Modern autoregressive architectures for the collective variable problem

Andrej Leban (T6)
Department of Statistics
University of Michigan
Ann Arbor, MI 48109
leban@umich.edu

Abstract

In molecular dynamics (MD), the challenge of discovering collective variables (CVs) remains central to understanding complex biomolecular processes. While existing approaches often treat the data as *i.i.d.*, we attempt to leverage modern autoregressive Transformer architectures to try to learn CVs directly by taking advantage of the autoregressive nature of the data - molecular trajectories - itself. Despite being originally developed for discrete data, we build on recent successes of such in areas with continuous data such as protein structure generation. We re-implement such architectures to suit our data and find that they are able to generate new trajectories, classify states, and that their internal representations reveal clues about the underlying physical process.

1 Introduction

Molecular dynamics (MD) simulations face two fundamental challenges in modeling the behavior: computational complexity due to the large number of component parts, and the rarity of important state transitions due to high energy barriers. These challenges have led to the development of collective variables (CVs) - low-dimensional descriptors that aim to parameterize the *free energy surface* as a lower-dimensional manifold and capture essential conformational changes while filtering out irrelevant atomic vibrations. The identification of effective CVs is crucial for enhanced sampling techniques that can bias simulations to increase the observation of rare events.

Current approaches to automatic CV discovery can be categorized into two main methodologies: approaches that treat data as *i.i.d.* points and employ techniques from PCA to deep autoencoders, and time-dependent approaches that model the system as a Markov process, exemplified by methods like TICA (time-lagged independent component analysis) and VAMP (variational approach for Markov processes). A fundamental challenge in this field is the *chicken-and-egg* problem, where effective CV discovery requires good sampling of the phase space, but good sampling typically requires knowledge of appropriate CVs [Bhakat, 2022].

Recent breakthroughs in transformer architectures for protein structure generation [Jumper et al., 2021] suggest the possibility of modeling temporal dynamics without explicit Markovian assumptions. This development opens up new avenues for CV discovery, particularly the potential that architectures capable of generating realistic molecular trajectories might inherently learn representations that could serve as useful and/or interpretable CVs.

Preprint. Under review.

2 Related Work

Two works have explored autoregressive models for the collective variable problem specifically: Tsai et al. [2020] demonstrated an LSTM-based approach for learning molecular dynamics with simple language models, while Zeng et al. [2021] compared LSTM and transformer performance on rare events. However, both works used relatively simple architectures with mixed results.

Since then, there has been significant development in the use of transformers for protein structure generation [Wu et al., 2024, Lu et al., 2024, Lin et al., 2022]; Wu et al. [2024] specifically demonstrated how angular representations and specialized transformer architectures can effectively model molecular structures while maintaining physical interpretability. Additionally, concurrent developments in the use of transformers for time-series forecasting [Zeng et al., 2023], autonomous driving [Zhang et al., 2022], and robotics [Mirchandani et al., 2023] all essentially deal with trajectories in continuous space.

Finally, Peebles and Xie [2023] use a vision transformer as the denoiser in a latent diffusion setup, where the latent space is continuous and the images are *transformed* to sequences via *patching*, a linear projection of a flattened two-dimensional patch into a one-dimensional token.

3 The data

In this work, we desired to have control over the data generation and satisfy the following constraints:

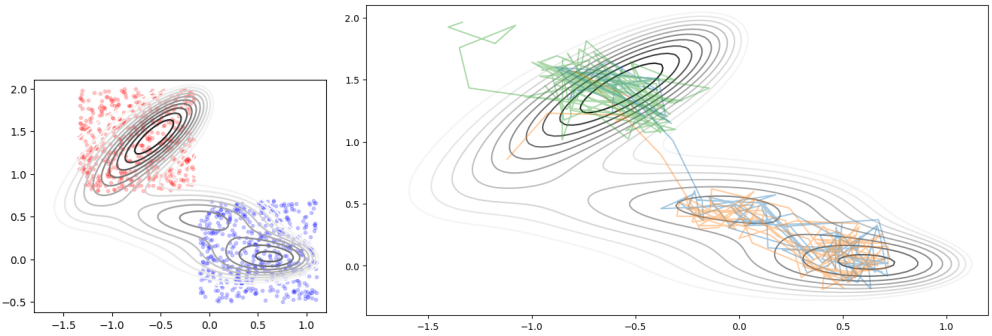
- Since this is a relatively novel application, using a well-understood system makes interpreting the performance of the model much easier.
- Due to the computational demands, using a simple system ensures the models are properly trained.
- We wish to be able to generate different versions of the data to better evaluate the model performance landscape with regards to the model complexity, for instance.
- Since the architectures operate on a *trajectory*-basis, we wish to have a significant number of trajectories available for training. This is often an issue with using data from more complex chemical systems, since chemists tend to focus on a few very long trajectories, usually hoping to observe the so-called *rare events*. Concatenating different datasets is also not feasible since the simulation parameters for the freely available data rarely coincide.

Thus we have settled on using the benchmark analytic *Müller-Brown potential* [Müller and Brown, 1979]: a particle traveling in a two-dimensional potential with two minima, separated by an energy barrier.

We generate several sets of data ourselves using Langevin dynamics, modifying the diffusion constant and the temperature. For the results presented, we have increased the diffusion constant to 100 (while setting the temperature to 300K) so that the trajectories become less predictable and the problem becomes harder for the models; thus, the effect of model size can be evaluated. Note that this entails a degree of sacrifice in the physical realism.

The dataset for the results thus consists of a train and test set, each composed of a 1000 two-dimensional trajectories that are 1000 steps long.

Additionally, in the infinite limit (with a reasonable diffusion coefficient) the vast majority of trajectories started from a vicinity of global minimum (which represents a *stable state*) will end up in that minimum; hence we can assign labels to trajectories in the following way: select a starting position uniformly randomly from a square centered around the minima and assign the corresponding label to the trajectory. This, as well as some generated trajectories, is illustrated in Figure 1. Note however, that there is inherently a degree of noise in such labels since we only run the simulation for a finite duration.



(a) Starting positions. The colors represent the assigned classes.

(b) Example generated trajectories. Every fifth step shown for presentation purposes.

Figure 1: Data simulated from the Müller-Brown potential.

4 Methodology

4.1 Model Architecture, training, and generation

Following an example that used a BERT [Devlin et al., 2019] encoder followed by a custom network that predicted torsional angles for the denoising part of a diffusion model [Wu et al., 2024], we use the same general idea while redesigning the model from scratch for trajectory modeling.

As illustrated in Fig. 2, our model uses a standard *Huggingface* implementation of the BERT encoder [Huggingface, 2024] as its central part.

On the input side, the usual tokenization and word embedding is replaced by a direct linear projection into the model's `hidden_size` dimension, which makes it compatible with the base model. Positional embeddings are retained as-is from the base model and added to the input projection. Additionally, the data is standardized on the input and unstandardized on generation (either based on the training data or the "prompt").

This is followed by the base BERT encoder. Its complexity is used to classify the three models for which the results are presented:

- *Small model:* four hidden layers, eight attention heads
- *Medium model:* eight hidden layers, sixteen attention heads
- *Large model:* twelve hidden layers, twenty-four attention heads.

The total number of parameters for the three models is presented in Tab. 1. Note that by opting for a

	Small Model	**Medium Model**	**Large Model**
`inputs_to_hidden_dim`	48	192	360
`embeddings`	16.4 K	65.7 K	123 K
`encoder`	21.6 K	399 K	2.2 M
`feature_predictor`	338	4.4 K	15.0 K
`state_classifier`	338	4.4 K	15.0 K
Trainable Params	**38.7 K**	**474 K**	**2.3 M**

Table 1: Model Architecture Summary

BERT-based encoder, we employ bidirectional attention: for a single step in the trajectory, all other steps are attended to.

Finally, the encoder embeddings are used for two "heads": a coordinate prediction head f that maps back to $\mathbb{R}^d$ and a state classification head g; both are implemented as an MLP.

On a single trajectory level, we define the following loss for the "regression" part of the model:

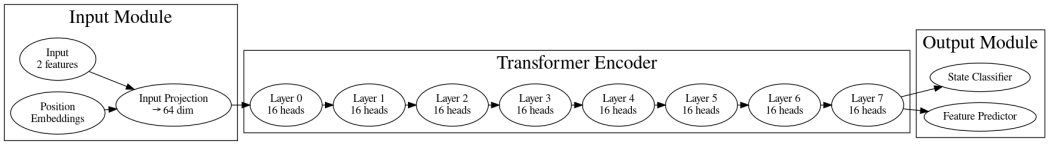


Figure 2: The architecture of a medium-sized model.

$$\mathcal{L}_{\text{traj}} = \frac{1}{N-1} \sum_{t=1}^{N-1} \|f(x_{1:N-1})_t - x_{t+1}\|^2 \tag{1}$$

To take advantage of the bidirectional attention, the model is trained to *"shift" the trajectory one step forward*: given all but the last step of the input $x_{1:N-1}$, the output is treated as $x_{2:N}$. The loss is then the average between the corresponding input-output pairs (where the last one is the one that the model did not see in the input).

For the classifier g, the whole trajectory serves as the input. Thus, again on a single trajectory level, the loss is a combination of the "regression" and classification losses:

$$\mathcal{L} = \gamma\, \mathcal{L}_{\text{traj}} + \lambda\, \text{CrossEntropy}(g(x_{1:N}), y), \tag{2}$$

where y is the label assigned as described in Sec. 3.

The goal of using a common encoder with two weakly related goals is to (hopefully) encourage a form of *representation learning*: having to satisfy both of these objectives simultaneously, we hope that the embeddings will correspond to some underlying physical reality as opposed to the model taking "shortcuts" which would be likelier were only a single objective optimized.

For generation, we maintain a rolling window of W steps that serve as "prompts"; in the simplest case, only the last output is appended to the prompt with the first input dropped:

$$f([x_1, \cdots, x_W]) = [y_1, \cdots, y_N] \to f([x_2, \cdots, y_W]) = \dots$$

Interestingly, "predicting" multiple steps at once seems to sometimes work, as well: pretend the last n steps of predicted output are actually predictions for the next n steps (as opposed to $[N-n+1, ...]$), append to the input, and repeat:

$$f([x_1, \cdots, x_W]) = [y_1, \cdots, y_N] \to f([x_{2+n}, \cdots, y_{W-n}, \cdots, y_W]) = \dots \tag{3}$$

Some clues as to why this works will be given in Sec. 5.3.

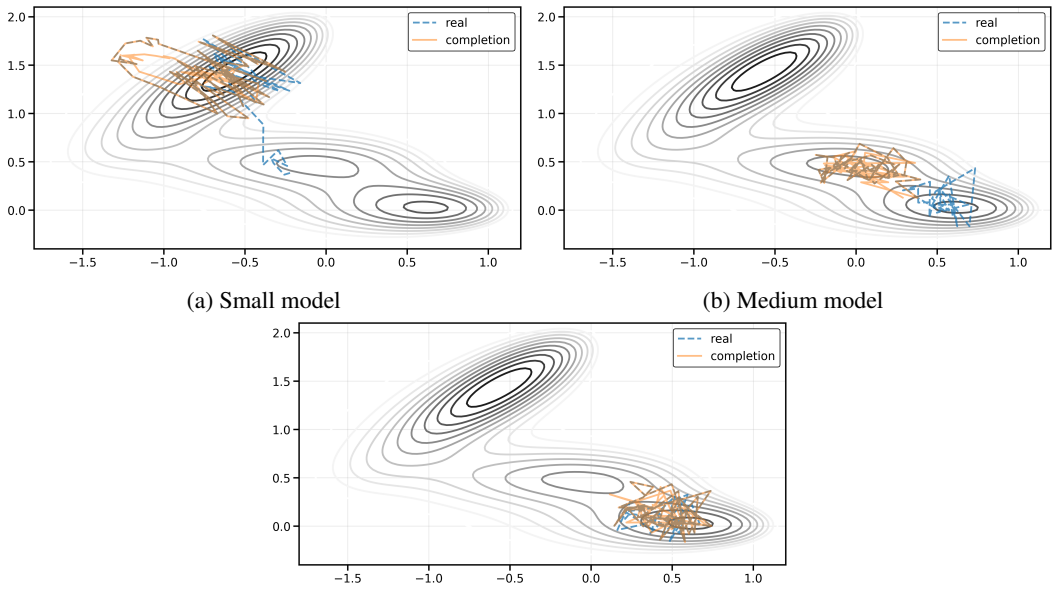
4.2 Evaluation

The model setup thus suggests three ways of evaluation:

1) The quality of the generated trajectories.
2) Prediction accuracy on the metastable state labels.
3) Analysis of the attention patterns.

The first point uses the model as a pure generative model, with the evaluation akin to that done for protein structure generation. For the second point, we are taking an approach often used with supervised CV discovery methods: after being trained with labels, the model as a whole is used a CV.

On the last point, we are building on existing work that examines attention patterns, such as Vig and Belinkov [2019] and the related `BERTViz` tool, as well as Voita et al. [2019], Clark et al. [2019]. Specifically, from the former we adapt the notion of *attention distance* to try to determine whether the model is learning an intrinsic scale on which the dynamics happen.



(a) Small model

(b) Medium model

(c) Large model

Figure 3: Trajectory generation: the initial "prompt" is represented by the coinciding dashed blue and solid orange lines.

5 Results

5.1 Generation

First, we show a few representative generated trajectories for the three models in Fig. 3.

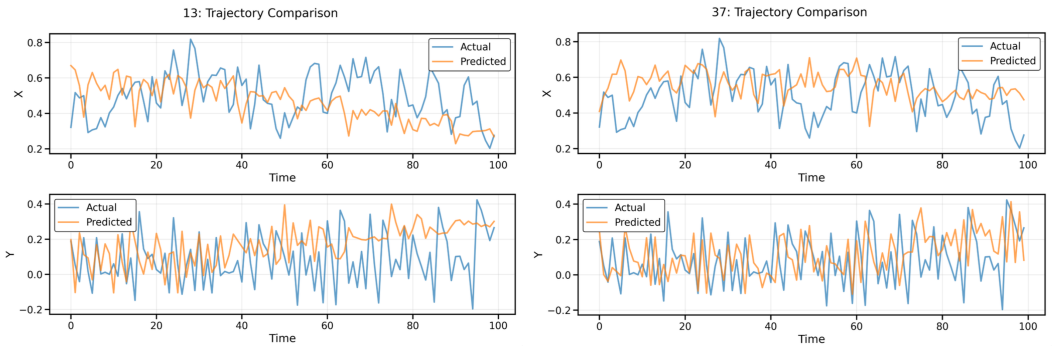
We note that the small model has some trouble learning the dynamics, as it generates steps that take the trajectory out of the vicinity of the minima. This is not the case with the bigger models; while the trajectory generated by the medium-sized model stays in the local minimum while the true trajectory finds the nearby global one, this is still physically realistic behavior. The large model trajectory typically follows the *behavior* of the test trajectory, as illustrated in both progressively concentrating on the global minimum of the second state.

As for quantitative metrics, the MSE on a 100 steps when prompted from the first 500 steps from the test set is presented in Tab. 2. Note however, that physically plausible trajectories might still carry a significant MSE, so measures such as this do not paint the whole picture.

	Small	Medium	Large
MSE	0.119	0.112	0.114

Table 2: Mean squared error on generated trajectories across model sizes

Interestingly, when using the procedure to predict multiple steps at once as described in Eq. 3, taking more steps (up to a point) empirically often seems to generate more realistic behavior, as illustrated in Fig. 4, where the case with fewer steps deteriorates towards a constant prediction.



(a) Predicting 13 steps (b) Predicting 37 steps

Figure 4: X and Y coordinate prediction when predicting 13 steps (left) and 37 steps (right) at a time.

5.2 State classification

The stable state classification results are presented in Tab. 3.

	Accuracy	Precision	Recall	F1 Score
Small Model	0.9150	0.9234	0.9150	0.9154
Medium Model	0.9780	0.9788	0.9780	0.9780
Large Model	0.9910	0.9910	0.9910	0.9910

Table 3: Classification Metrics

While the trajectory MSE might not indicate good performance on the face of it (for the reasons discussed), the very strong classification accuracy indicates that the embeddings contain almost all the information needed to classify the state. Also note that the labels themselves might be noisy since they're based on the vicinity of the starting state to the nearest minima; thus the largest model exhibits essentially perfect classification accuracy.

Classification accuracy deteriorates when prompted with a later trajectory slice. After dividing the length-1000 test sequences into segments of equal length, we obtain the results shown in Fig. 5.

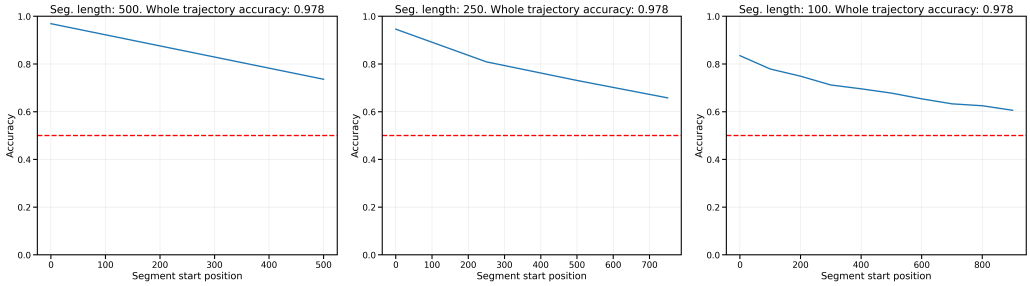


Figure 5: Medium model - classification accuracy by sequence slice (left to right): 2 sequences of length 500, 4 sequences of length 250, 10 sequences of length 100.

We notice that both sequence length and position matter; moreover, the earlier sequences are more predictable of the state. This is reasonable since the classifier is trained with the whole trajectory of length 1000 (cf. Eq. 2) and the starting position is, indeed, the best indicator of the label as this was the rationale for assigning the labels in the first place. However, note that even for smallest and most late slices the accuracy remains above random guessing. Additionally, the accuracy on the *second slice* (which naturally does not contain the starting position) is still very good, thus the classifier does not predict based on the starting state alone.

5.3 Learning the structure of the data

Finally, we wish to examine the models' attention patterns with hope of gleaning information about the internal representations of the models, focusing on evidence of learned physical properties.

The figures presented all feed the model with a single, random, trajectory from the test set. For presentation purposes, we slice the trajectories into 10 parts of length 100 and average over them, unless noted otherwise.

First, we examine the last layer attention for each of the heads of the smallest model in Fig. 6.

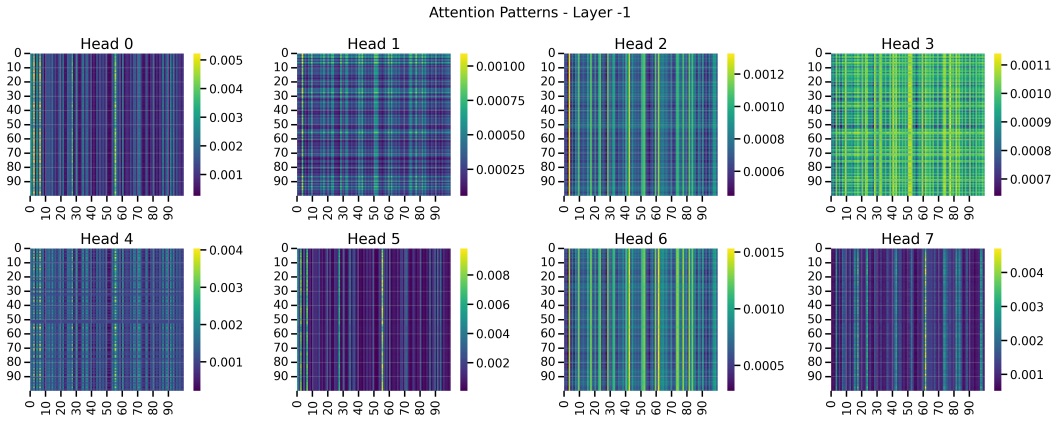
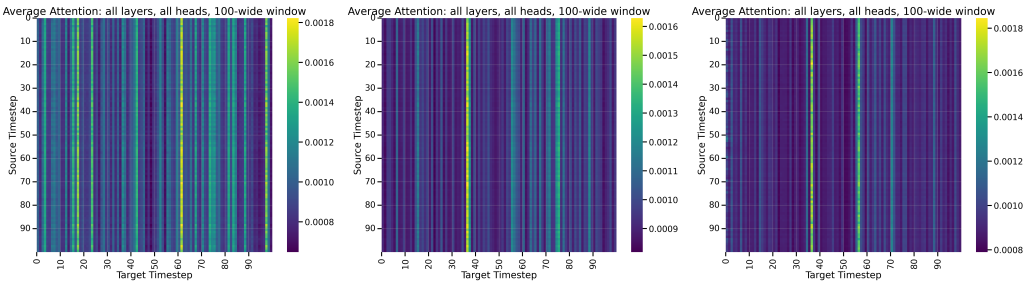


Figure 6: Small model: last layer per head attention patterns. Query on the y axis, key on the x axis.

We notice that most of the heads *focus on a discrete set of keys (timesteps)* as indicated by the bright vertical stripes. Furthermore, these are almost equally spaced. The exception are heads 1, 3, and 4, where the discretization works both ways.

This pattern persists for all the models even if we average over all layers and heads, as illustrated in Figure 7.



(a) Small model (b) Medium model (c) Large model

Figure 7

This leads us to conclude the following: the models learn a form of *coarse graining*. In each slice, the model identifies a couple of *key* (pun intended) timesteps that are most informative; examining the structure of attention on these steps additionally shows further discretization since some steps reference them much more than others. Additionally, the *larger* the model, the *fewer* key timesteps it needs to rely on, on average. What is not present are any kind of block patterns that would indicate grouping of timesteps into temporally localized groups that mostly interact within themselves.

Note however, that the number and identity of the key steps within each window might vary; these gets blurred when averaging over the 100-length windows (since there's no reason to believe the attention to be equivariant w.r.t 100-step translation). Thus the top row in Fig. 8 breaks up the prompt trajectory into sub-sections and shows that the behavior of discretizing into key steps is consistent, but the steps might vary. For the very last time-slice, the discretization is two-way and intense:

only a handful of query-key pairs matter. In general, the discretization is likely also the reason that generating multiple steps at once (cf. Eq. 3) works up to a reasonable number of steps since the *key steps* mostly remain in the prompt throughout.

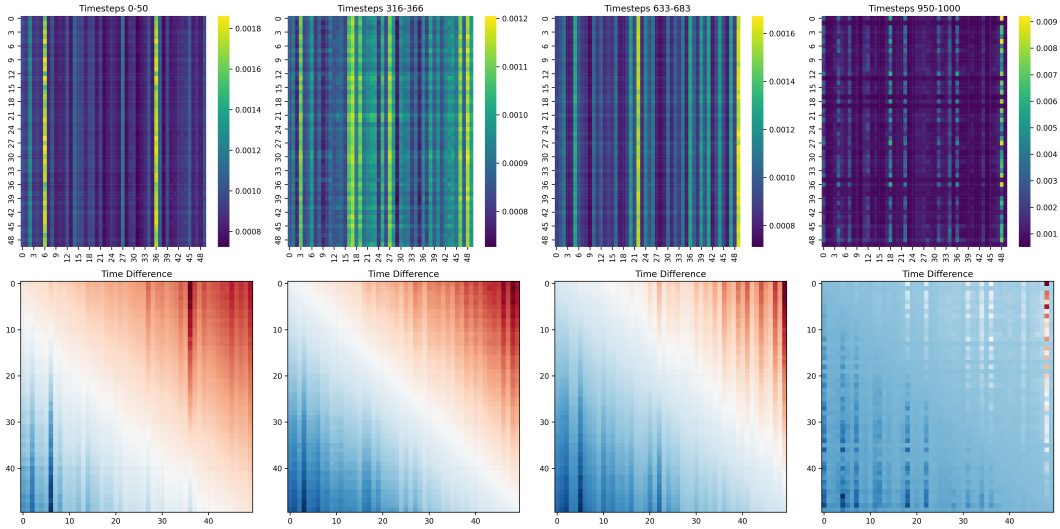


Figure 8: Large model. Top - per segment attention patterns. Bottom: attention patterns weighted by the time difference: future and past.

The bottom row of Fig. 8 additionally weights the attention by the time difference between the query and key timesteps. After accounting for the discussed discretization patterns, the attention seems to be evenly spread across time (the gradient of color intensity seems constant). This again illustrates that even after examining sub-trajectories, there seems to be no indication of preference to local timesteps, with, for the last slice, the initial states mostly attending to the very last states. This is a further argument for the coarse discretization being almost completely informative.

The deeper meaning of such discretization is the following: while the granularity of the data is chosen primarily by the requirements of stable simulation, at least for the objectives captured in the loss (cf. Eq. 2), the *semantic granularity* seems to be *much coarser*. The fact that larger models utilize fewer key steps than smaller reveals the dependence of the apparent semantic granularity on the model's ability to extract and represent information. While such time-domain coarse-graining is not a full CV in itself, it is the first step towards such since the purpose of CVs is to capture the longer-term dynamics while ignoring local fluctuations, and is an object of interest in itself [Fu et al., 2023].

6 Conclusions

In this work, we have implemented a Transformer encoder architecture designed for generating molecular dynamics trajectories. Learning physically meaningful embeddings was encouraged by combining the generation accuracy loss with a classification objective that both take the same embeddings as the input. The results show that for moderate model sizes, the architecture is able to generate physically plausible trajectories with excellent classification accuracy.

Examining the attention patterns, we discover that the model learns a *coarse-grained* time discretization by identifying a small number of *key steps* for every part of the trajectory. Discovering such timescales is also of potential relevance to the scientific community.

As for learning CVs in the vein of representation learning, the approach seems to be limited by the fact that the Transformer encoder does not perform any transformation of the input features; as a first step, the architecture could be combined with additional encoder modules to work in the latter's latent space, thus combining the time and space aspects of condensing the dynamics. Additionally, the inherent quadratic complexity of the attention mechanism and the need for reasonably large models would perhaps necessitate a shift to models such as state-space models [Gu and Dao, 2023] for the approach to be feasible for larger, scientifically interesting systems.

References

Soumendranath Bhakat. Collective variable discovery in the age of machine learning: reality, hype and everything in between. *RSC Advances*, 12(38):25010–25024, 2022. doi: 10.1039/D2RA03660F. URL https://pubs.rsc.org/en/content/articlelanding/2022/ra/d2ra03660f. Publisher: Royal Society of Chemistry.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What Does BERT Look At? An Analysis of BERT's Attention, June 2019. URL http://arxiv.org/abs/1906.04341. arXiv:1906.04341 [cs].

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL https://arxiv.org/abs/1810.04805.

Xiang Fu, Tian Xie, Nathan J. Rebello, Bradley D. Olsen, and Tommi Jaakkola. Simulate Time-integrated Coarse-grained Molecular Dynamics with Multi-Scale Graph Networks, August 2023. URL http://arxiv.org/abs/2204.10348. arXiv:2204.10348 [physics].

Albert Gu and Tri Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces, December 2023. URL http://arxiv.org/abs/2312.00752. arXiv:2312.00752 [cs].

Huggingface. Transformers: State-of-the-art machine learning for NLP, CV, and more, 2024. URL https://github.com/huggingface/transformers. Accessed: 2024-12-04.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021. doi: 10.1038/s41586-021-03819-2. URL https://doi.org/10.1038/s41586-021-03819-2. ISBN: 1476-4687.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. Evolutionary-scale prediction of atomic level protein structure with a language model, December 2022. URL https://www.biorxiv.org/content/10.1101/2022.07.20.500902v3. Pages: 2022.07.20.500902 Section: New Results.

Jiarui Lu, Xiaoyin Chen, Stephen Zhewen Lu, Chence Shi, Hongyu Guo, Yoshua Bengio, and Jian Tang. Structure Language Models for Protein Conformation Generation, October 2024. URL http://arxiv.org/abs/2410.18403. arXiv:2410.18403.

Suvir Mirchandani, Fei Xia, Pete Florence, Brian Ichter, Danny Driess, Montserrat Gonzalez Arenas, Kanishka Rao, Dorsa Sadigh, and Andy Zeng. Large Language Models as General Pattern Machines, October 2023. URL http://arxiv.org/abs/2307.04721. arXiv:2307.04721 [cs].

Klaus Müller and Leo D. Brown. Location of saddle points and minimum energy paths by a constrained simplex optimization procedure. *Theoretica chimica acta*, 53(1):75–93, March 1979. ISSN 1432-2234. doi: 10.1007/BF00547608. URL https://doi.org/10.1007/BF00547608.

William Peebles and Saining Xie. Scalable Diffusion Models with Transformers, March 2023. URL http://arxiv.org/abs/2212.09748. arXiv:2212.09748 [cs].

Sun-Ting Tsai, En-Jui Kuo, and Pratyush Tiwary. Learning molecular dynamics with simple language model built upon long short-term memory neural network. *Nature Communications*, 11(1):5115, October 2020. ISSN 2041-1723. doi: 10.1038/s41467-020-18959-8. URL https://www.nature.com/articles/s41467-020-18959-8. Publisher: Nature Publishing Group.

Jesse Vig and Yonatan Belinkov. Analyzing the Structure of Attention in a Transformer Language Model, June 2019. URL http://arxiv.org/abs/1906.04284. arXiv:1906.04284.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned, June 2019. URL http://arxiv.org/abs/1905.09418. arXiv:1905.09418 [cs].

Kevin E. Wu, Kevin K. Yang, Rianne van den Berg, Sarah Alamdari, James Y. Zou, Alex X. Lu, and Ava P. Amini. Protein structure generation via folding diffusion. *Nature Communications*, 2024.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are Transformers Effective for Time Series Forecasting? *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(9):11121–11128, June 2023. ISSN 2374-3468. doi: 10.1609/aaai.v37i9.26317. URL https://ojs.aaai.org/index.php/AAAI/article/view/26317. Number: 9.

Wenqi Zeng, Siqin Cao, Xuhui Huang, and Yuan Yao. A Note on Learning Rare Events in Molecular Dynamics using LSTM and Transformer, July 2021. URL http://arxiv.org/abs/2107.06573. arXiv:2107.06573 [physics].

Kunpeng Zhang, Xiaoliang Feng, Lan Wu, and Zhengbing He. Trajectory Prediction for Autonomous Driving Using Spatial-Temporal Graph Attention Transformer. *IEEE Transactions on Intelligent Transportation Systems*, 23(11):22343–22353, November 2022. ISSN 1558-0016. doi: 10.1109/TITS.2022.3164450. URL https://ieeexplore.ieee.org/document/9768029/?arnumber=9768029. Conference Name: IEEE Transactions on Intelligent Transportation Systems.