculate the accurate position of a particular marker. Compare to inertia based motion capture, optical marker based motion capture is usually more accurate. However, there still different noise lays in the motion capture task.

There are usually three different kinds of noise produced during the optical motion capture procedure. The major noise is produced by the marker switching problem. Compared to point cloud denoising [1], the major difference is that in current setting, markers are ordered and sparse. However, in commercial ready product like Shogun, the marker switching problem is common and usually lead to an intensive manual solving procedure after user have the noisy marker data. The second noise is due to the occlusion.

The occlusion usually happens when the target marker is covered by another body part say when one is bending. In [2], Holden usually set the occluded result to origin which was further proved not reliable by [3]. The third loss is obtained from the intrinsic un- controllable difference between camera and camera. Usually this problem will be largely reduced through camera calibration and other traditional method.

The mocap solving procedure is another different task, the problem we are facing is not simply denoising the data we have now. After we obtained the accurate marker position, we need to translate

the marker's motion to skeleton's motion or directly to human mesh [4].

In this paper we provide a simple baseline based on the encoder and decoder architecture. This baseline is implemented according to the MoCap Solver method [5]. And we added a guided diffusion architecture that outperform the old baseline by large- margin.

Compare to the traditional diffusion step we have in the image generation [6], we make the sampling procedure in the each diffusion time step to be definite. We change the architecture of the UNET [6] to be a large transformer obtained from SALAD[7]. We also apply an easy and useful training pipeline to distill knowledge from the skeleton encoder and decoder architecture.

To summarize our contribution:

(1) We provide a simple auto-encode base line to denoise marker.

(2) We propose a transformer based definite diffusion network to denoise the marker data.

(3) We carry out an extensive experiment based on CMU dataset and NCSOFT company data. Experiment shows a great boost to our method.

2. Related Work

2.1. Human Motion

There are relatively small amount of open source method for motion capture due to the strong correlation between the game&movie company, and motion capture technique. However, there still abundant literature for us to review. SMPL [8] is the most opening work for the deep learning based human motion analysis. It generates human meshes according to the one image. Following work including [9] [10] which all devote into the 3D mesh reconstruction from one RGB image. [11] [12], [13]push it further to study for human pose estimation from a sequence input such as video and skeleton. Another line of work is originate from different gaming company such as [14] and [2]. This line of works focus more on the solving of motion captured data to reduce the intensive laboring after they obtained they the marker position. However, in current work, we devolve deep into these two company work mentioned above. We find there are several problems. The first one is that there is no code and data has been published for Holden's method [2]. As for the MoCapSolver, the training hyperparameters is over 50 which is really hard to control and tunning. Despite that, we find that to train the MoCapSolver, it usually takes 6 days on only 200M meta data which makes even harder to adapt to other dataset with different marker setting. In our work, we simplify the MoCapSolver encoder decoder architecture as a baseline to shorten the training time and to decrease the number of hyper-parameter. Along with that, we provide a diffusion transfer method encoded with latent temporal information guidance. Our work outperform a large margin of the MoCapSolver on the real world data provided by NCSOFT company and CMU synthesize data.

2.2. Diffusion Network

Diffusion network is now the de-facto standard for computing graphic generation. Relatively older work including [15][16][17] make diffusion deep and scalable which means one can generate high fidelity RGB picture in the training data distribution. More recent works show an efficient method to guide the generating result through natural language[18][19]. After that, image editing has become the most popular topic in this field[20][21][22]. The 3D generation using NeRF as representation also becomes main stream after score distillation method was first invented by [23], other 3D generation method came into been such as [17][24][25]. However, since in our setting, we are focusing on the definite graph generation instead of generating of a grid like data. Recent work like molecular generation diffusion network such as [26][27] has been also vital. Our work was deeply influence by another line of work related to set generation and set arrangement, related work include[7]. In our

work, we utilize the set generation method provided by SALAD paper with definite label for each set element. To be specific, we transfer the set generation task to list generation task. We compare the generated result to noisy point cloud. When the difference is larger then some threshold, we will replace it with our generated result, otherwise it will stay as it was. We will discuss our method in details in method section[28].

3. Background

In the background session, we will introduce MoCapSolver method in details and some basic for the diffusion models, we will focus on the iterative denoising step with time stamp diffusion models.

3.1 MoCapSolver

MoCapSolver is the representative of the traditional encoder and decoder architecture. In that work, authors used a simple linear layer that can transfer a sequence of marker data to the latent space. The length of the sequence is called as windows length. By using this method, author try to utilize the temporal data. Furthermore, they divide MoCapSolver into three problem. One is for the skeleton autoencoder [5], one for the motion autoencoder, and one for the marker configuration autoencoder. However, according to the ablation study we have, the auto-encoder does not show a significant boost to the performance (see table 1) while it increases the training time dramatically.

Table 1: Marker Position Error Reproducing from MoCapSolver with Training Time

Code base	Marker Position Error (mm) without temporal smoothing	Training Time
Official pretrained model (with encoder trained)	15.4256	6 days
without encoder trained	18.75	12 hours
with encoder trained	17.57	6 days

$$\mathrm{MC}_\theta(\alpha) = \hat{\alpha}, \mathrm{MM}_\theta(\beta) = \hat{\beta}, \mathrm{MS}_\theta(\gamma) = \hat{\gamma} \tag{1}$$

$$\mathcal{L}_1(\mathrm{x}, \hat{\mathrm{x}}) = \mathrm{MSE}(\mathrm{x}, \hat{\mathrm{x}}) \tag{2}$$

$$\{\alpha, \beta, \gamma\} = \mathrm{x}, \{\hat{\alpha}, \hat{\beta}, \hat{\gamma}\} = \hat{\mathrm{x}} \tag{3}$$

The three autoencoders are denote as *MC*, *MM*, *MS* which are shown in equation one. They are marker configuration autoencoder, marker motion autoencoder and marker skeleton autoencoder. The input α, β, γ are the marker configuration, marker motion and joint position. Where $\alpha \subseteq \mathrm{R}^{B \times M \times J}$ where B is the batch size and M is the marker numbers and J is the joints number. Marker configuration is actually calculating the correlation between marker and skeleton. Intuitively, assume the skeleton for people is always the same, the difference between different people is related to the body shape, cloths and where the marker was places on target body. On the other hand, $\beta \subseteq \mathrm{R}^{B \times W \times M \times 3}$. Where W is the windows size of simply how long the temporal data you want to include in the motion autoencoder, since for each window, it is a 4d temporal data, we call it motion encoder. While for γ, $\gamma \subseteq \mathrm{R}^{B \times J \times 3}$. This is simply a skeleton encoder without any correlation between different frames of skeleton. i.e. without temporal data. Noted that θ is the parameter in different auto-encoder.

The loss function for these auto encoders are simply MSE (mean square loss), which is shown on the equation two, to make sure the auto encoder architecture converge. These three autoencoder will be decomposed into encoder and decoder. The decoder will be used in the next stage. Where we define *M* as the mocap solver encoder. It will transfer βn which represent the noisy motion into the

latent space. And it will take MC, MM, MS decoder to decode it back to β. We use a Huber loss as the loss function as illustrated in the equation 5.

$$\mathrm{M}_{\theta}\left(\beta_{\mathrm{n}}, \mathrm{MC}_{\theta \mathrm{d}}, \mathrm{MM}_{\theta \mathrm{d}}, \mathrm{MS}_{\theta \mathrm{d}}\right)=(\tilde{\alpha}, \tilde{\beta}, \tilde{\gamma}) \tag{4}$$

$$\mathcal{L}_2 = \mathcal{L}_{\delta}(\beta, \tilde{\beta}) \tag{5}$$

The Huber loss function is given by equation 6:

$$L_{\delta}(a)=\begin{cases}\frac{1}{2}a^2 \; if \; |a| \leq \; \delta \\ \left(|a|-\frac{1}{2}\delta\right) \text{if } |a|\end{cases} \tag{6}$$

Noted that we will use the absolute difference between predict y and ground truth to serve as alpha in the Huber loss function as illustrated by equation 7.

$$\alpha = |\mathrm{y}-\hat{\mathrm{y}}| \tag{7}$$

3.2 Diffusion Network

Latent Diffusion network[19] in general is a denoising process achieved by using different backbone architecture.

$$\mathrm{L}_{\mathrm{LDM}} \coloneqq \mathrm{E}_{\mathbb{E}(\mathrm{x}) \sim \mathrm{N}(\mathbb{0}, \mathbb{1}), \mathrm{t}}\left[|\epsilon-\epsilon_{\theta}(\mathrm{z}_{\mathrm{t}}, \mathrm{t})|_2^2\right] \tag{8}$$

In the equation 8, where we iteratively predict the noise that added to the original picture at the time stamp t, shows the essence of diffusion network. To minimize this loss function we need to minimize the difference between the real noise ϵ and the models prediction at time stamp t with previous noisy input z_t. Noted that we our noise ϵ is just some sample from a standard normal Gaussian distribution $N(0, 1)$

In diffusion procedure, we can alternate the random seed in the U-Net[6] architecture to vary the output that we want. However, in denoising problem, we don't want to do that. Therefore, we substitute the U-Net architecture to a transformer architecture[29] with time embedding. And utilize the cross attention mechanism to make different marker and joints be aware of others.

4. Method

We divide our method into three section, one for iteratively denoising step, one is for simplified encoder and decoder architecture. The last one is for the details of the method set up the big picture of our approach is shown in the Figure 1:

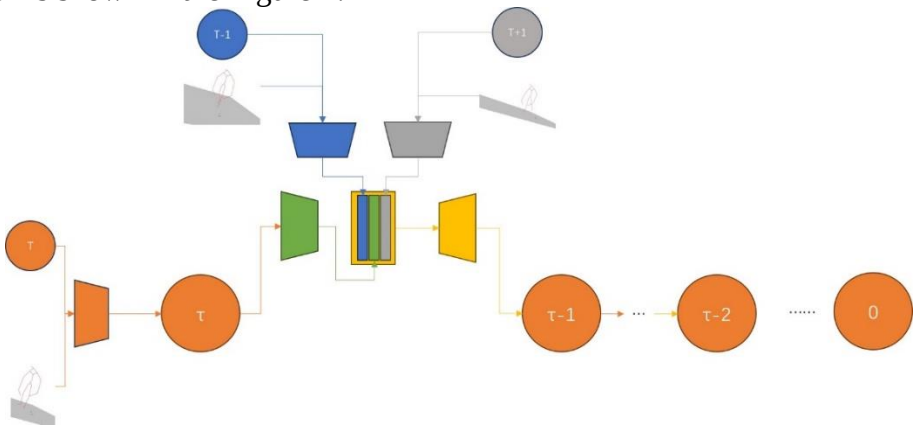

Figure 1: Model Architecture

4.1 Encoder and Decoder

As we mentioned in the background, the auto-encoder architecture is not useful and it will increase training time dramatically. Secondly, we also find out that in real world application, given the marker position we can easily obtain the skeleton position and the joint rotation easily. So here, our simplified version is that we directly use the simple fully connected layer as an auto encoder for denoising problem. The problem can be easily formulated as illustrated in the equation 9:

$$\mathcal{L}(\theta) = MSE\big(M_\theta(\beta_n), \beta_n\big) \tag{9}$$

Where M is the marker denoiser or simply an autoencoder. Notice that for simplicity, we did not implement cross attention in this the M denoiser. We only take one frame of the marker at one time. Therefore, it obviously decrease the computing amount.

4.2 MoCapDT

By obtaining the autoencoder we get form the last section, we first decompose this marker auto-encoder to encoder and decoder. Then we add a complex transformer layer inside that. The transformer layer takes the latent from the encoder or the latent from last time stamp as one input. And it also take two latent encoded by the same marker encoder as another input. Noted that those two frame is one frame behind and one frame before the current time stamp. We will do the denoising on the latent space for T times. Following equation 10 and 11 clarify this point in a mathematical style.

$$\mathrm{L_t} = \mathrm{M_{\theta e}}\big(\beta_{\mathrm{nt}}\big) \tag{10}$$

Where we denote $M\theta e$ as the marker encoder and L is the latent space for one marker frame at real world time sequence t. We will do the iteratively denoising using the following structure:

$$\mathrm{L_{LDM}} \coloneqq \mathrm{E_{\mathbb{E}(x)\sim N,t}}\big[|\epsilon - \epsilon_\theta(\mathrm{z_{t\tau}}, \mathrm{L_{t-\mathbb{1}}}, \mathrm{L_{t+\mathbb{1}}}, \tau,)|_2^2\big] \tag{11}$$

Where $z_{t\tau}$ is the the latent space denoised at denoising time stamp τ with real world time sequence number t. While for θ is the parameter we want to optimize for our transformer architecture. We also pass the latent code encoded by M at real world time stamp t-1 and t+1 to server as temporal guidance. The goal of the prior criterion $LLDM$ is to make our model predict the doise as accurate as possible. That means we will do the back-propagation using this criterion as the loss function.

4.3 Details

Our marker encoder is a 6 FC(fully connected) linear layer. While for the decoder is FC layer. Our transformer architecture has 4 self-attention blocks with 4 heads for the multi-head attention mechanism. We set the T to be 12 so that we can denoise our latent for 12 times using omne model.

5. Experiment and Result

Our experiment is mainly done on the sampled CMU synthesized dataset with more than 5000 training sequence and more than 600 testing sequence. We adding different noise including Gaussian Noise and out layer according to the MocapSolver. We measure the difference per-point according to the MSE embedded code in the pytorch-lightning matrix. We will introduce our experiment in 2 section. One for why simplified encoder is work and another for how our proposed denoising function is outperform the encoder decoder architecture.

5.1 Marker Autoencoder

As one can find from the graph shown here is that the loss during the training and the validation loss are keep dropping as the the epoch number is increasing. That means the simplified marker autoencoder is working.

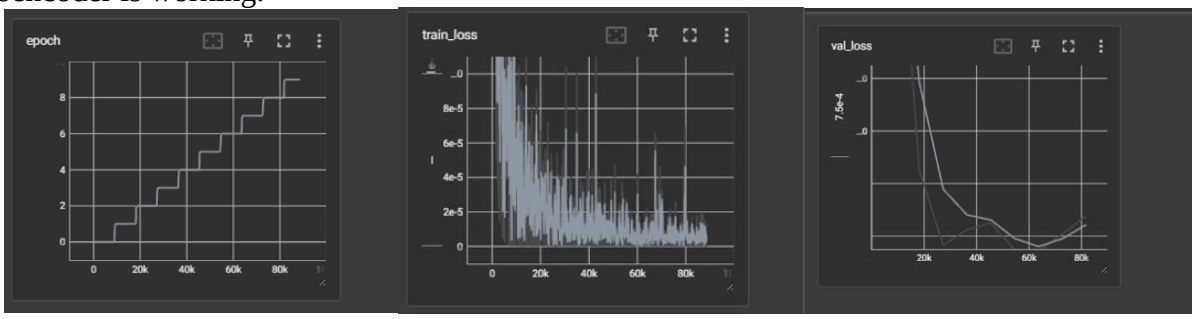

Figure 2: Training Convergence Varification

We set the batch size to be 512 and we sample each batch randomly to increase the robustness of our model.

5.2 MoCapDT

Then we set the same hyper parameters to our MoCapDT models and we find a much more stabled result. Here is the quantitative result.

Table 2: Final Result and Comparision with Baseline

	MPE	JPE	JOE
MoCapDT	20.13	17.61	6.66
Simplified Marker autoEncoder	142.98	134.58	31.17

MPE is the marker position error, JPE is the joint position error, and JOE is the joint rotation error. From the table 2 above we find that MoCapDT outperform the base line by large margin. Which means our architecture is working well.

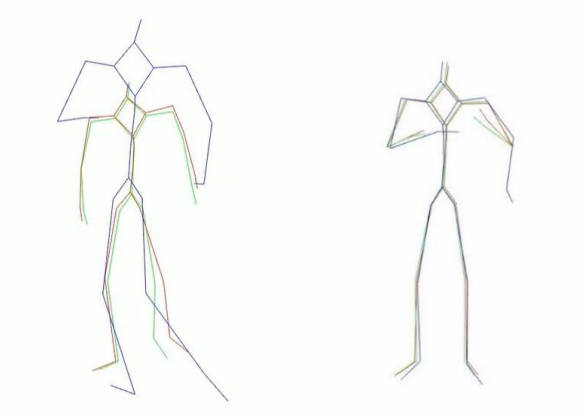

Figure 3: Quantitative Result Illustration

One can see from the qualitative result that our work outperforms the base line a lot in the figure 3. The blue line is the encoder base line while the red one is the ground truth, and the green line is our method. For more experiment result, one can visit this website: https://youtu.be/znuR6KtOj5g.

6. Conclusion

In this paper, we provide an easier baseline to implement the mocap solver. Along with that, we utilize the the temporal information by adding diffusion transformer to iteratively denoise the procedure. We use extensive experiment on the CMU dataset and NCSOFT company dataset to show that our method outperform the baseline by large margin. We also published NCSOFT company data and our baseline model code to make more contribution to the community.

References

[1] S. Luo and W. Hu, "Score-based point cloud denoising," in Proceedings of the IEEE/CVF Inter-national Conference on Computer Vision, 2021, pp. 4583–4592

[2] D. Holden, "Robust solving of optical motion capture data by denoising," ACM Transactions on Graphics (TOG), vol. 37, no. 4, pp. 1–12, 2018.

[3] K. Chen, Y. Wang, S.-H. Zhang, S.-Z. Xu, W. Zhang, and S.-M. Hu, "Mocap-solver: A neural solver for optical motion capture data," ACM Transactions on Graphics (TOG), vol. 40, no. 4, pp. 1–11, 2021.

[4] N. Ghorbani and M. J. Black, "Soma: Solving optical marker-based mocap automatically," in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 11 117–11126.

[5] K. Aberman, P. Li, D. Lischinski, O. Sorkine-Hornung, D. Cohen-Or, and B. Chen, "Skeleton-aware networks for deep motion retargeting," ACM Transactions on Graphics (TOG), vol. 39, no. 4, pp. 62–1, 2020.

[6] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18. Springer, 2015, pp. 234–241

[7] J. Koo, S. Yoo, M. H. Nguyen, and M. Sung, "Salad: Part-level latent diffusion for 3d shape generation and manipulation," arXiv preprint arXiv:2303.12236, 2023

[8] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "Smpl: A skinned multi- person linear model," ACM transactions on graphics (TOG), vol. 34, no. 6, pp. 1–16, 2015.

[9] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik, "End-to-end recovery of human shape and pose," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 7122–7131.

[10] N. Kolotouros, G. Pavlakos, and K. Daniilidis, "Convolutional mesh regression for single-image human shape reconstruction," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 4501–4510.

[11] N. Kolotouros, G. Pavlakos, M. J. Black, and K. Daniilidis, "Learning to reconstruct 3d human pose and shape via model-fitting in the loop," in Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 2252–2261.

[12] M. Kocabas, N. Athanasiou, and M. J. Black, "Vibe: Video inference for human body pose and shape estimation," in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 5253–5263.

[13] S. Goel, G. Pavlakos, J. Rajasegaran, A. Kanazawa, and J. Malik, "Humans in 4d: Reconstructing and tracking humans with transformers," arXiv preprint arXiv:2305.20091, 2023.

[14] Q. Cui and H. Sun, "Towards accurate 3d human motion prediction from incomplete observa-tions," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 4801–4810.

[15] A. Hertz, R. Mokady, J. Tenenbaum, K. Aberman, Y. Pritch, and D. Cohen-Or, "Prompt-to-prompt image editing with cross attention control," arXiv preprint arXiv:2208.01626, 2022.

[16] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," arXiv preprint arXiv:2010.02502, 2020.

[17] C.-H. Lin, J. Gao, L. Tang, T. Takikawa, X. Zeng, X. Huang, K. Kreis, S. Fidler, M.-Y. Liu, and T.-Y. Lin, "Magic3d: High-resolution text-to-3d content creation," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 300–309.

[18] J. Ho and T. Salimans, "Classifier-free diffusion guidance," arXiv preprint arXiv:2207.12598, 2022.

[19] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gon-tijo Lopes, B. Karagol Ayan, T. Salimans et al., "Photorealistic text-to-image diffusion models with deep language understanding," Advances in Neural Information Processing Systems, vol. 35, pp. 36 479–36 494, 2022.

[20] G. Couairon, J. Verbeek, H. Schwenk, and M. Cord, "Diffedit: Diffusion-based semantic image editing with mask guidance," arXiv preprint arXiv:2210.11427, 2022.

[21] A. Hertz, R. Mokady, J. Tenenbaum, K. Aberman, Y. Pritch, and D. Cohen-Or, "Prompt-to-prompt image editing with cross attention control," arXiv preprint arXiv:2208.01626, 2022.

[22] B. Kawar, S. Zada, O. Lang, O. Tov, H. Chang, T. Dekel, I. Mosseri, and M. Irani, "Imagic: Text -based real image

editing with diffusion models," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 6007–6017.
[23] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall, "Dreamfusion: Text-to-3d using 2d diffusion," arXiv preprint arXiv:2209.14988, 2022.
[24] Q. Shen, X. Yang, and X. Wang, "Anything-3d: Towards single-view anything reconstruction in the wild," arXiv preprint arXiv:2304.10261, 2023.
[25] G. Qian, J. Mai, A. Hamdi, J. Ren, A. Siarohin, B. Li, H.-Y. Lee, I. Skorokhodov, P. Wonka, S. Tulyakov et al., "Magic123: One image to high-quality 3d object generation using both 2d and 3d diffusion priors," arXiv preprint arXiv:2306.17843, 2023.
[26] M. Xu, L. Yu, Y. Song, C. Shi, S. Ermon, and J. Tang, "Geodiff: A geometric diffusion model for molecular conformation generation," arXiv preprint arXiv:2203.02923, 2022.
[27] M. Xu, A. S. Powers, R. O. Dror, S. Ermon, and J. Leskovec, "Geometric latent diffusion models for 3d molecule generation," in International Conference on Machine Learning. PMLR, 2023, pp. 38 592–38 610.
[28] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18. Springer, 2015, pp. 234–241.
[29] M. Xu, L. Yu, Y. Song, C. Shi, S. Ermon, and J. Tang, "Geodiff: A geometric diffusion model for molecular conformation generation," arXiv preprint arXiv:2203.02923, 2022.