we can generate B chains in parallel, all starting from the same initial state. This batched sampling procedure leads to even further speedups. For all exploration experiments we use a batch size of 100, and run $M = 10000$ exploration steps. The maximum allowed energy change cutoff is set at $\Delta U_{\max} = 300$kJ/mol.

D. Dataset details

We evaluate our model on three different datasets, AD, 2AA, and 4AA, as introduced in Section 6. All datasets are simulated in implicit solvent using the `openMM` library (Eastman et al., 2017). For all MD simulations we use the parameters shown in Table 2.

Table 2. OpenMM MD simulation parameters

Force Field	amber-14
Time step	0.5fs
Friction coefficient	$0.3\frac{1}{\text{ps}}$
Temperature	310K
Integrator	LangevinMiddleIntegrator

E. Hyperparameters

Depending on the dataset, different Timewarp model sizes were used, as shown in Table 3. For all datasets the Multihead kernel self-attention layer consists of 6 heads with lengthscales $\ell_i = \{0.1, 0.2, 0.5, 0.7, 1.0, 1.2\}$, given in nanometers.

Table 3. Timewarp model hyperparameters

Dataset	RealNVP layers	Transformer layers	Parameters	Atom-embedding dim H	Transformer feature dimension D
AD	12	6	1×10^8	64	128
2AA	12	6	1×10^8	64	128
4AA	16	16	4×10^8	128	128

The ϕ_{in} and ϕ_{out} MLPs use SiLUs as activation functions, while the Transformer MLPs use ReLUs. Note the transformer MLP refers to the atom-wise MLP shown in Figure 2, Middle inside the transformer block. The shapes of these MLPs vary for the different datasets as shown in Table 4.

Table 4. Timewarp MLP layer sizes

Dataset	ϕ_{in} MLP	ϕ_{out} MLP	Transformer MLP
AD	$[70, 256, 128]$	$[128, 256, 3]$	$[128, 256, 128]$
2AA	$[70, 256, 128]$	$[128, 256, 3]$	$[128, 256, 128]$
4AA	$[134, 2048, 128]$	$[128, 2048, 3]$	$[128, 2048, 128]$

The first linear layers in the kernel self-attention module always has the shape $[128, 768]$ (in Section 4 denoted as V), and the second (after concatenating the output of head head) has the shape $[768, 128]$.

After likelihood training, we fine-tune the model for the AD and 2AA dataset with a combination of all three losses discussed in Section 5. We did not perform fine tuning for the model trained on the 4AA dataset. We use a weighted sum of the losses with weights detailed in Table 5. We use the *FusedLamb* optimizer and the *DeepSpeed* library (Rasley et al., 2020) for all experiments. The batch size as well as the training times are reported in Table 6. All simulations are started with a learning rate of 5×10^{-4}, the learning rate is then consecutively decreased by a factor of 2 upon hitting training loss plateaus.