Aligned Diffusion Schrödinger Bridges (Supplementary Material)

Vignesh Ram Somnath[*1,2] **Matteo Pariset**[*1,3] **Ya-Ping Hsieh**[1]
Maria Rodriguez Martinez[2] **Andreas Krause**[1] **Charlotte Bunne**[1]

[1]Department of Computer Science, ETH Zürich
[2]IBM Research Zürich
[3]Department of Computer Science, EPFL

1 ADDITIONAL RESULTS

1.1 VARIANCE REDUCTION

In this paragraph, we elaborate on the need to parametrize also Doob's h_t function, along with the drift b_t. Introducing m^ϕ removes the need to evaluate (11) which is difficult to approximate in practice on high-dimensional spaces. This equation amounts, in fact, to a Gaussian Kernel Density Estimation of the conditional probability $\mathbb{P}(X_1 = \mathbf{x}_1 | X_t = \mathbf{x})$ along (unconditional) paths obtained from (5). Faithful approximations of (11) would, therefore, require:

- good-quality paths, which are scarce at the beginning of training when the drift b_t^θ has not yet been learned;
- exponentially many trajectories (in the dimension of the state space);
- that points x_1 (obtained from conditional trajectories, Eq. 6) be reasonably close to x_1 (obtained from unconditional trajectories, Eq. 5);

Even if all the above conditions were satisfied, the quantity $h_t(x) = \mathbb{P}(X_1 = \mathbf{x}_1 | X_t = \mathbf{x})$ would still be challenging to directly manipulate. It is, in fact, much smaller at earlier times t (see Table 1), since knowledge of the far past has a weaker influence on the location X_1 of particles at time $t = 1$. Precision errors at $t \approx 0$ would then be amplified when computing the score of h_t (i.e, $\nabla \log h_t$) –which appears in the loss (8)– and accumulate over timesteps, eventually leading trajectories astray. By directly parameterizing the score, we instead sidestep this problem. The magnitude of $m_t^\phi \approx \nabla \log h_t$ can, in fact, be more easily controlled and regularized.

	Time t						
	0	**0.15**	**0.30**	**0.45**	**0.60**	**0.75**	**0.90**
Mean $h_{t,\tau}$ value	2.92e-14	4.03e-13	2.54e-11	1.72e-09	1.47e-07	2.66e-05	8.53e-3

Table 1: Average $h_{t,\tau}$ values along paths, at different timesteps. $\mathbb{P}(X_1 = \mathbf{x}_1 | X_t = \mathbf{x})$ ranges over 11 orders of magnitude across the time interval and is smallest when $t \approx 0$.

1.2 RIGID PROTEIN DOCKING

Baselines. We compare our method to EQUIDOCK as well as traditional docking software including ATTRACT [Schindler et al., 2017, de Vries et al., 2015], HDOCK [Yan et al., 2020], CLUSPRO [Desta et al., 2020, Kozakov et al., 2017], and PATCHDOCK [Mashiach et al., 2010, Schneidman-Duhovny et al., 2005]. As mentioned in the paragraph above, for

[*]Equal contribution.

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

Table 2: **Rigid docking results.** Complex and interface RMSD between predicted and true bound structures (after Kabsch alignment). * denotes methods for which we use values directly from [Ganea et al., 2022]. All other results show the performance on our test set.

	DB5.5 Test Set					
	Complex RMSD			Interface RMSD		
Methods	Median	Mean	Std	Median	Mean	Std
ATTRACT*	9.55	10.09	9.88	7.48	10.69	10.90
HDOCK*	0.30	5.34	12.04	0.24	4.76	10.83
CLUSPRO*	3.38	8.25	7.92	2.31	8.71	9.89
PATCHDOCK*	18.26	18.00	10.12	18.88	18.75	10.06
EQUIDOCK*	14.13	14.72	5.31	11.97	13.23	4.93
EQUIDOCK	14.12	14.73	5.31	11.97	13.23	4.93
SBALIGN	6.59	6.69	2.04	7.69	8.11	2.39

ligands in the test set, we generate the corresponding unbound versions by applying the rotation and translation sampled during training. We evaluate the trained model from EQUIDOCK and SBALIGN on these unbound structures and report corresponding evaluation metrics. For the remaining baselines, we include the numbers from [Ganea et al., 2022]. These baselines typically sample several candidate complexes by considering small increments of rotation angles. We expect this makes them somewhat invariant to arbitrary initialization, and the corresponding docking scores to not be severely impacted.

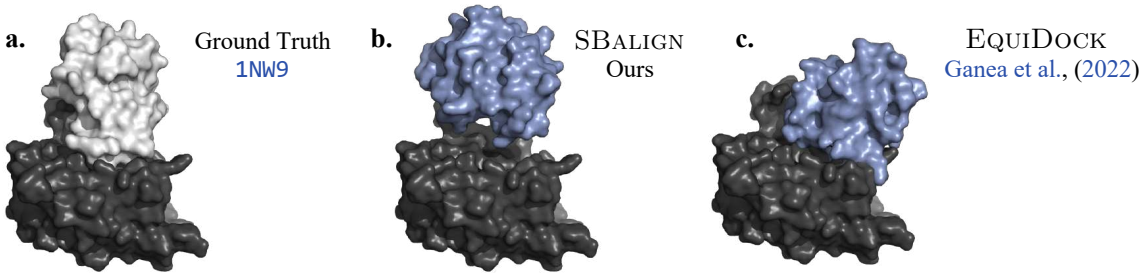

Figure 1: Ground truth and predicted bound structures for the complex with PDB ID: 1NW9. SBALIGN is able to identify the true binding pocket.

Results. The model performance is summarized in Table 2. Our method SBALIGN considerably outperforms EQUIDOCK across all metrics. SBALIGN also achieves comparable or better performance than traditional docking software without relying on extensive candidate sampling and re-ranking or learning surface templates from parts of the current test set. An example of docked structures, in direct comparison with EQUIDOCK is displayed Fig. 5. Beyond the results in Table 2, we display the ground truth and docked complexes in Figs. 5, 1, and 2.

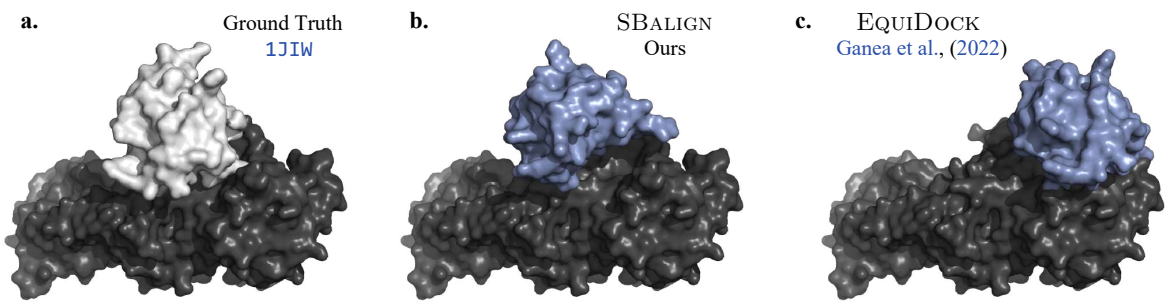

Figure 2: Ground truth and predicted bound structures for the complex with PDB ID: 1JIW. SBALIGN is able to find the true binding interface compared to EQUIDOCK.

2 DATASETS

2.1 SYNTHETIC DATASETS

In the following, we provide further insights and experimental results in order to access the performance of SBALIGN in comparison with different baselines and across tasks of various nature. For each dataset, we describe in detail its origin as well as preprocessing and featurization steps.

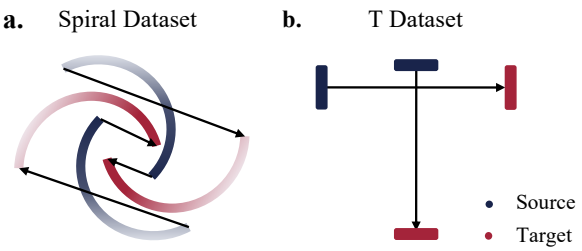

Figure 3: Initial (*blue*) and final (*red*) marginals for the two toy datasets **(a)** moon and **(b)** T, together with arrows indicating a few alignments

Moon dataset. The `moon` toy dataset (Fig. 3a) is generated by first sampling $\hat{\mathbb{P}}_1$ and then applying a clockwise rotation of 233° around the origin to obtain $\hat{\mathbb{P}}_0$. The points on the two semi-circumferences supporting $\hat{\mathbb{P}}_1$ are initially placed equally-spaced along each semi-circumference and then moved by applying additive Gaussian noise to both coordinates. While classic generative models will choose the shortest path and connect ends of both moons closest in Euclidean distance, only methods equipped with additional knowledge or insight on the intended alignment will be able to solve this task.

T dataset. This toy dataset (Fig. 3b) is generated by placing an equal amount of samples at each of the four extremes of a T-shaped area having ratio between x and y dimensions equal to 51/55. If run with a Brownian prior, classical Schrödinger bridges (SBs) also fail on this dataset because they produce swapped pairings: i.e., they match the left (*resp.* top) point cloud with the bottom (*resp.* right) one. At the same time, though, this dataset prevents reference drifts with simple analytical forms (such as spatially-symmetric or time-constant functions) from fixing classical SBs runs. It therefore illustrates the need for general, plug-and-play methods capable of generating approximate reference drifts to use in the computation of classical SBs.

2.2 CELL DIFFERENTIATION DATASETS

Dataset description. We obtain the datapoints used in our cell differentiation task from the dataset generated by Weinreb et al. [2020], which contains 130861 observations/cells. We follow the preprocessing steps in Bunne et al. [2021] and use the Python package `scanpy` [Wolf et al., 2018]. After processing, each observation records the level of expression of 1622 different highly-variable genes as well as the following meta information per cell:

- a `timestamp`, expressed in days and taking values in {2, 4, 6};
- a `barcode`, which is a short DNA sequence that allows tracing the identity of cells and their lineage by means of single-cell sequencing readouts;
- an additional `annotation`, which describes the current differentiation fate of the cell.

Dataset preparation. We only retain cells with barcodes that appear both on days 2 and 4, taking care of excluding cells that are already differentiated on day 2. We construct matchings by pairing cells measured at two different times but which share the barcode. Additionally, we filter cells to make sure that no one appears in more than one pair. To reduce the very high dimensionality of these datapoints, we perform a PCA projection down to 50 components.

We end up with a total of 4702 pairs of cells, which we partition into train, validation, and test sets according to the split 80%/10%/10%.

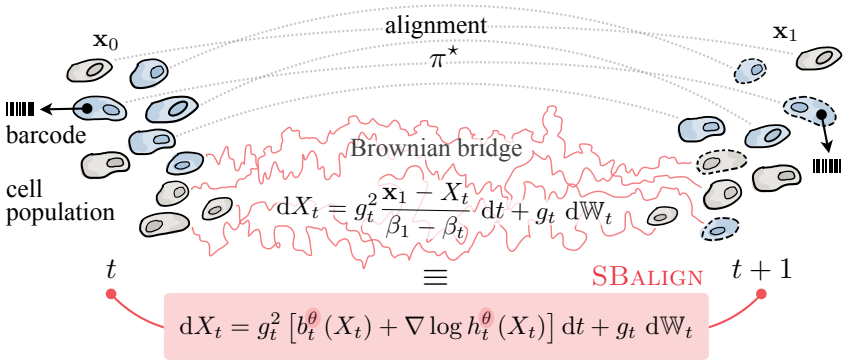

Figure 4: Overview of SBALIGN in the setting of cell differentiation with the goal of learning the evolutionary process that morphs a population from its stat at t to $t+1$. Through genetic tagging (i.e., barcodes) we are able to trace progenitor cells at time point t into their descendants $t+1$. This provides us with an alignment between populations at consecutive time steps. Our goal is then to recover a stochastic trajectory from $\mathbf{x}_0$ to $\mathbf{x}_1$. To achieve this, we connect the characterization of a SDE conditioned on $\mathbf{x}_0$ and $\mathbf{x}_1$ (utilizing the Doob's *h-transform*) with that of a Brownian bridge between $\mathbf{x}_0$ and $\mathbf{x}_1$ (classical Schrödinger bridge theory), leading to a simpler training procedure with lower variance and strong empirical results.

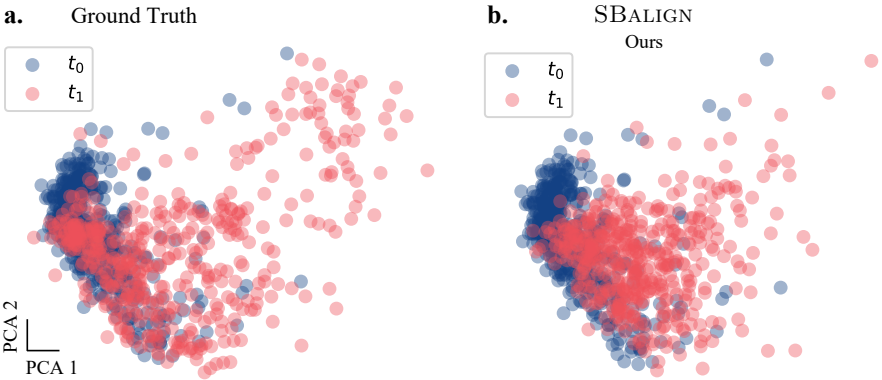

Figure 5: Distribution of the cell population (i.e., marginals) at time $t = t_0$ and $t = t_1$ for (**a**) the ground truth, and (**b**) SBALIGN, after projection along their first two principal components.

2.3 PROTEIN DOCKING

2.3.1 Conformational Changes

Dataset description. For the task of predicting protein conformational changes, we utilize the D3PM dataset. The dataset consists of both unbound and bound structures for 4330 proteins, under different types of protein motions. The PDB IDs were downloaded from `https://www.d3pharma.com/D3PM/`. For the PDB IDs making up the dataset, we download the corresponding (.cif) files from the Protein Data Bank.

Dataset preparation. For the scope of this work, we only focus on protein structure pairs, where the provided RMSD between the Cα carbon atoms is > 3Å, amounting to 2370 examples in the D3PM dataset. For each pair of structures, we first identify common residues, and compute the RMSD between Cα carbon atoms of the common residues after superimposing them using the Kabsch [Kabsch, 1976] algorithm, and only accept the structure if the computed Cα RMSD is within a certain margin of the provided Cα RMSD. The rationale behind this step was to only retain examples where we could reconstruct the RMSD values provided with the dataset. Common residues are identified through a combination of residue position and name. This step is however prone to experimental errors, and we leave it to future steps to improve the common residue identification step (using potentially, a combination of common subsequences and/or residue positions).

After applying the above preprocessing steps, we obtain a dataset with 1591 examples, which is then split into a train/valid/test

split of 1291/150/150 examples respectively. The structures used in training and inference are the Kabsch superimposed versions, therefore ensuring that the Brownian bridges are sampled between the unbound and bound states of the proteins, and no artifacts are introduced by 3D rotations and translations, which do not contribute to conformational changes.

Featurization. Following standard practice and for memory and computational efficiency, we only use the Cα coordinates of the residues to represent our protein structures instead of the full-atom structures. For each amino acid residue, we compute the following features: a one hot encoding of the amino acid identity f_e of size 23, hydrophobicity $f_h \in [-4.5, 4.5]$, volume $f_v \in [60.1, 227.8]$, the charge $f_c \in \{-1, 0, 1\}$, polarity $f_p \in \{0, 1\}$, and whether the amino acid residue is a hydrogen bond donor $f_d \in \{0, 1\}$ or acceptor $f_a \in \{0, 1\}$. The hydropathy and volume features are expanded into a radial basis with interval sizes 0.1 and 10 respectively. To equip the model with a notion of time, we use a sinusoidal embedding of time $\phi(t)$ of embedding dimensionality 32. These are concatenated to the amino acid features to form our input features for the amino acid residues. The edge features consist of a radial basis expansion of the distances between the residues. We also compute the spherical harmonics of the edge vectors between the residues, which is used in the tensor product message passing layers.

Position at t. For any time t, we sample the positions of the Cα atoms using the Brownian Bridge - given the coordinates $\mathbf{x}_0$ at $t = 0$ and the coordinates $\mathbf{x}_1$ at $t = 1$ with a Brownian bridge between $\mathbf{x}_0$ and $\mathbf{x}_1$, we know that $x_t \sim \mathcal{N}(x_t; (1-t)\mathbf{x}_0 + t\mathbf{x}_1, t(1-t))$.

2.3.2 Rigid Protein Docking

Dataset description. We use the DB5.5 dataset [Vreven et al., 2015] for our empirical evaluation. The DB5.5 dataset is a standard dataset used in protein-protein docking, however, it only has 253 complexes. The dataset was downloaded from `https://zlab.umassmed.edu/benchmark/`. The dataset consists of both unbound and bound structures, but the structures are largely rigid, with an average complex RMSD of 0.96 between the bound and unbound structures. We utilize the same splits as EquiDock [Ganea et al., 2022], with 203 complexes in the training set, 25 complexes in the validation set, and 25 complexes in the test set. For the evaluation in Table 2, we use the full DB5.5 test set. For ligands in the test set, we generate the corresponding unbound versions by applying the rotation and translation sampled during training.

Dataset preparation. Following similar convention as EQUIDOCK [Ganea et al., 2022], we treat the receptor as fixed. We use the same splits as EquiDock [Ganea et al., 2022], with 203 complexes in the training set, 25 complexes each in the validation and test sets. For each ligand, the final 3D structure corresponds to its bound version, and the unbound version is generated by applying a random rotation R and translation b to the bound version. However, applying a different rotation and translation to each ligand would result in a different Brownian bridge, thus providing limited learning signal for the drift b_t^θ. To avoid this, we create a rotation matrix R during training by sampling a random angle between $30 - 45°$ along each axis, and a translation b with a maximum magnitude between $5.0 - 10.0$. The same R and b are also applied to the validation and test sets. We leave it to future work to extend the algorithm to work for arbitrary rotations and translations.

Featurization. Following standard practice and for memory and computational efficiency, we only use the Cα coordinates of the residues to represent our protein structures instead of the full-atom structures. For each amino acid residue, we compute the following features: a one hot encoding of the amino acid identity f_e of size 23, hydrophobicity $f_h \in [-4.5, 4.5]$, volume $f_v \in [60.1, 227.8]$, the charge $f_c \in \{-1, 0, 1\}$, polarity $f_p \in \{0, 1\}$, and whether the amino acid residue is a hydrogen bond donor $f_d \in \{0, 1\}$ or acceptor $f_a \in \{0, 1\}$. The hydropathy and volume features are expanded into a radial basis with interval size 0.1 and 10 respectively. To equip the model with a notion of time, we use a sinusoidal embedding of time $\phi(t)$ of embedding dimensionality 32. These are concatenated to the amino acid features to form our input features for the amino acid residues.

Position at t. For any time t, we sample the positions of the Cα atoms using the Brownian Bridge - given the coordinates $\mathbf{x}_0$ at $t = 0$ and the coordinates $\mathbf{x}_1$ at $t = 1$ with a Brownian bridge between $\mathbf{x}_0$ and $\mathbf{x}_1$, we know that $x_t \sim \mathcal{N}(x_t; (1-t)\mathbf{x}_0 + t\mathbf{x}_1, t(1-t))$.

3 EXPERIMENTAL DETAILS

In the following, we provide further experimental details on the chosen evaluation metrics, network architectures, and hyperparameters.

3.1 EVALUATION METRICS

3.1.1 Cell Differentiation

For fairness of comparison between our method and the baseline (FBSB) —which only works at the level of distribution of cells— we also consider three evaluation metrics (i.e., W_ε, MMD and ℓ_2) that capture the similarity between the end marginal $\hat{\mathbb{P}}_1$ and our prediction $\pi_1^\star$, irrespective of matchings.

In what follows, we denote with $\hat{\nu}$ the predicted end marginal $\pi_1^\star$ —i.e., the predicted status of cells at day 4— and with ν the distribution of observed transcriptomes.

Wasserstein-2 distance. We measure accuracy of the predicted target population $\hat{\nu}$ to the observed target population ν using the entropy-regularized Wasserstein distance [Cuturi, 2013] provided in the `OTT` library [Bradbury et al., 2018, Cuturi et al., 2022] defined as

(1)

where $H(\mathbf{P}) := -\sum_{ij} \mathbf{P}_{ij}(\log \mathbf{P}_{ij} - 1)$ and the polytope $U(\hat{\nu}, \nu)$ is the set of $n \times m$ matrices $\{\mathbf{P} \in \mathbb{R}_+^{n\times m}, \mathbf{P}\mathbf{1}_m = \hat{\nu}, \mathbf{P}^\top \mathbf{1}_n = \nu\}$.

Maximum mean discrepancy. Kernel maximum mean discrepancy [Gretton et al., 2012] is another metric to measure distances between distributions, i.e., in our case between predicted population $\hat{\nu}$ and observed one ν. Given two random variables x and y with distributions $\hat{\nu}$ and ν, and a kernel function ω, Gretton et al. [2012] define the squared MMD as:

$$\text{MMD}(\hat{\nu}, \nu; \omega) = \mathbb{E}_{x,x'}[\omega(x, x')] + \mathbb{E}_{y,y'}[\omega(y, y')] - 2\mathbb{E}_{x,y}[\omega(x, y)].$$

We report an unbiased estimate of $\text{MMD}(\hat{\nu}, \nu)$, in which the expectations are evaluated by averages over the population particles in each set. We utilize the RBF kernel, and as is usually done, report the MMD as an average over the length scales: $2, 1, 0.5, 0.1, 0.01, 0.005$.

Perturbation signature ℓ_2. A common method to quantify the effect of a perturbation on a population is to compute its perturbation signature [Stathias et al., 2018, (PS)], computed via the difference in means between the distribution of perturbed states and control states of each feature, e.g., here individual genes. ℓ_2(PS) then refers to the ℓ_2-distance between the perturbation signatures computed on the observed and predicted distributions, ν and $\hat{\nu}$. The ℓ_2(PS) is defined as

$$\text{PS}(\nu, \mu) = \frac{1}{m}\sum_{y_i \in \nu} y_i - \frac{1}{n}\sum_{x_i \in \mu} x_i,$$

where n is the size of the unperturbed and m of the perturbed population. We report the ℓ_2 distance between the observed signature $\text{PS}(\nu, \mu)$ and the predicted signature $\text{PS}(\hat{\nu}, \mu)$, which is equivalent to simply computing the difference in the means between the observed and predicted distributions.

RMSD. To measure the quality of matchings sampled from SBALIGN $(\hat{x}_0^i, \hat{x}_1^i)$ —compared to the observed ones (x_0^i, x_1^i)— we compute:

$$\text{RMSD}(\{x_1^i\}^n, \{\hat{x}_1^i\}^n) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \|x_1^i - \hat{x}_1^i\|^2} \tag{2}$$

which, when squared, represents the mean of the square norm of the differences between predicted and observed statuses of the cells on day 4.

Cell type classification accuracy. We assess the quality of SBALIGN trajectories by trying to predict the differentiation fate of cells, starting from (our compressed representation of) their transcriptome. For this, we train a simple MLP-based classifier on observed cells and use it on the last time-frame of trajectories sampled from SBALIGN to infer the differentiation of cells on day 4. We use the classifier `MLPClassifier` offered by the library `scikit-learn` with the following parameters:

- 2 hidden layers, each with a hidden dimension of 50,
- the logistic function as non-linearity
- ℓ_2 norm, regularization with coefficient 0.1.

We report the subset accuracy of the predictions on the *test* set, measured as the number of labels (i.e., cell types) coinciding with the ground truth.

3.1.2 Rigid Protein Docking

We report two metrics, Complex Root Mean Square Deviation (Complex RMSD), and Interface Root Mean Square Deviation (Interface RMSD). Following [Ganea et al., 2022], the ground truth and predicted complex structures are first superimposed using the Kabsch algorithm [Kabsch, 1976], and the Complex RMSD is then computed between the superimposed versions. A similar procedure is used for computing Interface RMSD, but only using the residues from the two proteins that are within 8 Å of each other. Given a ligand with m residues and a receptor with n residues, we denote the predicted bound structures with $\mathbf{Z}' \in \mathbb{R}^{(n+m)}$ and the ground truth bound structure with $\mathbf{Z}^* \in \mathbb{R}^{(n+m)}$. We first superimpose the predicted and ground truth bound structures using the Kabsch algorithm and then compute the Complex RMSD as $\text{C}_{\text{rmsd}} = \sqrt{\frac{1}{n+m}\|\mathbf{Z}' - \mathbf{Z}\|_F}$. The Interface RMSD I_{rmsd} is computed similarly, but only using the residues from the two proteins that are within 8Å of each other.

3.2 NETWORK ARCHITECTURES

3.2.1 Cell Differentiation and Synthetic Datasets

We parameterize both $b^\theta(t, X_t)$ and $m^\phi(t, b_t, X_t)$ using a model composed of:

1. `x_enc`: 3-layer MLP performing the expansion of spatial coordinates (or drift) into hidden states (of dimension 64 to 256);
2. `t_enc`: sinusoidal embedding of time (on 64 to 256 dimensions), followed by a two layer MLP;
3. `mlp`: 3-layer MLP which maps the concatenation of embedded spatial and temporal information (output of modules 1 and 2 above) to drift magnitude values along each dimension.

After every linear layer (except the last one), we apply a non-linearity and dropout (level 0.1). In all the experiments, we set the diffusivity function $g(t)$ in (5) to a constant g, which is optimized (see § 3.3).

3.2.2 Protein Conformational Changes

As our architecture $b_t^\theta(X_t)$ suitable for approximating the true drift b_t, we construct a graph neural networks with tensor-product message passing layers using `e3nn` [Thomas et al., 2018, Geiger and Smidt, 2022]. To build the graph, we consider a maximum of 40 neighbors –located within a radius of 40Å for each residue. The model is SE(3) equivariant and receives node and edge features capturing relevant residue properties, and distance embeddings. For the baseline EGNN model, we consider the variant of the EGNN model proposed in Xu et al. [2022], owing to its strong performance on the molecule conformer generation task.

3.2.3 Rigid Protein Docking

For the scope of this paper, we use a MLP as b_t^θ and m^ϕ. As inputs, both b_t^θ and m^ϕ receive input node features and the C_α coordinates at time t, as described in Section 3.3.4, with m^ϕ receiving the prediction b_t^θ as additional input. Both models

have 3 hidden layers, each with a dimension of 64 and an output dimension of 3, with around 50K parameters in total. Our current architectures are not equivariant to global rotations and translations, which is a desirable property in protein docking as the structures of the proteins themselves are invariant to the choice of reference coordinates frames. We leave a thorough exploration of other architectures, such as equivariant GNN architectures similar to those adopted in [Ganea et al., 2022] to future work.

3.3 HYPERPARAMETERS

In the following, we will provide an overview of the selected hyperparameters as well as chosen training procedures.

3.3.1 Synthetic Tasks

We perform hyper-parameter optimization using the Python package `ray.tune` [Liaw et al., 2018] on:

- **activation**, chosen among `leaky_relu`, `relu`, `selu` and `silu` as implemented in the Python library `PyTorch` [Paszke et al., 2019]. We find `selu` to achieve marginally better performance on toy datasets.
- **g**, the value of the diffusivity constant, chosen among $\{1, 2, 5, 10\}$. We find $g = 1$ to yield optimal results.

3.3.2 Cell Differentiation

We perform hyper-parameter optimization using the Python package `ray.tune` [Liaw et al., 2018] on:

- **activation**, chosen among `leaky_relu`, `relu`, `selu`, and `silu` as implemented in the Python library `PyTorch` [Paszke et al., 2019]. We observe that `silu` brings noticeable performance improvements on the cell differentiation dataset.
- **g**, the value of the diffusivity constant, chosen among $\{0.01, 0.1, 0.8, 1, 1.2, 2, 5\}$. We find $g = 1$ to yield optimal results.

3.3.3 Protein Conformational Changes

We use `AdamW` as our optimizer with a initial learning rate of 0.001, and training batch size of 2. For each protein pair, we sample 10 timepoints in every epoch, so the model sees realizations from different timepoints of the corresponding Brownian Bridge. This was done to improve the training speed. We use a regularization strength of 1.0 for m^ϕ for all t. Inference on the validation set using training is carried out using the exponential moving average of parameters, and the moving average is updated every optimization step with a decay rate of 0.9. The model training is set to a maximum of 1000 epochs but training is typically stopped after 200 epochs beyond which no improvements in the validation metrics are observed.

Our model has 0.54M parameters and is trained for 200 epochs. After every epoch, we simulate trajectories on the validation set using our model and compute the mean RMSD. The best model selected using this procedure is used for inference on the test set. The baseline EGNN model has 0.76M parameters and is trained for 1000 epochs.

3.3.4 Rigid Protein Docking

We use ADAM as our optimizer with a learning rate of 0.001, and training batch size of 2. For each ligand, we sample 5 timepoints during every training epoch so that the model is exposed to different timepoints from the corresponding Brownian Bridge for each ligand. This number was chosen as a tradeoff between CUDA memory and coverage of timepoints between 0 and 1. We use a regularization strength of 1.0 for m^ϕ for all t. Inference on the validation set using training is carried out using the exponential moving average of parameters, and the moving average is updated every optimization step with a decay rate of 0.999. The model training is set to a maximum of 1000 epochs but training is typically stopped after 100 epochs beyond which no improvements in the validation metrics are observed.

4 REPRODUCIBILITY

Code utilized in this publication can be found at `https://github.com/vsomnath/aligned_diffusion_bridges`, with a mirror at `https://github.com/IBM/aligned_diffusion_bridges`.

References

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Charlotte Bunne, Stefan G Stark, Gabriele Gut, Jacobo Sarabia del Castillo, Kjong-Van Lehmann, Lucas Pelkmans, Andreas Krause, and Gunnar Ratsch. Learning Single-Cell Perturbation Responses using Neural Optimal Transport. *bioRxiv*, 2021.

Marco Cuturi. Sinkhorn Distances: Lightspeed Computation of Optimal Transport. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 26, 2013.

Marco Cuturi, Laetitia Meng-Papaxanthos, Yingtao Tian, Charlotte Bunne, Geoff Davis, and Olivier Teboul. Optimal Transport Tools (OTT): A JAX Toolbox for all things Wasserstein. *arXiv Preprint arXiv:2201.12324*, 2022.

Sjoerd J de Vries, Christina EM Schindler, Isaure Chauvot de Beauchêne, and Martin Zacharias. A Web Interface for Easy Flexible Protein-Protein Docking with ATTRACT. *Biophysical Journal*, 108(3), 2015.

Israel T Desta, Kathryn A Porter, Bing Xia, Dima Kozakov, and Sandor Vajda. Performance and Its Limits in Rigid Body Protein-Protein Docking. *Structure*, 28(9), 2020.

Octavian-Eugen Ganea, Xinyuan Huang, Charlotte Bunne, Yatao Bian, Regina Barzilay, Tommi S. Jaakkola, and Andreas Krause. Independent SE(3)-Equivariant Models for End-to-End Rigid Protein Docking. In *International Conference on Learning Representations (ICLR)*, 2022.

Mario Geiger and Tess Smidt. e3nn: Euclidean neural networks. *arXiv preprint arXiv:2207.09453*, 2022.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A Kernel Two-Sample Test. *Journal of Machine Learning Research*, 13, 2012.

Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5), 1976.

Dima Kozakov, David R Hall, Bing Xia, Kathryn A Porter, Dzmitry Padhorny, Christine Yueh, Dmitri Beglov, and Sandor Vajda. The ClusPro web server for protein–protein docking. *Nature Protocols*, 12(2), 2017.

Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. Tune: A Research Platform for Distributed Model Selection and Training. *arXiv preprint arXiv:1807.05118*, 2018.

Efrat Mashiach, Dina Schneidman-Duhovny, Aviyah Peri, Yoli Shavit, Ruth Nussinov, and Haim J Wolfson. An integrated suite of fast docking algorithms. *Proteins: Structure, Function, and Bioinformatics*, 78(15), 2010.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Christina EM Schindler, Isaure Chauvot de Beauchêne, Sjoerd J de Vries, and Martin Zacharias. Protein-protein and peptide-protein docking and refinement using ATTRACT in CAPRI. *Proteins: Structure, Function, and Bioinformatics*, 85(3):391–398, 2017.

Dina Schneidman-Duhovny, Yuval Inbar, Ruth Nussinov, and Haim J Wolfson. PatchDock and SymmDock: servers for rigid and symmetric docking. *Nucleic Acids Research*, 33, 2005.

Vasileios Stathias, Anna M Jermakowicz, Marie E Maloof, Michele Forlin, Winston Walters, Robert K Suter, Michael A Durante, Sion L Williams, J William Harbour, Claude-Henry Volmar, et al. Drug and disease signature integration identifies synergistic combinations in glioblastoma. *Nature Communications*, 9(1), 2018.

Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

Thom Vreven, Iain H Moal, Anna Vangone, Brian G Pierce, Panagiotis L Kastritis, Mieczyslaw Torchala, Raphael Chaleil, Brian Jiménez-García, Paul A Bates, Juan Fernandez-Recio, et al. Updates to the integrated protein–protein interaction benchmarks: docking benchmark version 5 and affinity benchmark version 2. *Journal of Molecular Biology*, 427(19), 2015.

Caleb Weinreb, Alejo Rodriguez-Fraticelli, Fernando D Camargo, and Allon M Klein. Lineage tracing on transcriptional landscapes links state to fate during differentiation. *Science*, 367, 2020.

F Alexander Wolf, Philipp Angerer, and Fabian J Theis. SCANPY: large-scale single-cell gene expression data analysis. *Genome Biology*, 19(1), 2018.

Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. Geodiff: A geometric diffusion model for molecular conformation generation. In *International Conference on Learning Representations*, 2022.

Yumeng Yan, Huanyu Tao, Jiahua He, and Sheng-You Huang. The HDOCK server for integrated protein–protein docking. *Nature Protocols*, 15(5), 2020.