Illuminating protein space with a programmable generative model

John Ingraham, Max Baranov, Zak Costello, Vincent Frappier, Ahmed Ismail, Shan Tie, Wujie Wang, Vincent Xue, Fritz Obermeyer, Andrew Beam, Gevorg Grigoryan

Generate Biomedicines

December 1, 2022

Abstract

Three billion years of evolution have produced a tremendous diversity of protein molecules, and yet the full potential of this molecular class is likely far greater. Accessing this potential has been challenging for computation and experiments because the space of possible protein molecules is much larger than the space of those likely to host function. Here we introduce Chroma, a generative model for proteins and protein complexes that can directly sample novel protein structures and sequences and that can be conditioned to steer the generative process towards desired properties and functions. To enable this, we introduce a diffusion process that respects the conformational statistics of polymer ensembles, an efficient neural architecture for molecular systems based on random graph neural networks that enables long-range reasoning with sub-quadratic scaling, equivariant layers for efficiently synthesizing 3D structures of proteins from predicted inter-residue geometries, and a general low-temperature sampling algorithm for diffusion models. We suggest that Chroma can effectively realize protein design as Bayesian inference under external constraints, which can involve symmetries, substructure, shape, semantics, and even natural language prompts. With this unified approach, we hope to accelerate the prospect of programming protein matter for human health, materials science, and synthetic biology.

Introduction

Protein molecules carry out most of the biological functions necessary for life, but inventing them is a complicated task that has taken evolution millions to billions of years. The field of computational protein design aims to shortcut this by automating the design of proteins for desired functions in a manner that is *programmable*. While there has been significant progress towards this goal over the past three decades [Kuhlman and Bradley, 2019, Huang et al., 2016], including the design of novel topologies, assemblies, binders, catalysts, and materials [Koga et al., 2012, Cao et al., 2022, Kries et al., 2013, Joh et al., 2014], most *de novo* designs have yet to approach the complexity and variety of macromolecules that are found in nature, which possess complex and asymmetric layered architectures built from many distinct sub-domains. Reasons for this include that (i) modeling the relationship between sequence, structure, and function is difficult, and (ii) most computational design methods rely on iterative search and sampling processes which, just like evolution, must navigate a rugged fitness landscape incrementally Maynard Smith [1970]. While many computational techniques have been developed to accelerate this search [Huang et al., 2016] and to improve the prediction of natural protein structures [Jumper et al., 2021], the space of *possible* proteins remains combinatorially large and only partially accessible by traditional computational methods. Determining how to efficiently explore the space of designable protein structures while also biasing towards specific functions remains an open challenge.

An alternative and potentially appealing approach to protein design would be to directly sample from the space of proteins that are compatible with a set of desired functions. While this could address the fundamental limitation of iterative search methods, it would require an effective parameterization of a-priori "plausible" protein space, a way to draw samples from this space, and a way to bias this sampling towards desired properties and functions. Deep generative models have proven successful in solving these kinds of high-dimensional modeling and inference problems in other domains, for example, in the text-conditioned generation of photorealistic images [Ramesh et al., 2021, 2022, Saharia et al., 2022]. For this reason, there has been considerable work developing generative models of protein space, applied to both protein sequences [Riesselman et al., 2018, Greener et al., 2018, Ingraham et al., 2019, Anand et al., 2022, Madani et al., 2020, Rives et al., 2021, Notin et al., 2022] and structures [Anand and Huang, 2018, Lin et al., 2021, Eguchi et al., 2022, Anand and Achim, 2022, Trippe et al., 2022, Wu et al., 2022a].

Despite these recent advances in generative models for proteins, we argue that there are three desiderata that have yet to be realized simultaneously in one system. These are (i) to jointly model the 3D structures and sequences of *full protein complexes*, (ii) to do so with computation that scales *sub-quadratically* with system size, and (iii) to enable *conditional sampling* under diverse cues without re-training. The first, generating full complexes, is important because protein function is often interpretable only in the context of a bound complex. The second, sub-quadratic scaling of computation, is important because it has been an essential ingredient for managing complexity in other modeling disciplines, such as in computer vision, where convolutional neural networks scale linearly with the number of pixels in an image, and in computational physics, where fast N-body methods are used for efficient simulation of everything from stellar to molecular systems Barnes and Hut [1986]. And lastly, the requirement to sample *conditionally* from a model without having to retrain it on new target functions is of significant interest because protein design projects often involve many complex and composite requirements which may vary over time.

Here we introduce Chroma, a generative model for proteins that achieves all three of these requirements by modeling full complexes with quasi-linear computational scaling and by admitting arbitrary conditional sampling at generation time. It builds on the framework of diffusion models Sohl-Dickstein et al. [2015], Song et al. [2021], which model high-dimensional distributions by gradually transforming them into simple distributions and learning to reverse this process, and of graph neural networks Gilmer et al. [2017], Battaglia et al. [2018], which can efficiently reason over complex molecular systems. We show that it produces high-quality, diverse, novel, and designable structures, and that it enables *programmable* generation of proteins conditioned on several

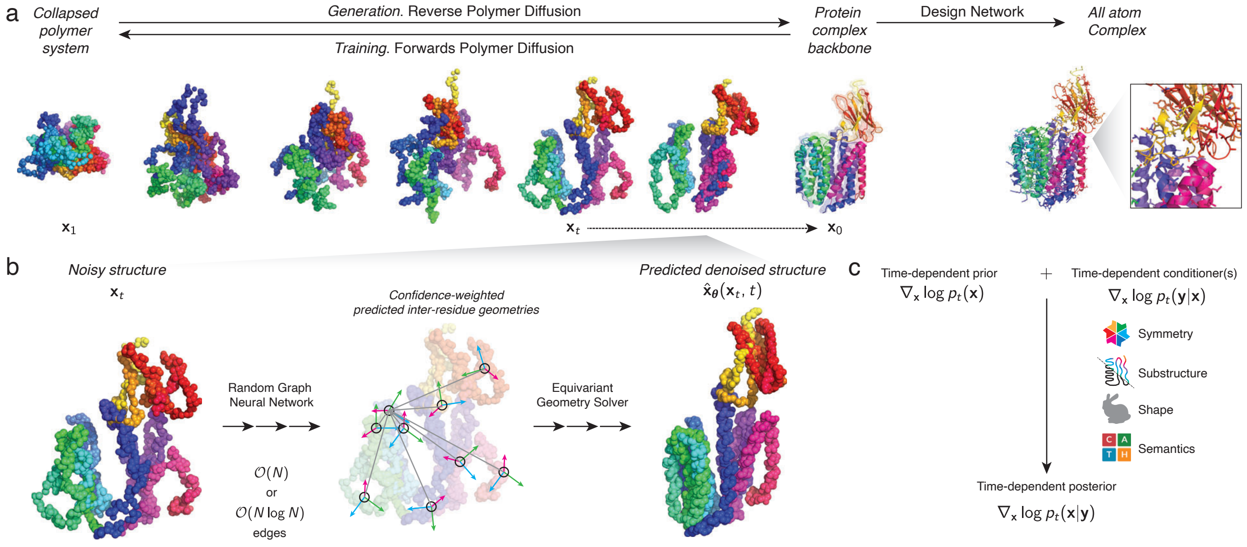

a
Collapsed polymer system
Generation. Reverse Polymer Diffusion
Training. Forwards Polymer Diffusion
Protein complex backbone
Design Network
All atom Complex
$\mathbf{x}_1$
$\mathbf{x}_t$
$\mathbf{x}_0$
b
Noisy structure
$\mathbf{x}_t$
Random Graph Neural Network
$\mathcal{O}(N)$
or
$\mathcal{O}(N \log N)$
edges
Confidence-weighted predicted inter-residue geometries
Equivariant Geometry Solver
Predicted denoised structure
$\hat{\mathbf{x}}_{\theta}(\mathbf{x}_t, t)$
c
Time-dependent prior
$\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$
+
Time-dependent conditioner(s)
$\nabla_{\mathbf{x}} \log p_t(\mathbf{y}|\mathbf{x})$
Symmetry
Substructure
Shape
C A T H
Semantics
Time-dependent posterior
$\nabla_{\mathbf{x}} \log p_t(\mathbf{x}|\mathbf{y})$

Figure 1: ***Chroma* is a generative model for proteins and protein complexes that combines a structured diffusion model for protein backbones with scalable molecular neural networks for backbone synthesis and all-atom design. a**, A correlated diffusion process with chain and radius of gyration constraints gradually transforms protein structures into random collapsed polymers (right to left). The reverse process (left to right) can be expressed in terms of a time-dependent optimal denoiser $\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ (**b**), which we parameterize in terms of a random graph neural network with long-range connectivity inspired by efficient N-body algorithms (**b:**, middle) and a fast method for solving for a global consensus structure given predicted inter-residue geometries (**b:**, right). (**a**, top right) Another graph-based *design network* generates protein sequences and side-chain conformations conditionally based on the sampled backbone. **c**, The time-dependent protein prior learned inside of the diffusion model can be combined with auxiliary conditioning information for programmable generation of protein systems.

different properties such as symmetry, shape, protein class, and even textual input. We anticipate that scalable generative models like Chroma will enable a widespread and rapid increase in our ability to design and build protein systems fit for function.

Results

A scalable generative model for protein systems

Chroma achieves high-fidelity and efficient generation of proteins by introducing a new diffusion process, neural network architecture, and sampling algorithm based on principles from contemporary generative modeling and biophysical knowledge. Diffusion models generate data by learning to reverse a noising process, which for previous image modeling applications has typically been uncorrelated Gaussian noise. In contrast, our model learns to reverse a *correlated* noise process to match the empirical covariance structure in real proteins which is dominated by the constraints of a collapsed polymer to be chain structured with a particular radius of gyration (Fig. 1a, Appendix C). Prior models for protein structure have typically leveraged computation that scales as $\mathcal{O}(N^2)$

[Trippe et al., 2022, Wu et al., 2022a] or $\mathcal{O}(N^3)$ [Jumper et al., 2021, Anand and Achim, 2022] in the number of residues N, which has limited their application to small systems or required large amounts of computation for modestly sized systems. To overcome this, Chroma introduces a novel neural network architecture (Fig. 1b) for processing and updating molecular coordinates that uses random long range graph connections with connectivity statistics inspired by fast N-body methods [Barnes and Hut, 1986] and that scales sub-quadratically ($\mathcal{O}(N)$ or $\mathcal{O}(N \log N)$, Appendix D). Finally, we also introduce methods for principled *low-temperature sampling* from diffusion models with a modified diffusion process that allows us to increase quality of sampled backbones (increasing likelihood) while reducing conformational diversity (reducing entropy). A *design network* then generates sequence and side-chain conformations conditioned on the sampled backbone, yielding a joint generative process for the full sequence and the 3D positions of all heavy atoms in a protein complex. The design network is based on the same graph neural network as the backbone network, but with conditional sequence decoding layers and side-chain decoding layers that are similar to prior works [Ingraham et al., 2019, Anand et al., 2022] and have recently seen further refinement and experimental validation [Jing et al., 2020, Hsu et al., 2022, Dauparas et al., 2022]. While it is also possible to model sequence and side chain degrees of freedom as part of a joint diffusion [Hoogeboom et al., 2021], we found that a sequential factorization is effective while being considerably more efficient.

An important aspect of our diffusion-based framework is that it enables *conditional* sampling under combinations of user-specified constraints. This is made possible by a key property of diffusion models: they can recast the diffusion process for a target conditional distribution $p(\mathbf{x}|\mathbf{y})$ of proteins $\mathbf{x}$ given constraints $\mathbf{y}$, as a combination of the learned gradient field from the diffusion model $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ and gradients from external classifiers that have been trained to predict labels $\mathbf{y}$ from noisy examples of $\mathbf{x}$, i.e. $\nabla_{\mathbf{x}} \log p_t(\mathbf{y}|\mathbf{x})$ [Song et al., 2021] (Fig. 1c, Appendix B). This means that any classifier that predicts a protein property (e.g. $p_t(\mathbf{y}|\mathbf{x})$) from structure can be repurposed to guide the diffusion process towards proteins with those properties. To demonstrate and explore the extent of this programmatic conditioning, we introduce a variety of analytic and learned conditioners $p_t(\mathbf{y}|\mathbf{x})$ (Fig. 1c, Appendix I). This includes geometrical constraints, which are typically analytic and include constraints on distance (Appendix J), substructure root mean-squared deviation (RMSD) from a target substructure (Appendix K), symmetric complexes under arbitrary symmetry groups (Appendix L), and shape matching to arbitrary point clouds (Appendix M). We also explore the possibilities of *semantic* prompting by training graph neural networks to predict multi-scale classifications (Appendix N) and natural language annotations (Appendix O) directly from protein structures. Any subset of all of these constraints may then be combined for bespoke, on-demand protein generation subject to problem-specific desiderata.

Analysis of unconditional samples

We first sought to characterize the diffusion model for protein backbone structures by analyzing a large set of unconditional samples of proteins and protein complexes. When initially exploring unconditional samples using the diffusion model, we observed an interesting phenomenon where the model assigned high likelihoods to natural structures but still produced samples that usually were mostly unstructured with little backbone hydrogen bonding and secondary structure content (Appendix B). We reasoned that this phenomenon was analogous to a common issue with

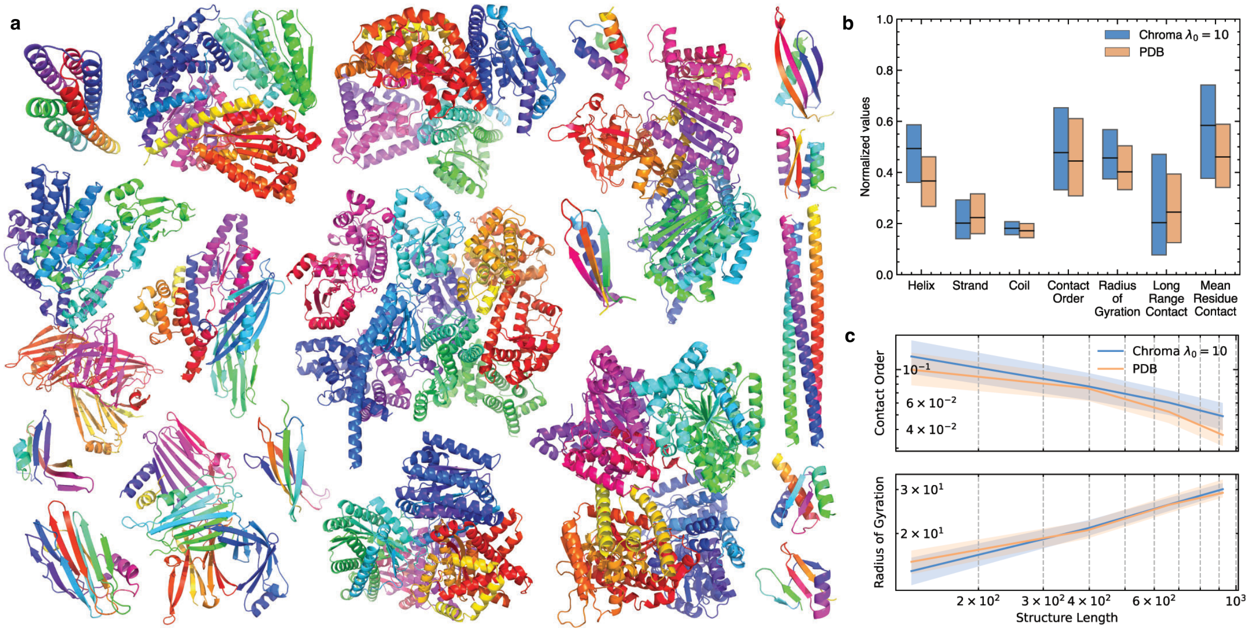


Figure 2: **Analysis of unconditional samples reveals diverse geometries that reproduce low-level structural statistics but exhibit novel higher-order structure.** **a**, A representative set of proteins and protein complexes sampled from the unconditioned backbone model at inverse temperature $\lambda_0 = 10$ exhibits complex and diverse architectures with high secondary structure content. **b**, Across a set a set of 10,000 single chains, samples from Chroma have structural properties that are similar to natural protein structures from the PDB, including secondary structure utilization and length-normalized contacBt order, radius of gyration, and contact density statistics. Low-temperature samples from Chroma tend to slightly favor helices over strands and are more compact than those found in the PDB. **c**, Chroma samples reproduce length-dependent scaling of contact order [Plaxco et al., 1998] and radius of gyration.

likelihood-based generative models such as language models [Holtzman et al., 2020] and diffusion models [Dhariwal and Nichol, 2021], where it is typical for models to take longer to eliminate the probability mass for poorly structured states (of which there are usually exponentially many more) than it does for models to assign high probabilities to well-structured states (e.g. well-formed sentences and images). The standard solution to this issue of *overdispersion* of likelihood-based generative models is to leverage modified sampling procedures which bias towards higher probability states, such as beam-search or greedy decoding in lanaguage models [Holtzman et al., 2020] or *classifier guidance* and *classifier-free guidance* for conditional diffusion models [Ho and Salimans, 2021]. The latter methods of classifier guidance heavily rely on strong classifiers to be effective, and we found them insufficient to improve sample quality of our backbones. Instead, we developed a novel and general sampling algorithm for diffusion models that enables sampling from the temperature-perturbed distribution $\frac{1}{Z}p(\mathbf{x})^{\lambda_0}$ (Appendix B). Increasing the inverse temperature parameter λ_0 redistributes probability mass towards higher likelihood states. We emphasize that these low-temperature sampling methods are of critical importance to our framework, and throughout the paper we sample backbones at inverse temperatures ranging from $\lambda_0 = 16$ to $\lambda_0 = 1$ (Appendix H).

Equipped with our low-temperature sampling method, we characterized a large number of samples from the prior and to compare their structural statistics to data from the Protein Data Bank (PDB). We sampled 50,000 single-chain backbones and 10,000 complex backbones at inverse temperature $\lambda_0 = 10$. As can be seen in Fig. 2a, the unconditional samples display many properties shared by natural proteins, such as complex layering of bundled alpha helices and beta sheets in cooperative unknotted folds. We provide grids of randomly picked subsets of these samples in Appendix H (Supplementary Figs. 6 and 7 for single-chain and complex structures, respectively). To quantitatively measure the agreement with natural folds, we sampled another set 10,000 single chain samples and computed several key structural properties, including secondary structure utilization, contact order [Plaxco et al., 1998], length-dependent radius of gyration [Tanner, 2016], length-dependent long-range contact frequency and density of inter-residue contacts (Appendix H). We generally observe agreement between the distribution of these statistics for Chroma and samples from the PDB, indicateing that these backbone structures appear to be similar to native proteins (Fig. 2b and c). We do see a slight over-utilization of α-helices (by $\sim$ 0.5-1.0 standard deviations), which we suspect may be a consequence of low-temperature sampling (i.e., helices are used more frequently than strands in natural proteins, but the ratio is somewhat accentuated in low-temperature samples).

Evaluation of generated protein structures

An important question is whether the backbone structures generated by Chroma can be realized with sequences of natural amino acids—i.e., whether they are "designable". While the only way to answer this question definitively is through experimental characterization, we performed two types of analyses to provide *in-silico* support for the designability of our generated structures. In the first, we created sequences for our generated backbones and assessed whether open-source structure prediction models representative of the current state of art [Wu et al., 2022b] would predict that the sequences would correctly fold into the original, generated structure (Fig. 3a). We note that this type of sequence-structure mutual consistency test rests on the ability of the structure prediction model to generalize to novel folds and topologies, which has yet to be conclusively demonstrated. Nevertheless, this evaluation is able to provide partial supporting evidence for designability in the instances where the predicted and generated structures have strong agreement, and there have been some successful applications of structure prediction for de novo design [Anishchenko et al., 2021]. Figs. 3b-d show the results of this analysis for a set of 100 backbones generated at random using Chroma with length uniformly randomly sampled in the range $[100, 500]$ (Appendix H.4). Fig. 3b shows the TM-score [Zhang and Skolnick, 2005] achieved between generated and predicted structures in this test as a function of protein length. While it is not surprising that this task is more challenging for longer proteins, as the difficulty of both generation and prediction will generally increase with chain length, it is remarkable that TM > 0.5 (a broadly utilized cutoff to indicate "the same fold") is achieved even for proteins as long as 480 amino acids. The overall distribution of TM-scores in Fig. 3c shows the cutoff of 0.5 is achieved in 55% of cases overall.

To isolate the possible confounding effect of structure prediction, we also directly analyzed the higher-order, local backbone geometries in Chroma-generated backbones versus natural backbones. Natural protein structures exhibit considerable degeneracy in their use of local tertiary backbone geometries, such that completely unrelated proteins tend to utilize very similar struc-

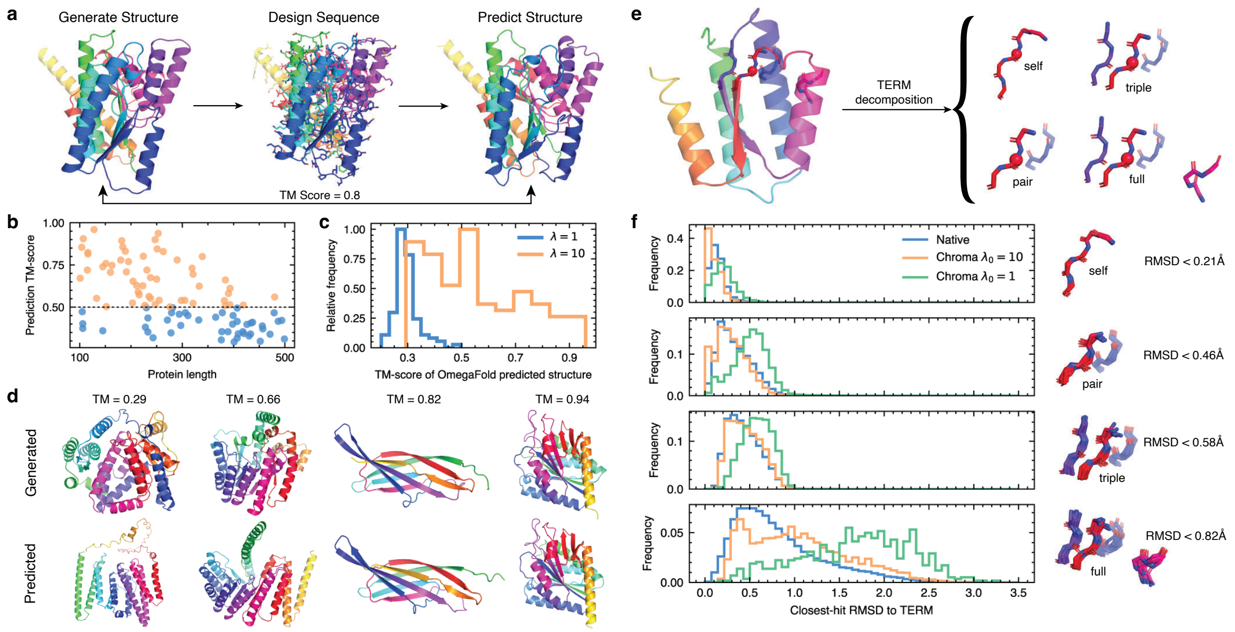


Figure 3: ***Chroma*-generated backbones are designable by a variety of computational metrics.** **a**, Workflow of the test involving generation of a backbone using Chroma, design of sequences for this backbone using our sequence design model, structure prediction for these sequences with OmegaFold [Wu et al., 2022b], and comparison of predicted versus generated structures. **b:** Resulting TM-scores (best out of 100 design attempts for each structure) vs. to protein length. **c**, Distribution of resulting TM-scores (orange). Shown in blue is the same distribution from a control calculation involving the same workflow but with Chroma sampling at inverse temperature $\lambda_0 = 1$ (versus $\lambda_0 = 10$ in the original test). **d**, Comparisons between generated and predicted structures for several cases spanning a range of TM-scores from the worst (left) to the best (right) observed in this test. **e**, A visual depiction of a TERMs decomposition. **f**, The distribution of closest-match RMSD for TERMs of increasing order originating from native or Chroma-generated backbones (with inverse temperature λ_0 being 1 or 10).

tural motifs and relatively few local tertiary geometries account for the majority of the observed structures [Mackenzie et al., 2016]. These tertiary motifs, or *TERMs*, consist of a central residue, its backbone-contiguous neighbors, neighboring residues capable of contacting the central residue, and their backbone-contiguous neighbors [Mackenzie et al., 2016, Zheng et al., 2015]. Depending on how many contacting residues are combined into the motif, TERMs can be distinguished as self, pair, triple, or higher-order, corresponding to having zero, one, two, or more contacting neighbors (Fig. 3e). To compare the local geometry of Chroma-generated backbones with that of native structures, we isolated all possible self, pair, triple, and full TERMs (i.e., TERMs containing all contacting residues for a given central residue) and identified the closest neighbor (by backbone RMSD) to each within a redundancy-pruned subset of the PDB, the search database (Appendix H.5). We performed a similar analysis on a set of native proteins not contained within the search database, additionally taking care to remove any homologous matches (Appendix H.5). Figure 3f shows the distribution of closest-neighbor RMSDs for TERMs derived from both natural (native) and Chroma-sampled backbones that were generated at inverse temperatures $\lambda_0 = 10$ and

$\lambda_0 = 1$. The distributions of nearest neighbor RMSD were very close for low-temperature samples from Chroma and native proteins, indicating that Chroma geometries are valid and likely to be as designable as native proteins, including complex motifs with four or five disjoint fragments (Fig. 3f, bottom panel). Because native amino-acid choices are driven by these local geometries [Zhou et al., 2019], and adherence to TERM statistics has been previously shown to correlate with structural model accuracy and success in *de-novo* design Zheng et al. [2015], Zhou et al. [2019], this argues for the general designability of Chroma-generated backbones in a model-independent manner. Notably, the samples from Chroma without temperature adjustment (i.e. $\lambda_0 = 1$) exhibit unfolded and unstructured geometries (Fig. 3f), which underlines the importance of having a method for low-temperature sampling.

Programmability

Being able to efficiently sample realistic proteins is necessary but not sufficient for downstream applications such as therapeutic development, since unconditional samples are unlikely to possess desired functional properties. An important aspect of Chroma is its *programmability*, which means that it is straightforward to directly bias sampling towards desired properties (Section "A scalable generative model for protein systems" and Fig. 1). This leverages a remarkable property of diffusion models, which is that they can make traditionally difficult Bayesian inversion problems tractable [Sohl-Dickstein et al., 2015, Song et al., 2021]. Specifically, to condition on a property or set of properties (collection of events) $\mathbf{y}$, it is sufficient to train a time-dependent classifier model $p_t\left(\mathbf{y}|\mathbf{x}\right)$ on the noised structures $\mathbf{x}_t \sim p_t(\mathbf{x}|\mathbf{x}_0)$ and to adjust the sampling process by $\nabla_x \log p_t\left(\mathbf{y}|\mathbf{x}\right)$, the gradient of the log likelihood that structure $\mathbf{x}_0$ will have the desired property at time $t = 0$ (the end of the reverse diffusion). Depending on the property in question, this probability can be expressed analytically in a closed form (e.g., see Appendix J), as an empirical, analytic approximation (e.g., see Appendix K), or as the prediction of a neural network (e.g., see Appendix N and O). This conditioning formalism is very natural for protein design as it decouples the problem of parameterizing the space of likely structured proteins, the cost of which is amortized by training a strong prior model once, from the problem of expressing the correct determinants of the desired function, which can be isolated to the classifier model. Thus, by focusing effort on building the right classifiers, designers can spend most of their time on solving specific functional objectives.

To demonstrate what may be achievable with conditional generation, we built several $p_t\left(\mathbf{y}|\mathbf{x}\right)$ classifier models (Appendix I), including those where $\mathbf{y}$ encoded: (i) a distance-based constraint (e.g., a "contact" between residues), (ii) the presence of a disjoint sub-structure (based on backbone RMSD), either anywhere in the generated structure or in a pre-specified alignment, (iii) various residue-, domain-, and complex-level classes (e.g., CATH or PFAM domains, secondary structure labels, interfacial residues), and (iv) natural language annotations trained on protein captions from the PDB and Uniprot [Consortium, 2020]. While we believe that each of these classifiers represents only a preliminary realization of these conditioning modes, we already see that they suggest tremendous possibilities for *programmable* design.

We begin by considering analytic conditioners that can control protein backbone geometry. We found that conditioning on the symmetry of protein complexes, which can also be cast as sampling under a symmetry constraint, can very flexibly generate samples under arbitrary symmetry

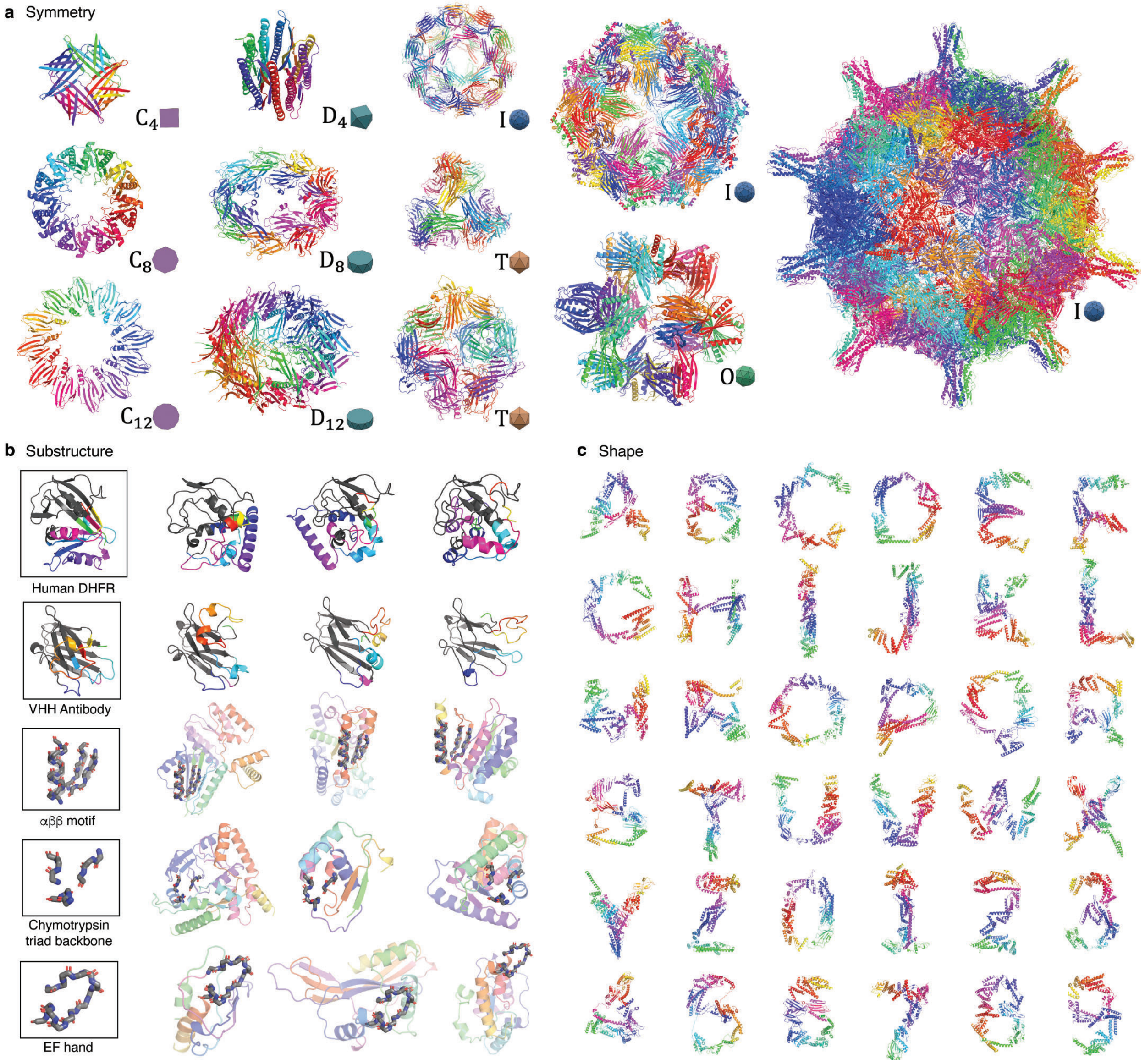


Figure 4: **Symmetry, substructure, and shape conditioning enable geometric molecular programming.** **a**, Conditioning on arbitrary symmetry groups is possible by symmetrizing gradient, noise, and initialization through the sampling process (Appendix L). We show how cyclic C_n, dihedral D_n, tetrahedral T, octahedral O, and icosahedral I symmetries can produce a wide variety of possible homomeric complexes. The righmost protein complex contains 60 subunits and 96,000 total residues. **b**, Conditioning on partial substructure (monochrome) enables protein "infilling" or "outfilling". Top two rows illustrate regeneration (color) of half of a protein (enzyme DHFR, first row) or CDR loops of an antibody (second row); Appendix K. Next three rows show conditioning on a pre-defined motif; order and matching location of motif segments is not pre-specified here. **c**, Lastly, it is possible to condition on arbitrary volumetric shapes by using gradients derived from Optimal Transport (Appendix L). We test the ability of Chroma to solve for backbone configurations subject to the complex geometries of the Latin alphabet and numerals.

groups (Fig. 4a, Appendix L). Figure 4a illustrates symmetry-conditioned generation across many groups, from simple 4-subunit cyclic symmetries up to a capsid-sized icosahedral complex with 96,000 total residues and over 380,000 atoms. This also demonstrates why favorable computational scaling properties, such as quasilinear computation time (Appendix D), are important, as efficient computation facilitates scaling to larger systems. Symmetric assemblies are common in nature and there have been some successes with *de novo* symmetric designs [Wicky et al., 2022, King et al., 2014], but it has been generally challenging to simultaneously optimize for both the molecular interaction details between protomers and the desired overall symmetry in design. By enabling simple symmetry conditioning within the generation process, joint Chroma should make it simpler to more easily sample structures that simultaneously meet both requirements.

Next, we explore substructure conditioning in Fig. 4b, which is a central problem for protein design as it can facilitate preserving one part of a protein's structure (and function) while modifying another part of the structure (and potentially function). In the top row, we "cut" the structure of human dihydrofolate reductase (PDB code 1DRF) into two halves with a plane, remove one of the halves, and regenerate the other half anew. The cut plane introduces several discontinuities in the chain simultaneously, and the generative process needs to sample a solution that satisfies these boundary conditions while being realistic. Nevertheless, the samples achieve both goals and, interestingly, do so in a manner very different from both each other and from natural DHFR. In the second row of Fig. 4b, we cut out the complementarity-determining regions of a VHH antibody and rebuild them conditioned on the remaining framework structure. The generated structures are once again plausible despite similar difficulties to the DHFR example. Lastly, in the bottom three rows of Fig. 4b, condition on sub-structure in a *unregistered* manner, meaning that the exact alignment of the substructure (motif) within the chain is not specified *a priori* as it was in the prior examples. We "outfill" protein structure around several structural and functional motifs, including an $\alpha\beta\beta$ packing motif, backbone fragments encoding the catalytic triad active site of chymotrypsin, and the EF-hand Ca-binding motif. Again, these motifs are accommodated in a realistic manner using diverse and structured solutions.

In Fig. 4c we provide an early demonstration of a more exotic kind of conditioning in which we attempt to solve for backbone configurations subject to arbitrary volumetric shape specifications. We accomplish this by adding heuristic classifier gradients based on optimal transport distances [Peyré et al., 2019] between atoms in the structures and user-provided point clouds (Appendix M). As a stress test of this capability, we conditioned the generation of 1,000-residue single protein chains on the shapes of the Latin alphabet and Arabic numerals. While it remains unclear if any of these backbones would be sufficiently realistic to autonomously fold into their intended shapes, we see the model routinely implementing several core phenomena of protein backbones such as high secondary structure content, close packing with room for designed sidechains, and volume-spanning alpha-helical bundle and beta sheet elements. Although these shapes represent purely a challenging set of test geometries, more generally, shape is intimately related to functions in biology, for example, with membrane transporters, receptors, and structured assemblies that organize molecular events in space. Being able to control shape would be a useful subroutine for generalized protein engineering and design.

Finally, we demonstrate in Fig. 5 that it is possible to condition on protein *semantics* such as secondary structure, fold class (Fig. 5a) and natural language (Fig. 5b). Unlike for the geometric

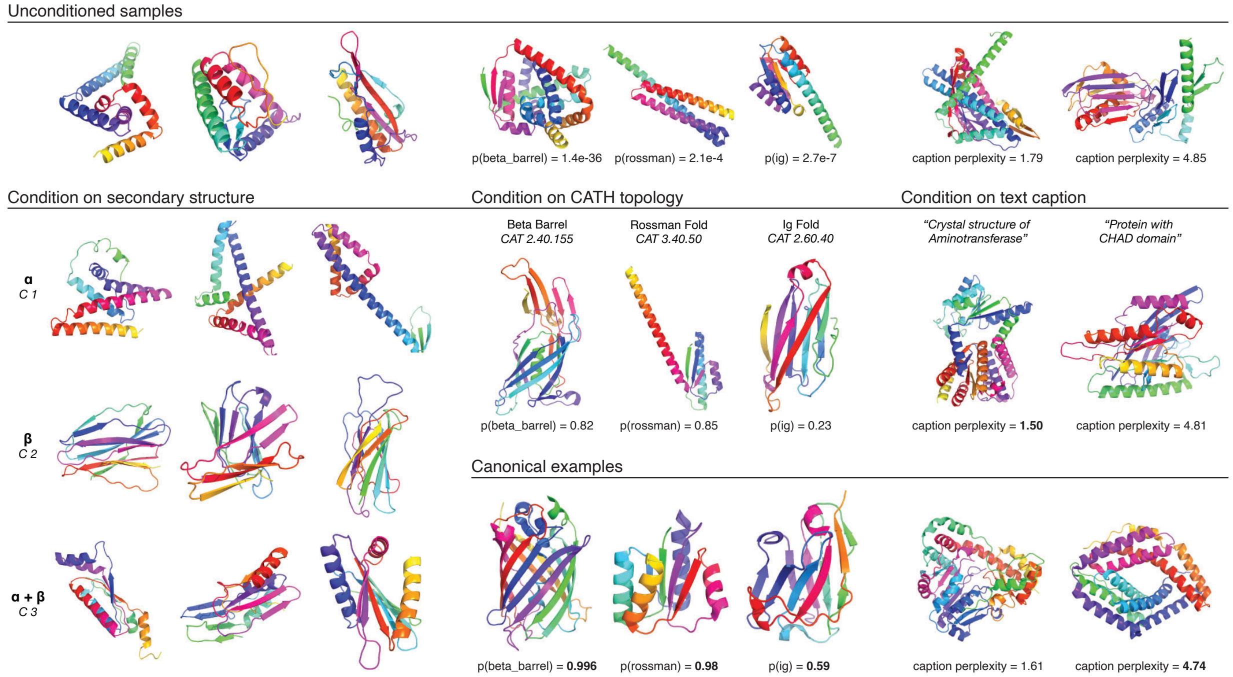

Unconditioned samples
p(beta_barrel) = 1.4e-36
p(rossman) = 2.1e-4
p(ig) = 2.7e-7
caption perplexity = 1.79
caption perplexity = 4.85
Condition on secondary structure
Condition on CATH topology
Condition on text caption
α
C 1
β
C 2
α + β
C 3
Beta Barrel
CAT 2.40.155
Rossman Fold
CAT 3.40.50
Ig Fold
CAT 2.60.40
"Crystal structure of Aminotransferase"
"Protein with CHAD domain"
p(beta_barrel) = 0.82
p(rossman) = 0.85
p(ig) = 0.23
caption perplexity = 1.50
caption perplexity = 4.81
Canonical examples
p(beta_barrel) = 0.996
p(rossman) = 0.98
p(ig) = 0.59
caption perplexity = 1.61
caption perplexity = 4.74

Figure 5: **Protein structure classifiers and caption models can bias the sampling process towards user-specified properties.** The top row shows example structures drawn unconditionally from the $p(\text{structure})$ model. Below, models trained to predict protein semantics are used to conditionally sample structures with desired secondary structures, belonging to particular topologies, or corresponding to natural language captions. In each column, all conditional samples are drawn starting from the same random seed as the unconditional sample shown at the top of the column. The samples based on secondary structure conditioning show the impact of classifiers trained to predict mainly alpha, mainly beta, and mixed alpha-beta structures. In the columns with topology-conditioned samples, the classifier's predicted probabilities for the intended topology are indicated. Similarly, in the columns with samples based on text conditioning, the caption model's average perplexities are shown. For the topology and text caption columns, PDB structures are shown ("Canonical examples") that exemplify the target condition.

conditioning where the classifier is correct by construction (e.g., the presence of a motif under a certain RMSD is unambiguous), here the classifiers are neural networks trained on structure data (and structure data are considerably more sparse than image data), so there can be a discrepancy between the label assigned by the classifier and the ground truth class. Thus, looking at the fold-conditioned generation (Fig. 5a), we see that conditional samples always improve classifier probabilities over unconditioned samples taken from the same random seed, but the classification is not always perfect. For example, for the cases of "beta barrel" and "Ig fold" classes, the generated samples look like believable representatives of the respective class. On the other hand, in the "Rossman fold" example, the structure has some of the features characteristic of the class (i.e., two helices packed against a sheet on one side), but does not contain all such features (e.g., the opposing side of the sheet is not fully packed with helices like in a classical Rossman fold). In Fig. 5b we demonstrate semantic conditioning on natural language captions, which similarly improves probabilities while not generically being valid. It is exciting to imagine the potential

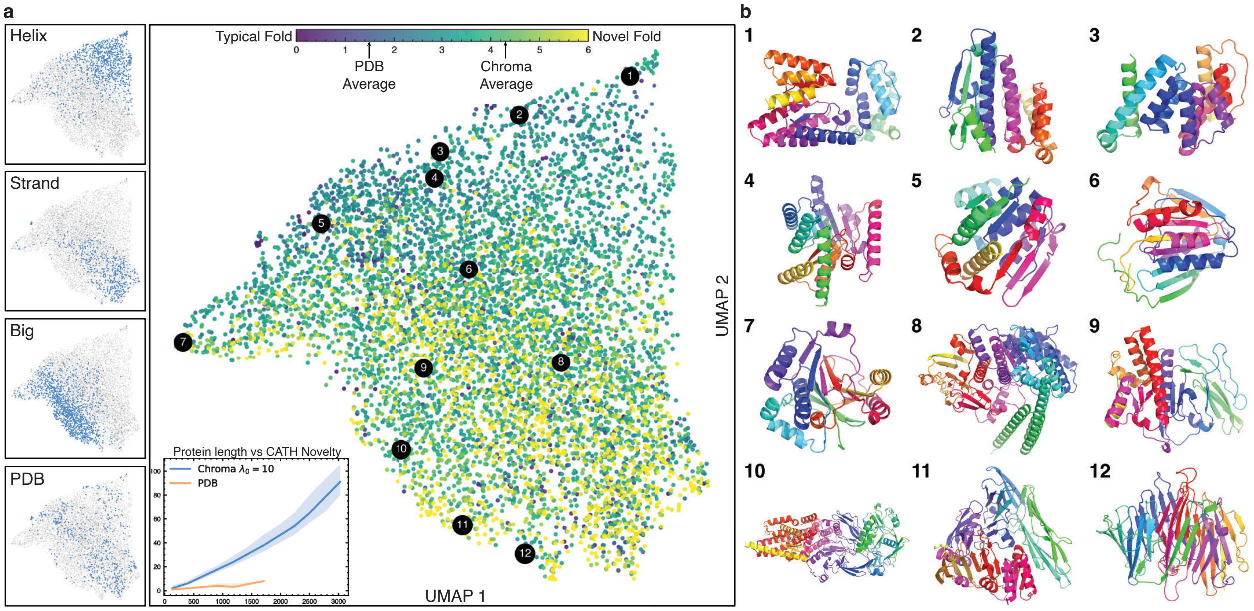


Figure 6: **Chroma-generated structures span natural protein space while also frequently demonstrating high novelty. a**, Proteins from the PDB and Chroma are featurized with 31 global fold descriptors derived from knot theory [Røgen and Fain, 2003, Harder et al., 2012] and are embedded into two dimensions using UMAP. The large figure is colored by the CATH coverage novelty measure normalized by protein length. Structural novelty was assessed by counting the number of CATH domains needed to achieve a greedy cover at least 80% of residues with TM > 0.5. On average Chroma needs 4.3 CATH domains per 200 amino acids to cover 80% of its residues while structures from the PDB need only 1.6. (**a**, inset) Chroma generated structures are more diverse and novel compared to structures from the PDB (regardless of protein length). The line represents the median value and is bounded by first (25%) and third (75%) quartile bands. The 4 smaller UMAP plots demonstrate the structure of the embedding by highlighting populations of structures that are mainly helices, strands, large (more than 500 residues), or natural proteins. The panel labeled *PBD* shows the distribution of natural proteins used to train the model. **b**, We render twelve proteins from across the embedding space with high novelty score (numbered in the embedding plot). The highlighted structures all have a novelty score of at least one standard deviation greater than the PDB.

of such a capability—i.e., being able to request desired protein features and properties directly via natural language prompts. Generative models such as Chroma can reduce the challenge of function-conditioned generation to the problem of building accurate classifiers for functions given structures. While there is clearly much more work to be done to make this useful in practice, high-throughput experiments and evolutionary data can likely make this possible in the near term.

Expanding protein space

So far, we have shown how samples from Chroma are realistic, designable, and can be conditioned, but it remains unclear how much they simply reproduce structures from the training set or involve novel fold topologies. We sought to characterize the novelty of samples from Chroma by taking our large set of single-chain samples (Section “Analysis of unconditional samples”) and mapping their structural homology to proteins in the PDB. Classifying protein structures into distinct topologies is not simple, but initiatives such as the CATH database [Sillitoe et al., 2021] have identified a relatively compact set of structural domains, which can efficiently explain most of the structures in the PDB as sub-domain combinations. Therefore, we can invoke a notion of “Novelty as CATH-compressibility” where we ask how easy it is to reduce Chroma-sampled structures to the compositions of CATH domains. We define a novelty score as the number of CATH domains required to achieve a greedy cover of 80% of the residues in a protein at a TM score above 0.5. Note that most valid proteins will be covered by at least *some* finite number of CATH domains as we include even very small domains in the coverage test (including those comprising a single secondary-structural element). The results of this analysis are shown in the inset of Fig. 6a, where the novelty score is plotted as a function of protein length for both Chroma-generated and native proteins. The gap between the two is apparent. Most native backbones are described by a handful of CATH domains, with this number rising slowly as a function of length. On the other hand, Chroma-generated structures routinely require tens of CATH domains to reach 80% coverage, and this number rises sharply with length.

We further find that samples from Chroma are diverse and covering of all of natural protein space. In Fig. 6a, we jointly represent samples from Chroma and a set of structures from the PDB with global topology descriptors derived from knot theory [Røgen and Fain, 2003, Harder et al., 2012], and embed these into two dimensions with UMAP [McInnes et al., 2018]. The resulting embedding appears to be semantically meaningful as sub-sets of structures belonging to different categories by size and secondary structures appear to cluster in this projection (sub-panels on the left in Fig. 6a). Coloring individual points in the embedding by the degree of novelty (i.e., the length-normalized number of CATH domains required to achieve 80% coverage), we can see that novelty in the space being sampled is spread broadly and not biased to only certain types of structures (Fig. 6a, right). This is especially clear when looking at a representative selection of novel samples shown in Fig. 6b (novelty threshold for selecting being > 2.2 CATH domains needed for 80% per 200 amino acids of length). Taken together, these results show that Chroma samples are diverse and have not yet appeared to exhibit any obvious biases.

Discussion

In this work, we present Chroma, a new generative model capable of generating novel and diverse proteins across a broad array of structures and properties. Chroma is *programmable* in the sense that it can be conditioned to sample proteins with a wide-array of user-specified properties, including: inter-residue distances and contacts, domains, sub-structures, and semantic specifications from classifiers. Chroma is able to generate proteins that have arbitrary and complex shapes, and we have shown the beginning of the ability to accept descriptions of desired properties as free text. Due to an efficient design with a new diffusion process, quasilinear scaling neural architecture, and low-temperature sampling method, Chroma is able to generate extremely large proteins and protein complexes (e.g. with ≥ 3000 residues) on a commodity GPU (e.g., an NVIDIA V100) in a few minutes.

These results are particularly striking in light of the historical difficulty of sampling realistic protein structures. The task of exploring the structure space in a way that can produce physically reasonable and designable conformations has been a long-standing challenge in protein design. In a few protein systems, it has been possible to parameterize the backbone conformation space mathematically–most notably the α-helical coiled coil [Grigoryan and DeGrado, 2011] and a few other cases with high symmetry [Woolfson et al., 2015]—and in these cases design efforts have benefited tremendously creating possibilities not available in other systems [Beesley and Woolfson, 2019, Woolfson et al., 2015]. For all other structure types, however, a great amount of computational time is being spent on the search for reasonable backbones, often leaving the focus on actual functional specifications out of reach. Chroma has the potential to address this problem, enabling a shift from focusing on generating feasible structures towards a focus on the specific task at hand—i.e., what the protein is intended to do. By leveraging proteins sampled over the first 3+ billion years of evolution on Earth and finding new ways to assemble stable protein matter, generative models such as Chroma are well poised to drive another expansion of biomolecular diversity for human health and bioengineering.

Acknowledgements

We would like to thank William F. DeGrado and Generate employees Adam Root, Alan Leung, Alex Ramos, Brett Hannigan, Eugene Palovcak, Frank Poelwijk, James Lucas, James McFarland, Karl Barber, Kristen Hopson, Martin Jankowiak, Mike Nally, Molly Gibson, Ross Federman, Stephen DeCamp, Thomas Linsky, Yue Liu, and Zander Harteveld for reading of the manuscript draft and providing helpful comments.

References

Brian Kuhlman and Philip Bradley. Advances in protein structure prediction and design. *Nat. Rev. Mol. Cell Biol.*, 20(11):681–697, November 2019.

Po-Ssu Huang, Scott E Boyken, and David Baker. The coming of age of de novo protein design. *Nature*, 537(7620):320–327, September 2016.

Nobuyasu Koga, Rie Tatsumi-Koga, Gaohua Liu, Rong Xiao, Thomas B Acton, Gaetano T Montelione, and David Baker. Principles for designing ideal protein structures. *Nature*, 491(7423): 222–227, November 2012.

Longxing Cao, Brian Coventry, Inna Goreshnik, Buwei Huang, William Sheffler, Joon Sung Park, Kevin M Jude, Iva Marković, Rameshwar U Kadam, Koen H G Verschueren, Kenneth Verstraete, Scott Thomas Russell Walsh, Nathaniel Bennett, Ashish Phal, Aerin Yang, Lisa Kozodoy, Michelle DeWitt, Lora Picton, Lauren Miller, Eva-Maria Strauch, Nicholas D DeBouver, Allison Pires, Asim K Bera, Samer Halabiya, Bradley Hammerson, Wei Yang, Steffen Bernard, Lance Stewart, Ian A Wilson, Hannele Ruohola-Baker, Joseph Schlessinger, Sangwon Lee, Savvas N Savvides, K Christopher Garcia, and David Baker. Design of protein-binding proteins from the target structure alone. *Nature*, 605(7910):551–560, May 2022.

Hajo Kries, Rebecca Blomberg, and Donald Hilvert. De novo enzymes by computational design. *Curr. Opin. Chem. Biol.*, 17(2):221–228, April 2013.

Nathan H Joh, Tuo Wang, Manasi P Bhate, Rudresh Acharya, Yibing Wu, Michael Grabe, Mei Hong, Gevorg Grigoryan, and William F DeGrado. De novo design of a transmembrane zn2+-transporting four-helix bundle. *Science*, 346(6216):1520–1524, December 2014.

John Maynard Smith. Natural selection and the concept of a protein space. *Nature*, 225(5232): 563–564, 1970.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022.

Adam J Riesselman, John B Ingraham, and Debora S Marks. Deep generative models of genetic variation capture the effects of mutations. *Nature methods*, 15(10):816–822, 2018.

Joe G Greener, Lewis Moffat, and David T Jones. Design of metalloproteins and novel protein folds using variational autoencoders. *Scientific reports*, 8(1):1–12, 2018.

John Ingraham, Vikas Garg, Regina Barzilay, and Tommi Jaakkola. Generative models for graph-based protein design. *Advances in neural information processing systems*, 32, 2019.

Namrata Anand, Raphael Eguchi, Irimpan I Mathews, Carla P Perez, Alexander Derry, Russ B Altman, and Po-Ssu Huang. Protein sequence design with a learned potential. *Nature communications*, 13(1):1–11, 2022.

Ali Madani, Bryan McCann, Nikhil Naik, Nitish Shirish Keskar, Namrata Anand, Raphael R Eguchi, Po-Ssu Huang, and Richard Socher. Progen: Language modeling for protein generation. *arXiv preprint arXiv:2004.03497*, 2020.

Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C Lawrence Zitnick, Jerry Ma, et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15):e2016239118, 2021.

Pascal Notin, Mafalda Dias, Jonathan Frazer, Javier Marchena Hurtado, Aidan N Gomez, Debora Marks, and Yarin Gal. Tranception: protein fitness prediction with autoregressive transformers and inference-time retrieval. In *International Conference on Machine Learning*, pages 16990–17017. PMLR, 2022.

Namrata Anand and Possu Huang. Generative modeling for protein structures. *Advances in neural information processing systems*, 31, 2018.

Zeming Lin, Tom Sercu, Yann LeCun, and Alexander Rives. Deep generative models create new and diverse protein structures. In *Machine Learning for Structural Biology Workshop, NeurIPS*, 2021.

Raphael R Eguchi, Christian A Choe, and Po-Ssu Huang. Ig-vae: Generative modeling of protein structure by direct 3d coordinate generation. *PLoS computational biology*, 18(6):e1010271, 2022.

Namrata Anand and Tudor Achim. Protein structure and sequence generation with equivariant denoising diffusion probabilistic models. *arXiv preprint arXiv:2205.15019*, 2022.

Brian L Trippe, Jason Yim, Doug Tischer, Tamara Broderick, David Baker, Regina Barzilay, and Tommi Jaakkola. Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. *arXiv preprint arXiv:2206.04119*, 2022.

Kevin E Wu, Kevin K Yang, Rianne van den Berg, James Y Zou, Alex X Lu, and Ava P Amini. Protein structure generation via folding diffusion. *arXiv preprint arXiv:2209.15611*, 2022a.

Josh Barnes and Piet Hut. A hierarchical $\mathcal{O}(n \log n)$ force-calculation algorithm. *Nature*, 324 (6096):446–449, 1986.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinícius Flores Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Çaglar Gülçehre, H. Francis Song, Andrew J. Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey R. Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261, 2018. URL http://arxiv.org/abs/1806.01261.

Bowen Jing, Stephan Eismann, Patricia Suriana, Raphael JL Townshend, and Ron Dror. Learning from protein structure with geometric vector perceptrons. *arXiv preprint arXiv:2009.01411*, 2020.

Chloe Hsu, Robert Verkuil, Jason Liu, Zeming Lin, Brian Hie, Tom Sercu, Adam Lerer, and Alexander Rives. Learning inverse folding from millions of predicted structures. *bioRxiv*, 2022.

Justas Dauparas, Ivan Anishchenko, Nathaniel Bennett, Hua Bai, Robert J Ragotte, Lukas F Milles, Basile IM Wicky, Alexis Courbet, Rob J de Haas, Neville Bethel, et al. Robust deep learning–based protein sequence design using proteinmpnn. *Science*, 378(6615):49–56, 2022.

Emiel Hoogeboom, Alexey A Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. Autoregressive diffusion models. *arXiv preprint arXiv:2110.02037*, 2021.

Kevin W Plaxco, Kim T Simons, and David Baker. Contact order, transition state placement and the refolding rates of single domain proteins 1 1edited by p. e. wright. *Journal of Molecular Biology*, 277(4):985–994, April 1998. doi: 10.1006/jmbi.1998.1645. URL https://doi.org/10.1006/jmbi.1998.1645.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rygGQyrFvH.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI.

John J Tanner. Empirical power laws for the radii of gyration of protein oligomers. *Acta Crystallographica Section D: Structural Biology*, 72(10):1119–1129, 2016.

Ruidong Wu, Fan Ding, Rui Wang, Rui Shen, Xiwen Zhang, Shitong Luo, Chenpeng Su, Zuofan Wu, Qi Xie, Bonnie Berger, Jianzhu Ma, and Jian Peng. High-resolution de novo structure prediction from primary sequence. *bioRxiv*, 2022b. doi: 10.1101/2022.07.21.500999. URL `https://www.biorxiv.org/content/early/2022/07/22/2022.07.21.500999`.

Ivan Anishchenko, Samuel J Pellock, Tamuka M Chidyausiku, Theresa A Ramelot, Sergey Ovchinnikov, Jingzhou Hao, Khushboo Bafna, Christoffer Norn, Alex Kang, Asim K Bera, Frank DiMaio, Lauren Carter, Cameron M Chow, Gaetano T Montelione, and David Baker. De novo protein design by deep network hallucination. *Nature*, 600(7889):547–552, December 2021.

Yang Zhang and Jeffrey Skolnick. Tm-align: a protein structure alignment algorithm based on the tm-score. *Nucleic acids research*, 33(7):2302–2309, 2005.

Craig O. Mackenzie, Jianfu Zhou, and Gevorg Grigoryan. Tertiary alphabet for the observable protein structural universe. *Proceedings of the National Academy of Sciences*, 113(47), nov 2016. doi: 10.1073/pnas.1607178113.

Fan Zheng, Jian Zhang, and Gevorg Grigoryan. Tertiary structural propensities reveal fundamental sequence/structure relationships. *Structure*, 23(5):961–971, May 2015. doi: 10.1016/j.str.2015.03.015. URL `https://doi.org/10.1016/j.str.2015.03.015`.

Jianfu Zhou, Alexandra E. Panaitiu, and Gevorg Grigoryan. A general-purpose protein design framework based on mining sequence–structure relationships in known protein structures. *Proceedings of the National Academy of Sciences*, 117(2):1059–1068, 2019. doi: 10.1073/pnas.1908723117.

The UniProt Consortium. UniProt: the universal protein knowledgebase in 2021. *Nucleic Acids Research*, 49(D1):D480–D489, November 2020. doi: 10.1093/nar/gkaa1100. URL `https://doi.org/10.1093/nar/gkaa1100`.

B. I. M. Wicky, L. F. Milles, A. Courbet, R. J. Ragotte, J. Dauparas, E. Kinfu, S. Tipps, R. D. Kibler, M. Baek, F. DiMaio, X. Li, L. Carter, A. Kang, H. Nguyen, A. K. Bera, and D. Baker. Hallucinating symmetric protein assemblies. *Science*, 378(6615):56–61, October 2022. doi: 10.1126/science.add1964. URL `https://doi.org/10.1126/science.add1964`.

Neil P. King, Jacob B. Bale, William Sheffler, Dan E. McNamara, Shane Gonen, Tamir Gonen, Todd O. Yeates, and David Baker. Accurate design of co-assembling multi-component protein nanomaterials. *Nature*, 510(7503):103–108, May 2014. doi: 10.1038/nature13404. URL `https://doi.org/10.1038/nature13404`.

Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

Peter Røgen and Boris Fain. Automatic classification of protein structure by using gauss integrals. *Proceedings of the National Academy of Sciences*, 100(1):119–124, 2003.

Tim Harder, Mikael Borg, Wouter Boomsma, Peter Røgen, and Thomas Hamelryck. Fast large-scale clustering of protein structures using gauss integrals. *Bioinformatics*, 28(4):510–515, 2012.

Ian Sillitoe, Nicola Bordin, Natalie Dawson, Vaishali P Waman, Paul Ashford, Harry M Scholes, Camilla SM Pang, Laurel Woodridge, Clemens Rauer, Neeladri Sen, et al. Cath: increased structural coverage of functional space. *Nucleic acids research*, 49(D1):D266–D273, 2021.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861, 2018. doi: 10.21105/joss.00861. URL https://doi.org/10.21105/joss.00861.

Gevorg Grigoryan and William F. DeGrado. Probing designability via a generalized model of helical bundle geometry. *Journal of Molecular Biology*, 405(4):1079–1100, January 2011. doi: 10.1016/j.jmb.2010.08.058. URL https://doi.org/10.1016/j.jmb.2010.08.058.

Derek N Woolfson, Gail J Bartlett, Antony J Burton, Jack W Heal, Ai Niitsu, Andrew R Thomson, and Christopher W Wood. De novo protein design: how do we expand into the universe of possible protein structures? *Current Opinion in Structural Biology*, 33:16–26, August 2015. doi: 10.1016/j.sbi.2015.05.009. URL https://doi.org/10.1016/j.sbi.2015.05.009.

Joseph L Beesley and Derek N Woolfson. The de novo design of α-helical peptides for supramolecular self-assembly. *Current Opinion in Biotechnology*, 58:175–182, August 2019. doi: 10.1016/j.copbio.2019.03.017. URL https://doi.org/10.1016/j.copbio.2019.03.017.

Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

Dimitra Maoutsa, Sebastian Reich, and Manfred Opper. Interacting particle solutions of fokker–planck equations through gradient–log–density estimation. *Entropy*, 22(8):802, 2020.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

Bowen Jing, Gabriele Corso, Renato Berlinghieri, and Tommi Jaakkola. Subspace diffusion generative models. *arXiv preprint arXiv:2205.01490*, 2022.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

Wouter Kool, Herke Van Hoof, and Max Welling. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. In *International Conference on Machine Learning*, pages 3499–3508. PMLR, 2019.

David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

Enzo Marinari and Giorgio Parisi. Simulated tempering: a new monte carlo scheme. *EPL (Europhysics Letters)*, 19(6):451, 1992.

Ulrich HE Hansmann. Parallel tempering algorithm for conformational studies of biological molecules. *Chemical Physics Letters*, 281(1-3):140–150, 1997.

Liu Hong and Jinzhi Lei. Scaling law for the radius of gyration of proteins and its dependence on hydrophobicity. *Journal of Polymer Science Part B: Polymer Physics*, 47(2):207–214, 2009. doi: https://doi.org/10.1002/polb.21634. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/polb.21634.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33:17283–17297, 2020.

Jianwei Yang, Chunyuan Li, Pengchuan Zhang, Xiyang Dai, Bin Xiao, Lu Yuan, and Jianfeng Gao. Focal attention for long-range interactions in vision transformers. In *Advances in Neural Information Processing Systems*, volume 34, pages 30008–30022. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/fc1a36821b02abbd2503fd949bfc9131-Paper.pdf.

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *9th ISCA Speech Synthesis Workshop*, pages 125–125, 2016.

Mohammed AlQuraishi. End-to-end differentiable learning of protein structure. *Cell systems*, 8 (4):292–301, 2019.

Andrew W Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander WR Nelson, Alex Bridgland, et al. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, 2020.

Debora S Marks, Thomas A Hopf, and Chris Sander. Protein structure prediction from sequence variation. *Nature biotechnology*, 30(11):1072–1080, 2012.

David Belanger and Andrew McCallum. Structured prediction energy networks. In *International Conference on Machine Learning*, pages 983–992. PMLR, 2016.

John Ingraham, Adam Riesselman, Chris Sander, and Debora Marks. Learning protein structure with a differentiable simulator. In *International Conference on Learning Representations*, 2018.

Samuel Schoenholz and Ekin Dogus Cubuk. Jax md: a framework for differentiable physics. *Advances in Neural Information Processing Systems*, 33:11428–11441, 2020.

Wujie Wang, Simon Axelrod, and Rafael Gómez-Bombarelli. Differentiable molecular simulations for control and learning. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

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):922–923, 1976.

Kevin P Murphy. Conjugate bayesian analysis of the gaussian distribution. *def*, 1(2σ2):16, 2007.

Alex J Li, Vikram Sundar, Gevorg Grigoryan, and Amy E Keating. Terminator: A neural framework for structure-based protein design using tertiary repeating motifs. *arXiv preprint arXiv:2204.13048*, 2022.

Robert C. Edgar. Search and clustering orders of magnitude faster than BLAST. *Bioinformatics*, 26(19):2460–2461, 08 2010. ISSN 1367-4803. doi: 10.1093/bioinformatics/btq461. URL https://doi.org/10.1093/bioinformatics/btq461.

Sidhartha Chaudhury, Sergey Lyskov, and Jeffrey J Gray. Pyrosetta: a script-based interface for implementing molecular modeling algorithms using rosetta. *Bioinformatics*, 26(5):689–691, 2010.

Jaina Mistry, Sara Chuguransky, Lowri Williams, Matloob Qureshi, Gustavo A Salazar, Erik LL Sonnhammer, Silvio CE Tosatto, Lisanna Paladin, Shriya Raj, Lorna J Richardson, et al. Pfam: The protein families database in 2021. *Nucleic acids research*, 49(D1):D412–D419, 2021.

Alex Bateman, Maria-Jesus Martin, Sandra Orchard, Michele Magrane, Rahat Agivetova, Shadab Ahmad, Emanuele Alpi, Emily H Bowler-Barnett, Ramona Britto, Borisas Bursteinas, Hema Bye-A-Jee, Ray Coetzee, Austra Cukura, Alan Da Silva, Paul Denny, Tunca Dogan, ThankGod Ebenezer, Jun Fan, Leyla Garcia Castro, Penelope Garmiri, George Georghiou, Leonardo Gonzales, Emma Hatton-Ellis, Abdulrahman Hussein, Alexandr Ignatchenko, Giuseppe Insana, Rizwan Ishtiaq, Petteri Jokinen, Vishal Joshi, Dushyanth Jyothi, Antonia Lock, Rodrigo Lopez, Aurelien Luciani, Jie Luo, Yvonne Lussi, Alistair MacDougall, Fabio Madeira, Mahdi Mahmoudy, Manuela Menchi, Alok Mishra, Katie Moulang, Andrew Nightingale, Carla Susana Oliveira, Sangya Pundir, Guoying Qi, Shriya Raj, Daniel Rice, Milagros Rodriguez Lopez, Rabie Saidi, Joseph Sampson, Tony Sawford, Elena Speretta, Edward Turner, Nidhi Tyagi, Preethi Vasudev, Vladimir Volynkin, Kate Warner, Xavier Watkins, Rossana Zaru, Hermann Zellner, Alan Bridge, Sylvain Poux, Nicole Redaschi, Lucila Aimo, Ghislaine Argoud-Puy, Andrea Auchincloss, Kristian Axelsen, Parit Bansal, Delphine Baratin, Marie-Claude Blatter, Jerven Bolleman, Emmanuel Boutet, Lionel Breuza, Cristina Casals-Casas, Edouard de Castro, Kamal Chikh Echioukh, Elisabeth Coudert, Beatrice Cuche, Mikael Doche, Dolnide Dornevil, Anne Estreicher, Maria Livia Famiglietti, Marc Feuermann, Elisabeth Gasteiger, Sebastien Gehant, Vivienne Gerritsen, Arnaud Gos, Nadine Gruaz-Gumowski, Ursula Hinz, Chantal Hulo, Nevila Hyka-Nouspikel, Florence Jungo, Guillaume Keller, Arnaud Kerhornou, Vicente Lara, Philippe Le Mercier, Damien Lieberherr, Thierry Lombardot, Xavier Martin, Patrick Masson, Anne Morgat, Teresa Batista Neto, Salvo Paesano, Ivo Pedruzzi, Sandrine Pilbout, Lucille Pourcel, Monica Pozzato, Manuela Pruess, Catherine Rivoire, Christian Sigrist, Karin Sonesson, Andre Stutz, Shyamala Sundaram, Michael Tognolli, Laure Verbregue, Cathy H Wu, Cecilia N Arighi, Leslie Arminski, Chuming Chen, Yongxing Chen, John S Garavelli, Hongzhan Huang, Kati Laiho, Peter McGarvey, Darren A Natale, Karen Ross, C R Vinayaka, Qinghua Wang, Yuqi Wang, Lai-Su Yeh, Jian Zhang, Patrick Ruch, and Douglas Teodoro. UniProt: the universal protein knowledgebase in 2021. *Nucleic Acids Research*, 49(D1):D480–D489, November 2020. doi: 10.1093/nar/gkaa1100. URL https://doi.org/10.1093/nar/gkaa1100.

Martin Steinegger and Johannes Söding. Mmseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature biotechnology*, 35(11):1026–1028, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

D Frishman and P Argos. Knowledge-based protein secondary structure assignment. *Proteins*, 23 (4):566–579, 1995.

DN Ivankov, SO Garbuzynskiy, E Alm, KW Plaxco, D Baker, and AV Finkelstein. Contact order revisited: influence of protein size on the folding rate. *Protein Sci*, 12(9):2057–2062, 2003.

M Borg, K Mardia, W Boomsma, J Frellsen, T Harder, K Stovgaard, J Ferkinghoff-Borg, P Røgen, and T Hamelryck. A probabilistic approach to protein structure prediction: Phaistos in casp9. *LASR2009-Statistical tools for challenges in bioinformatics*, pages 65–70, 2009.

John C. Wootton and Scott Federhen. Statistics of local complexity in amino acid sequences and sequence databases. *Computers & Chemistry*, 17(2):149–163, June 1993. doi: 10.1016/0097-8485(93)85006-x. URL https://doi.org/10.1016/0097-8485(93)85006-x.

Jianfu Zhou and Gevorg Grigoryan. Rapid search for tertiary fragments reveals protein sequence-structure relationships. *Protein Science*, 24(4):508–524, December 2014. doi: 10.1002/pro.2610. URL https://doi.org/10.1002/pro.2610.

Jianfu Zhou and Gevorg Grigoryan. A c++ library for protein sub-structure search. April 2020a. doi: 10.1101/2020.04.26.062612. URL https://doi.org/10.1101/2020.04.26.062612.

K T Simons, C Kooperberg, E Huang, and D Baker. Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and bayesian scoring functions. *J. Mol. Biol.*, 268(1):209–225, April 1997.

Will Grathwohl, Kevin Swersky, Milad Hashemi, David Duvenaud, and Chris Maddison. Oops i took a gradient: Scalable sampling for discrete distributions. In *International Conference on Machine Learning*, pages 3831–3841. PMLR, 2021.

Benjamin Rhodes and Michael Gutmann. Enhanced gradient-based mcmc in discrete spaces. *arXiv preprint arXiv:2208.00040*, 2022.

Jianfu Zhou and Gevorg Grigoryan. Rapid search for tertiary fragments reveals protein sequence-structure relationships. *Protein Sci.*, 24(4):508–524, April 2015.

Jianfu Zhou and Gevorg Grigoryan. A c++ library for protein sub-structure search. *bioRxiv preprint 2020.04.26.062612*, 2020b.

David S Goodsell and Arthur J Olson. Structural symmetry and protein function. *Annual review of biophysics and biomolecular structure*, 29:105, 2000.

Yang Hsia, Jacob B Bale, Shane Gonen, Dan Shi, William Sheffler, Kimberly K Fong, Una Nattermann, Chunfu Xu, Po-Ssu Huang, Rashmi Ravichandran, et al. Design of a hyperstable 60-subunit protein icosahedron. *Nature*, 535(7610):136–139, 2016.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.

Sam Cox and Andrew D White. Symmetric molecular dynamics. *arXiv preprint arXiv:2204.01114*, 2022.

Hagit Zabrodsky, Shmuel Peleg, and David Avnir. Continuous symmetry measures. *Journal of the American Chemical Society*, 114(20):7843–7851, 1992.

Stephen C Harvey, Robert K-Z Tan, and Thomas E Cheatham III. The flying ice cube: velocity rescaling in molecular dynamics leads to violation of energy equipartition. *Journal of computational chemistry*, 19(7):726–740, 1998.

Justin Solomon, Gabriel Peyré, Vladimir G Kim, and Suvrit Sra. Entropic metric alignment for correspondence problems. *ACM Transactions on Graphics (ToG)*, 35(4):1–13, 2016.

David Alvarez-Melis and Tommi S Jaakkola. Gromov-wasserstein alignment of word embedding spaces. In *EMNLP*, 2018.

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020.

H. M. Berman. The protein data bank. *Nucleic Acids Research*, 28(1):235–242, January 2000. doi: 10.1093/nar/28.1.235. URL `https://doi.org/10.1093/nar/28.1.235`.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow, March 2021. URL `https://doi.org/10.5281/zenodo.5297715`.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling. *CoRR*, abs/2101.00027, 2021. URL `https://arxiv.org/abs/2101.00027`.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *CoRR*, abs/2104.08691, 2021. URL `https://arxiv.org/abs/2104.08691`.

Supplementary Information

Table of Contents

List of Figures

Symbol	Definition
N	number of atoms
$G = \{g_i\}_{i=1}^{\lvert G\rvert}$	a group, g_i is an individual group element in $E(3)$
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	a graph and $\mathcal{G}(\mathbf{x})$ defined as the graph generation operation
$\mathbf{x}_t \in \mathbb{R}^{N\times 3}$	atom coordinate sampled at time t
$\mathbf{x}_t^{\mathcal{M}} \in \mathbb{R}^{\lvert\mathcal{M}\rvert\times 3}$	retrieve atom coordinates based on index set $\mathcal{M} \subset [\![1, N]\!]$
$\mathbf{x}_t^{(i)} \in \mathbb{R}^3$	the i_{th} coordinate in $\mathbf{x}_t$
$\mathbf{X}_t \in \mathbb{R}^{\lvert G\rvert\times N\times 3}$	coordinates of G-symmetric complex
$\mathbf{X}_t^{(i)} \in \mathbb{R}^{\lvert G\rvert\times N\times 3}$	coordinates of the i_{th} subunit in the G-symmetric complex
D_{ij}	Euclidean distance between i and j $\Vert\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\Vert_2$
$d_t^{(ij)}$	time-dependent noised Euclidean distance between i and j
$\mathbf{z} \in \mathbb{R}^{N\times 3}$	whitened noise, and z_i is the individual noise component
$\Sigma = \mathbf{R}\mathbf{R}^\intercal$	covariance matrix for polymer-structured prior, $[\mathbf{Rz}]_{ik} = \sum_j [\mathbf{R}]_{ij}\mathbf{z}_{jk}$
$\mathbf{T} = (\mathbf{t}, \mathbf{O})$	Euclidean transformation with translation $\mathbf{t}$ and rotation $\mathbf{O}$
β_t	time-dependent noise schedule
α_t	integrated noise in the forward diffusion
λ_t	time-dependent inverse temperature
ψ	mixing parameter for amount of Langevin equilibration in Hybrid SDE
T	number of integration time steps
$\hat{\mathbf{x}}_\theta$	denoising network in Cartesian space
$\hat{\mathbf{z}}_\theta$	denoising network in the whitened space
$\nabla_{\mathbf{x}} \log p_t(\mathbf{x}, t)$	score estimator network
$\mathrm{d}\mathbf{w}$, $\mathrm{d}\bar{\mathbf{w}}$	forward Brownian noise, reverse Brownian noise

Table 1: **Table of notation**

A Diffusion Models with Structured Correlations

A.1 Correlated diffusion as uncorrelated diffusion in whitened space

Correlation and diffusion Most natural data possess a hierarchy of correlation structures, some of which are very simple (e.g., most nearby pixels in natural images will tend to be a similar color) and some of which are very subtle (e.g., complex constraints govern the set of pixels forming an eye or a cat). With finite computing resources and modeling power, it can be advantageous to design learning systems that capture simple correlations as efficiently as possible such that most model capacity can be dedicated to nontrivial dependencies (see Appendix C).

Diffusion models capture complex constraints in the data by learning to reverse a diffusion process that transforms data into noise [Sohl-Dickstein et al., 2015, Song et al., 2021]. While most of these original diffusion frameworks considered the possibility of correlated noise, it is typical in contemporary models to use isotropic noise that is standard normally distributed. In this configuration, models must learn both simple correlations and complex correlations in data *from scratch*.

Whitening transformations and linear generative models One classical approach for removing nuisance correlations in data is to apply a “whitening transformation”, i.e., an affine linear transformation $\mathbf{z} = \Sigma^{-\frac{1}{2}}(\mathbf{x} - \boldsymbol{\mu})$ that decorrelates all factors of variation by subtracting the empirical mean $\boldsymbol{\mu}$ and multiplying by a square root of the inverse covariance matrix $\mathbf{R} = \Sigma^{-\frac{1}{2}}$.

Whitening data can also be related to fitting the data to a Gaussian model $\mathbf{x} = F(\mathbf{z}) = \mathbf{R}\mathbf{z} + \mathbf{b}$ where the whitened factors $\mathbf{z}$ are standard normally distributed as $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ [Murphy, 2012]. The density in the whitened space can be related to the density in the transformed space by the change of variables formula as

$$
\begin{aligned}
\log p(\mathbf{x}) &= \log p_{\mathbf{z}}(F^{-1}(\mathbf{x})) - \log \left| \det \frac{dF}{d\mathbf{x}} \right| \\
&= \log p_{\mathbf{z}}(\mathbf{R}^{-1}(\mathbf{x} - \mathbf{b})) - \log |\det \mathbf{R}| \\
&= \log \mathcal{N}(\mathbf{R}^{-1}(\mathbf{x} - \mathbf{b}); \mathbf{0}, \mathbf{I}) - \log |\det \mathbf{R}| \\
&= \log \mathcal{N}(\mathbf{x}; \mathbf{b}, \mathbf{R}\mathbf{R}^{\intercal}).
\end{aligned}
$$

From whitened diffusion to dewhitened diffusion If we possess a linear Gaussian prior for our data $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \mathbf{b}, \mathbf{R}\mathbf{R}^{\intercal})$ which can be sampled as $\mathbf{x} = \mathbf{R}\mathbf{z}$ with $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$[1], then an *uncorrelated* diffusion process on the whitened coordinates $\mathbf{z}_t \sim p_t(\mathbf{z}|\mathbf{z}_0)$ will induce a *correlated* diffusion process on the original coordinates $\mathbf{x}_t \sim p_t(\mathbf{x}|\mathbf{x}_0)$. When the diffusion process is the so-called Variance-Preserving (VP) diffusion [Sohl-Dickstein et al., 2015, Ho et al., 2020], then the diffusion will transition from the data distribution at time $t = 0$ to the Gaussian prior distribution at time $t = 1$. Throughout this work we use the continuous time formulation of VP diffusion [Song et al., 2021]

[1] We will assume the data are centered (have zero mean) for ease of notation.

in whitened space. This process evolves in time $t \in (0,1)$ according to the Stochastic Differential Equation (SDE)

$$\mathrm{d}\mathbf{z} = -\frac{\beta_t}{2}\,\mathbf{z}\,\mathrm{d}t + \sqrt{\beta_t}\mathrm{d}\mathbf{w},$$

where $\mathbf{w}$ is a standard Wiener process and β_t is the time-dependent schedule at which noise is injected into the process. We can also write the correlated SDE in terms of $\mathbf{x}_t$ if we substitute $\mathbf{x} = \mathbf{R}\mathbf{z}$ as

$$\begin{aligned}\mathrm{d}\mathbf{x} = \mathbf{R}\mathrm{d}\mathbf{z} &= -\frac{\beta_t}{2}\,\mathbf{R}\,\mathbf{z}\,\mathrm{d}t + \sqrt{\beta_t}\mathbf{R}\mathrm{d}\mathbf{w} \\ &= -\frac{\beta_t}{2}\,\mathbf{x}\,\mathrm{d}t + \sqrt{\beta_t}\mathbf{R}\mathrm{d}\mathbf{w}.\end{aligned}$$

Sampling from the diffusion This diffusion process is simple to integrate forward in time [Sohl-Dickstein et al., 2015, Song et al., 2021]. Given an initial data point $\mathbf{x}_0$, then $\mathbf{x}_t$ will be distributed as $\mathbf{x}_t \sim \mathcal{N}(\mathbf{x}; \sqrt{\alpha_t}\mathbf{x}_0, (1-\alpha_t)\mathbf{R}\mathbf{R}^\intercal)$ where $\alpha_t = \int_0^t \exp(-\beta_s)ds$ is the integrated noise. Samples at any time t can thus be generated from standard normally distributed noise as

$$\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\mathbf{R}\epsilon \qquad\qquad \epsilon \sim \mathcal{N}(\mathbf{0},\mathbf{I}).$$

A.2 Evidence Lower Bound (ELBO)

Denoising loss Diffusion models can be parameterized in terms of a denoising neural network $\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{x},t)$ that is trained to predict $\mathbf{x}_0$ given a noisy sample $\mathbf{x}_t$. Typically this is done by minimizing a *denoising loss*

$$\mathcal{L}(\mathbf{x};\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x}_t \sim p(\mathbf{x}_t|\mathbf{x}), t\sim \mathrm{Unif}(0,1)}\left[\tau_t \|\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{x}_t,t) - \mathbf{x}\|_2^2\right]$$

where τ_t is a time-dependent weighting to emphasize the loss at particular points in time (noise levels) [Song et al., 2021]. Training with this loss can be directly related to *score matching* and *noise prediction* which can be cast as alternative parameterizations of the target output of the network [Kingma et al., 2021].

Approximate likelihood bound We train the diffusion model by optimizing an approximate bound on the marginal likelihood of data together with a regularization loss. Following [Kingma et al., 2021], the negative Evidence Lower Bound (ELBO) in whitened space is

$$\mathcal{L}_{\mathbf{z}}(\mathbf{z};\boldsymbol{\theta}) = -\mathbb{E}_{\mathbf{x}_t \sim p(\mathbf{x}_t|\mathbf{x}), t\sim \mathrm{Unif}(0,1)}\left[\frac{\alpha_t\beta_t}{2(1-\alpha_t)^2}\|\hat{\mathbf{z}}_{\boldsymbol{\theta}}(\mathbf{x}_t,t) - \mathbf{z}\|_2^2\right]$$

where in this case $\tau_t = -\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\alpha_t}{1-\alpha_t}\right) = -\frac{\alpha_t\beta_t}{2(1-\alpha_t)^2}$ is the derivative of the *signal to noise ratio* $\frac{\alpha_t}{1-\alpha_t}$. To express this loss in terms of $\mathbf{x} = \mathbf{R}\mathbf{z}$ we may again apply change of variables formula to obtain

$$\begin{aligned}\mathcal{L}_{\mathbf{x}}(\mathbf{x};\boldsymbol{\theta}) &= \mathcal{L}_{\mathbf{z}}(\mathbf{R}^{-1}\mathbf{x}) + \log|\det\mathbf{R}| \\ &= -\mathbb{E}_{\mathbf{x}_t \sim p(\mathbf{x}_t|\mathbf{x}), t\sim \mathrm{Unif}(0,1)}\left[\frac{\alpha_t\beta_t}{2(1-\alpha_t)^2}\|\mathbf{R}^{-1}\left(\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{x}_t,t) - \mathbf{x}\right)\|_2^2\right] + \log|\det\mathbf{R}|,\end{aligned}$$

where the term $\log|\det \mathbf{R}|$ is a constant (we fit the parameters of $\mathbf{R}$ offline from training) and can be ignored during optimization of the denoiser parameters $\boldsymbol{\theta}$.

Regularized loss In practice we optimize a regularized variant of the ELBO loss which is the sum of whitened and unwhitened reconstruction errors as

$$\mathcal{L}_{\mathbf{x}}^{\text{reg}}(\mathbf{x};\boldsymbol{\theta}) = -\mathbb{E}_{\mathbf{x}_t \sim p(\mathbf{x}_t|\mathbf{x}), t\sim \text{Unif}(0,1)}\left[\frac{\alpha_t \beta_t}{2(1-\alpha_t)^2}\left\|\left(\mathbf{R}^{-1}+\omega \mathbf{I}\right)\left(\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)-\mathbf{x}\right)\right\|_2^2\right],$$

where $\mathbf{I}$ is the identity matrix and we set the scale factor ω to give $\mathbf{x}$ units of nanometers. We found this regularization to be important because in practice we care about absolute errors in $\mathbf{x}$ space, i.e. absolute spatial errors, at least as much as we care about errors in $\mathbf{z}$ space, which will correspond under our covariance models (Appendix C) to relative local chain geometries.

A.3 Reverse-time SDE

In whitened space, we can express the *reverse-time* dynamics for the forwards-time SDE in terms of another SDE [Anderson, 1982, Song et al., 2021] that depends on the *score function* of the time-dependent marginals $\nabla_{\mathbf{z}} \log p_t(\mathbf{z})$ as

$$\mathrm{d}\mathbf{z} = \left(-\frac{1}{2}\,\mathbf{z} - \nabla_{\mathbf{z}} \log p_t(\mathbf{z})\right)\beta_t \,\mathrm{d}t + \sqrt{\beta_t}\,\mathrm{d}\bar{\mathbf{w}}.$$

We can similarly express this in the score function of the transformed coordinate system as

$$\mathrm{d}\mathbf{x} = \left(-\frac{1}{2}\,\mathbf{x} - \mathbf{R}\mathbf{R}^{\intercal}\nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right)\beta_t \,\mathrm{d}t + \sqrt{\beta_t}\,\mathbf{R}\,\mathrm{d}\bar{\mathbf{w}}.$$

To sample from the diffusion model by taking a sample from the "prior" (time 1 distribution) and integrate the SDE above backward in time from $t=1$ to $t=0$. We can rewrite the above SDE in terms of our optimal denoising network $\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{x},t)$ (trained as described above) by leveraging the relationship [Song et al., 2021, Kingma et al., 2021] that

$$\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) = \left((1-\alpha_t)\mathbf{R}\mathbf{R}^{\intercal}\right)^{-1}\left(\sqrt{\alpha_t}\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{x},t)-\mathbf{x}\right).$$

Therefore we can express the reverse-time SDE in terms of the optimal denoising network $\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{x},t)$ as

$$\begin{aligned}
\mathrm{d}\mathbf{x} &= \left(-\frac{1}{2}\,\mathbf{x} - \mathbf{R}\mathbf{R}^{\intercal}\frac{(\mathbf{R}\mathbf{R}^{\intercal})^{-1}}{1-\alpha_t}\left(\sqrt{\alpha_t}\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{x},t)-\mathbf{x}\right)\right)\beta_t\,\mathrm{d}t + \sqrt{\beta_t}\,\mathbf{R}\,\mathrm{d}\bar{\mathbf{w}} \\
&= \left(-\frac{1}{2}\,\mathbf{x} - \frac{\sqrt{\alpha_t}\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{x},t)-\mathbf{x}}{1-\alpha_t}\right)\beta_t\,\mathrm{d}t + \sqrt{\beta_t}\,\mathbf{R}\,\mathrm{d}\bar{\mathbf{w}} \\
&= \left(\frac{-\sqrt{\alpha_t}\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{x},t)+\mathbf{x}-\frac{1}{2}\,\mathbf{x}(1-\alpha_t)}{1-\alpha_t}\right)\beta_t\,\mathrm{d}t + \sqrt{\beta_t}\,\mathbf{R}\,\mathrm{d}\bar{\mathbf{w}} \\
&= \left(\frac{\alpha_t+1}{2(1-\alpha_t)}\mathbf{x} - \frac{\sqrt{\alpha_t}}{1-\alpha_t}\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{x},t)\right)\beta_t\,\mathrm{d}t + \sqrt{\beta_t}\,\mathbf{R}\,\mathrm{d}\bar{\mathbf{w}}.
\end{aligned}$$

A.4 Probability Flow ODE

Probability Flow ODE for deterministic encoding and sampling Remarkably, it is also possible to derive a set of deterministic *ordinary differential equations* (ODEs) whose marginal evolution from the prior is identical to above SDEs [Song et al., 2021, Maoutsa et al., 2020]. In the context of our covariance model this can be expressed either in terms of the score function $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ as

$$\frac{d\mathbf{x}}{dt} = -\frac{\beta_t}{2}\left(\mathbf{x} + \mathbf{R}\mathbf{R}^\intercal \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right),$$

or in terms of the optimal denoiser network $\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{x}, t)$ as

$$\begin{aligned}
\frac{d\mathbf{x}}{dt} &= -\frac{\beta_t}{2}\left(\mathbf{x} + \mathbf{R}\mathbf{R}^\intercal\left((1-\alpha_t)\,\mathbf{R}\mathbf{R}^\intercal\right)^{-1}\left(\sqrt{\alpha_t}\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{x},t) - \mathbf{x}\right)\right) \\
&= -\frac{\beta_t}{2}\left(\mathbf{x} + (1-\alpha_t)^{-1}\left(\sqrt{\alpha_t}\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{x},t) - \mathbf{x}\right)\right) \\
&= -\frac{\beta_t}{2}\left(\mathbf{x}\left(1 - \frac{1}{1-\alpha_t}\right) + \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{x},t)\frac{\sqrt{\alpha_t}}{1-\alpha_t}\right) \\
&= \frac{\beta_t}{2}\left(\mathbf{x}\frac{\alpha_t}{1-\alpha_t} - \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{x},t)\frac{\sqrt{\alpha_t}}{1-\alpha_t}\right) \\
&= \frac{1}{2}\frac{\alpha_t\beta_t}{1-\alpha_t}\left(\mathbf{x} - \frac{\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\sqrt{\alpha_t}}\right).
\end{aligned}$$

The ODE formulation of sampling is especially important because it enables reformulating the model as a Continuous Normalizing Flow [Chen et al., 2018], which can admit efficient and exact likelihood calculations using the adjoint method [Grathwohl et al., 2018].

A.5 Conditional sampling from the posterior under auxiliary constraints

Bayesian posterior SDE for conditional sampling An extremely powerful aspect of the reverse diffusion formulation is that it can also be extended to enable conditional sampling from a Bayesian posterior $p(\mathbf{x}|\mathbf{y})$ by combining with auxilliary classifiers $\log p_t(\mathbf{y}|\mathbf{x})$ and without re-training the base diffusion model [Song et al., 2021]. When extended to the correlated diffusion case, this gives the SDE

$$\begin{aligned}
\mathrm{d}\mathbf{x} &= \left(-\frac{1}{2}\,\mathbf{x} - \;\mathbf{R}\mathbf{R}^\intercal\left(\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) + \nabla_{\mathbf{x}} \log p_t(\mathbf{y}|\mathbf{x})\right)\right)\beta_t\;\mathrm{d}t + \sqrt{\beta_t}\;\mathbf{R}\;\mathrm{d}\bar{\mathbf{w}} && (1) \\
&= \left(\frac{\alpha_t+1}{2(1-\alpha_t)}\mathbf{x} - \frac{\sqrt{\alpha_t}}{1-\alpha_t}\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{x},t) - \;\mathbf{R}\mathbf{R}^\intercal\nabla_{\mathbf{x}} \log p_t(\mathbf{y}|\mathbf{x})\right)\beta_t\;\mathrm{d}t + \sqrt{\beta_t}\;\mathbf{R}\;\mathrm{d}\bar{\mathbf{w}} && (2)
\end{aligned}$$

Bayesian posterior ODE for conditional sampling In the context of our covariance model and conditional constraints, the Probability Flow ODE for sampling from the posterior is

$$\frac{d\mathbf{x}}{dt} = -\frac{\beta_t}{2}\left(\mathbf{x} + \mathbf{R}\mathbf{R}^\intercal\left(\nabla_\mathbf{x}\log p_t(\mathbf{x}) + \nabla_\mathbf{x}\log p_t(\mathbf{y}|\mathbf{x})\right)\right) \tag{3}$$

$$= \frac{1}{2}\frac{\alpha_t\beta_t}{1-\alpha_t}\left(\mathbf{x} - \frac{\hat{\mathbf{x}}_\theta(\mathbf{x},t)}{\sqrt{\alpha_t}}\right) - \frac{\beta_t}{2}\mathbf{R}\mathbf{R}^\intercal\nabla_\mathbf{x}\log p_t(\mathbf{y}|\mathbf{x}) \tag{4}$$

A.6 Related work

Subspace diffusion models [Jing et al., 2022] also consider correlated diffusion, with a particular emphasis on focusing the diffusion to most relevant factors of variation for statistical and computational efficiency. Additionally, latent-space diffusion models [Rombach et al., 2022] might be viewed as learning a transformed coordinate system in which the diffusion process can more efficiently model the targer distribution. Our work provides further evidence for how correlated diffusion may be an underutilized approach to distributional modeling and shows how domain knowledge can be incorporated in the form of simple constraints on the covariance structure of the noise process.

B Low-Temperature Sampling for Diffusion Models

Maximum likelihood training of generative models enforces a tolerable probability of *all* datapoints and, as a result, misspecified or low-capacity models fit by maximum likelihood will typically be overdispersed. This can be understood through the perspective that maximizing likelihood is equivalent to minimizing the KL divergence from the model to the data distribution, which is the mean-seeking and mode-covering direction of KL divergence.

To mitigate overdispersion in generative models, it is common practice to introduce modified sampling procedures that increase sampling of high-likelihood states (mode emphasis, precision) at the expense of reduced sample diversity (mode coverage, recall). This includes approaches such as shrunken encodings in normalizing flows [Kingma and Dhariwal, 2018], low-temperature greedy decoding algorithms for language models [Holtzman et al., 2020], and stochastic beam search [Kool et al., 2019].

A powerful but often intractable way to trade diversity for quality in generative models is low-temperature sampling. This involves perturbing a base distribution $p(\mathbf{x})$ by exponentiating with an inverse temperature rescaling factor λ and renormalizing as $p_\lambda(\mathbf{x}) = \frac{1}{Z}p(\mathbf{x})^\lambda$. As the inverse temperature becomes large $\lambda \ll 1$, this perturbed distribution will trade diversity (entropy) for sample quality (likelihood) and ultimately will collapse into the global optimum as $\lambda \to \infty$. Unfortunately, low temperature sampling in the general case will require expensive iterative sampling methods such as Markov Chain Monte Carlo which typically offer no guarantee of convergence in a practical amount of time [MacKay, 2003].

Low temperature and diffusion models The issue of trading diversity for sample quality in diffusion models has been discussed previously, with some authors reporting that simple modifications like upscaling the score function and/or downscaling the noise were ineffective [Dhariwal and Nichol, 2021]. Instead, classifier guidance and classifier-free guidance have been widely adopted as critical components of contemporary text-to-image diffusion models such as Imagen and DALL-E 2 [Ho and Salimans, 2021, Saharia et al., 2022, Ramesh et al., 2022].

Equilibrium versus Non-Equilibrium Sampling Here we offer an explanation for why these previous attempts at low temperature sampling did not work and produce a novel algorithm for low-temperature sampling from diffusion models. We make two key observations, explained in the next two sections

1. **Upscaling the score function of the reverse SDE is insufficient** to properly re-weight populations in a temperature perturbed distribution.
2. **Annealed Langevin dynamics *can* sample from low temperature distributions** if given sufficient equilibration time.

B.1 Reverse-time SDE with temperature annealing

The isotropic Gaussian case To determine how the Reverse SDE can be modified to enable (approximate) low temperature sampling, it is helpful to first consider a case that can be treated exactly: transforming a Gaussian data distribution $\mathcal{N}(\mathbf{x}_0; \boldsymbol{\mu}_{\text{data}}, \sigma^2_{\text{data}})$ to a Gaussian prior $\mathcal{N}(\mathbf{x}_1; 0, \sigma^2_{\text{prior}})$. Under the Variance-Preserving diffusion, the time-dependent marginal density will be given by

$$p_t(\mathbf{x}) = \mathcal{N}\left(\mathbf{x}; \sqrt{\alpha_t}\boldsymbol{\mu}_{\text{data}},\ \alpha_t\sigma^2_{\text{data}} + (1-\alpha_t)\sigma^2_{\text{prior}}\right),$$

which means that the score function $\mathbf{s}_t$ will be

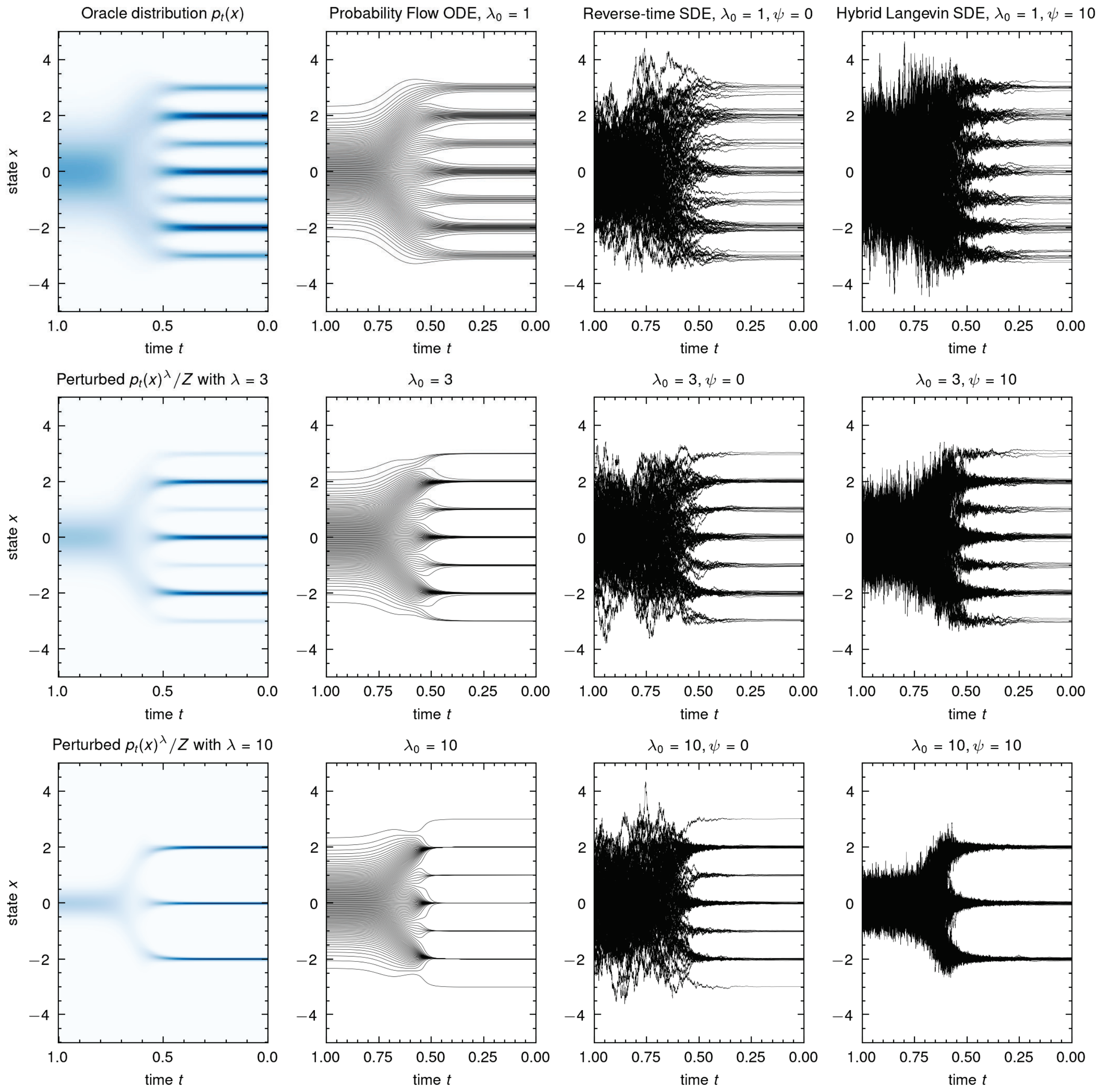
$$\begin{aligned}\mathbf{s}_t &\triangleq \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \\ &= \frac{\sqrt{\alpha_t}\boldsymbol{\mu}_{\text{data}} - \mathbf{x}}{\alpha_t\sigma^2_{\text{data}} + (1-\alpha_t)\sigma^2_{\text{prior}}}.\end{aligned}$$

Now, suppose we wish to *modify* the definition of the time-dependent score function so that, instead of transitioning to the original data distribution, it transforms to the perturbed data distribution, i.e. so the it transitions to $\frac{1}{Z}p_0(\mathbf{x})^{\lambda_0}$. For a Gaussian, this operation will simply multiply the precision (or equivalently, divide the covariance) by the factor λ_0. The perturbed score function will therefore be

$$\mathbf{s}_t^{\text{perturb}} = \frac{\sqrt{\alpha_t}\boldsymbol{\mu}_{\text{data}} - \mathbf{x}}{\alpha_t\sigma^2_{\text{data}}/\lambda_0 + (1-\alpha_t)\sigma^2_{\text{prior}}}.$$

Based on this, we can express the perturbed score function as a *time-dependent rescaling* of the original score function with scaling based on the ratios of the time-dependent inverse variances as

$$\mathbf{s}_t^{\text{perturb}} = \mathbf{s}_t \frac{(1-\alpha_t)\sigma^2_{\text{prior}} + \alpha_t\sigma^2_{\text{data}}}{(1-\alpha_t)\sigma^2_{\text{prior}} + \alpha_t\sigma^2_{\text{data}}/\lambda_0}.$$



Supplementary Figure 1: **The Hybrid Langevin SDE can sample from temperature-perturbed distributions.** The marginal densities of the diffusion process $p_t(x)$ (top left) gradually transform between a toy 1D data distribution at time $t = 0$ and a standard normal distribution at time $t = 1$. Reweighting the distribution by inverse temperature λ_0 as $\frac{1}{Z}p_t(x)^{\lambda_0}$ (left column, bottom two rows) will both concentrate and reweight the population distributions. The annealed versions of the reverse-time SDE and Probability Flow ODEs (middle columns) can concentrate towards local optima but do not correctly reweight the relative population occupancies. Adding in Langevin dynamics with the Hybrid Langevin xSDE (right column) increases the rate of equilibration to the time-dependent marginals and, when combined with low temperature rescaling, successfully reweights the populations (bottom right).

Therefore we see that, to achieve a particular inverse temperature λ_0 for the data distribution, we should rescale the learned score function by time-dependent factor

$$\lambda_t = \frac{(1-\alpha_t)\sigma^2_{\text{prior}} + \alpha_t \sigma^2_{\text{data}}}{(1-\alpha_t)\sigma^2_{\text{prior}} + \alpha_t \sigma^2_{\text{data}}/\lambda_0} \approx \frac{\lambda_0}{\alpha_t + (1-\alpha_t)\lambda_0}$$

where in the last step we assumed $\sigma^2_{\text{data}} = \sigma^2_{\text{prior}}$. So one interpretation of the previously observed insufficienes of low temperature sampling based on score-rescaling [Dhariwal and Nichol, 2021] is that these were hampered by *uniform* rescaling the score function in time instead of in a way that accounts for the shift of influence between the prior and the data distribution.

Temperature-adjusted reverse time SDE We can modify the Reverse-time SDE by simply rescaling the score function with the above time-dependent temperature rescaling as

$$\begin{aligned} \mathrm{d}\mathbf{x} &= \left(-\frac{1}{2}\,\mathbf{x} - \ \lambda_t \mathbf{R}\mathbf{R}^\intercal \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right) \beta_t \,\mathrm{d}t + \sqrt{\beta_t}\,\mathbf{R}\,\mathrm{d}\bar{\mathbf{w}} \\ &= \left(-\frac{1}{2}\,\mathbf{x} - \lambda_t \frac{\sqrt{\alpha_t}\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{x},t) - \mathbf{x}}{1-\alpha_t}\right) \beta_t \,\mathrm{d}t + \sqrt{\beta_t}\,\mathbf{R}\,\mathrm{d}\bar{\mathbf{w}}. \end{aligned}$$

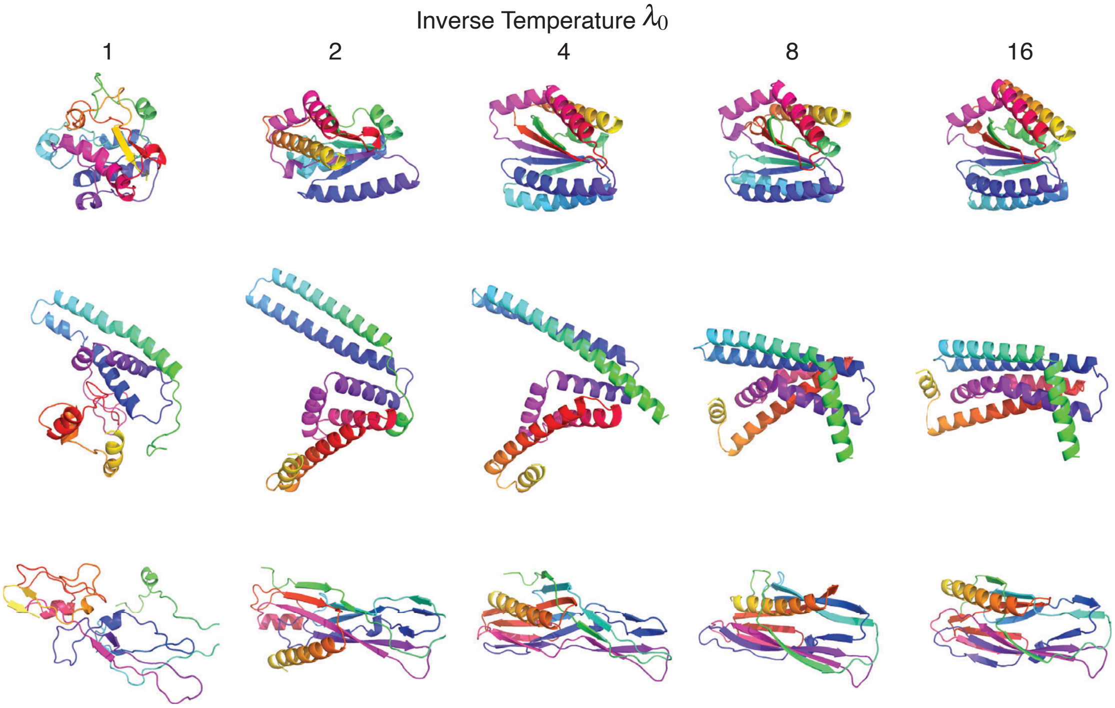
Temperature adjusted probability flow ODE Similarly for the Probability Flow ODE we can rescale as

$$\begin{aligned} \frac{d\mathbf{x}}{dt} &= -\frac{\beta_t}{2}\left(\mathbf{x} + \lambda_t \mathbf{R}\mathbf{R}^\intercal \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right) \\ &= \frac{\beta_t}{2}\left(\mathbf{x}\frac{\alpha_t + \lambda_t - 1}{1-\alpha_t} - \hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{x},t)\frac{\lambda_t\sqrt{\alpha_t}}{1-\alpha_t}\right). \end{aligned}$$

Rescaling does not reweight We derived the above rescaling rationale by considering a unimodal Gaussian, which has the simple property that the score of the perturbed diffusion can be expressed as a rescaling of the learned diffusion. This will not necessarily be true in general, and sure enough we find that the above dynamics *do drive towards local maxima* but *do not reweight populations based on their relative probability* (Supplementary Figure 1) as true low temperature sampling does. To address this, we next introduce an equilibration process that can be arbitrarily mixed in with the non-equilibrium reverse dynamics.

B.2 Annealed Langevin Dynamics SDE

Instead of reversing the forwards time diffusion in a non-equilibrium manner, we can also use the learned time-dependent score function $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ (expressed in terms of the optimal denoiser $\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{x},t)$) to do slow, approximately equilibrated sampling with annealed Langevin dynamics [Song and Ermon, 2019].



Supplementary Figure 2: **Low-temperature sampling trades reduced sample diversity (entropy) for increased sample quality (likelihood).** Given a fixed random seed for generation (each row), structures sampled at higher inverse temperature λ_0 leads have higher secondary structure content and tighter packing as compact, globular folds.

While the annealed Langevin dynamics of [Song and Ermon, 2019] was originally framed via discrete iteration, we can recast it in continuous time with the SDE

$$\begin{aligned} \mathrm{d}\mathbf{x} &= -\frac{\beta_t \psi}{2} \mathbf{R}\mathbf{R}^\intercal \nabla_{\mathbf{x}} \log p_t(\mathbf{x})^{\lambda_0}\, \mathrm{d}t + \sqrt{\beta_t \psi}\, \mathbf{R}\, \mathrm{d}\bar{\mathbf{w}} \\ &= -\frac{\beta_t \psi}{2} \lambda_0 \mathbf{R}\mathbf{R}^\intercal \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\, \mathrm{d}t + \sqrt{\beta_t \psi}\, \mathbf{R}\, \mathrm{d}\bar{\mathbf{w}} \end{aligned}$$

where ψ is an “equilibration rate” scaling the amount of Langevin dynamics per unit time. As $\psi \to \infty$ the system will instantaneously equilibrate in time, constantly adjusting to the changing score function. In practice, we can think about how to set these parameters by considering a single Euler-Maruyama integration step in reverse time with step size $\frac{1}{T}$ where T is the total number of steps

$$\mathbf{x}_{t-\frac{1}{T}} \leftarrow \mathbf{x}_t + \frac{\beta_t \psi}{2T} \lambda_0 \mathbf{R}\mathbf{R}^\intercal \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) + \sqrt{\frac{\beta_t \psi}{T}}\, \mathbf{R}\, \boldsymbol{\epsilon} \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

which is precisely preconditioned Langevin dynamics with step size $\frac{\beta_t \psi}{T}$. For a sufficiently small interval $(t - \mathrm{d}t, t)$ we can keep the target density approximately fixed while increasing T to do an arbitrarily large number of Langevin dynamics steps, which will asymptotically equilibrate to the current density $\log p_t(\mathbf{x})$.

B.3 Hybrid Langevin-Reverse Time SDE

We can combine the annealed Reverse-Time SDE and the Langevin Dynamics SDE into a hybrid SDE that infinitesimally combines both dynamics. Denoting the inverse temperature as λ_0 and the ratio of the Langevin dynamics to conventional dynamics as ψ, we have

$$\begin{aligned}
\mathrm{d}\mathbf{x} &= \left(-\frac{1}{2}\,\mathbf{x} - \left(\lambda_t + \frac{\lambda_0 \psi}{2}\right)\mathbf{R}\mathbf{R}^\intercal \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right)\beta_t \,\mathrm{d}t + \sqrt{\beta_t (1+\psi)}\,\mathbf{R}\,\mathrm{d}\bar{\mathbf{w}} \\
&= \left(-\frac{1}{2}\,\mathbf{x} - \left(\lambda_t + \frac{\lambda_0 \psi}{2}\right)\mathbf{R}\mathbf{R}^\intercal \frac{(\mathbf{R}\mathbf{R}^\intercal)^{-1}}{1-\alpha_t}\left(\sqrt{\alpha_t}\hat{\mathbf{x}}_\theta(\mathbf{x},t) - \mathbf{x}\right)\right)\beta_t \,\mathrm{d}t + \sqrt{\beta_t (1+\psi)}\,\mathbf{R}\,\mathrm{d}\bar{\mathbf{w}} \\
&= \left(-\frac{1}{2}\,\mathbf{x} - \left(\lambda_t + \frac{\lambda_0 \psi}{2}\right)\frac{\sqrt{\alpha_t}\hat{\mathbf{x}}_\theta(\mathbf{x},t) - \mathbf{x}}{1-\alpha_t}\right)\beta_t \,\mathrm{d}t + \sqrt{\beta_t (1+\psi)}\,\mathbf{R}\,\mathrm{d}\bar{\mathbf{w}}
\end{aligned}$$

where we highlight in pink the terms that, when set to unity, recover the standard reverse time SDE.

Representative samples using this modified SDE are shown in Fig 7. Without the low temperature modification, this idea is very reminiscent of the Predictor Corrector sampler proposed by Song et al. [2021], except in that case the authors explicitly alternated between reverse-time diffusion and Langevin dynamics while we fuse them into a single SDE.

Equilibration is not free Generally speaking, as we increase the amount of Langevin equilibration with ψ, we will need to simultaneously increase the resolution of our SDE solution to maintain the same level of accuracy. That said, we found that even a modest amount of equilibration was sufficient to significantly improve sample quality in practice with $\psi \in [1,5]$.

Even more equilibration Lastly, while the Hybrid Langevin-Reverse Time SDE can do an arbitrarily large amount of Langevin dynamics per time interval which would equilibrate asymptotically in principle, these dynamics will still inefficiently mix between basins of attraction in the energy landscape when $0 < t \ll 1$. We suspect that ideas from variable-temperature sampling methods, such as simulated tempering [Marinari and Parisi, 1992] or parallel tempering [Hansmann, 1997], would be useful in this context and would amount to deriving an augmented SDE system with auxiliary variables for the temperature and/or copies of the system at different time points in the diffusion.

C Polymer-Structured Diffusions

Most prior applications of diffusion models to images and molecules have leveraged uncorrelated diffusion in which data are gradually transformed by isotropic Gaussian noise. We found this approach to be non-ideal for protein structure applications for two reasons. First, noised samples break simple chain and density (e.g., radius of gyration, R_g) constraints that almost all structures satisfy [Hong and Lei, 2009, Tanner, 2016], forcing the model to allocate a significant amount of capacity towards re-implementing these basic constraints. And second, this "out-of-distribution" aspect of high-noise samples tends to limit the performance of efficient domain-specific neural architectures for molecular systems. To this end, we introduce multivariate Gaussian distributions for protein structures that (i) are $SO(3)$ invariant, (ii) enforce protein chain and radius of gyration statistics, and (iii) can be computed in linear time. Throughout this section, we will introduce covariance models for protein polymers (which can be thought of as a de-whitening transform $\mathbf{R}$, see Appendix A) with parameters that can be fit *offline* from training the diffusion model.

C.1 Preliminaries: Diffusion models linearly interpolate between the average quadratic forms of the data and prior

Here we show how the diffusion processes described above will predictably affect molecular geometry as a function of the covariance structure of the noising process. We will use this result to reflect on how the covariance structure should be designed. Squared distance D_{ij}^2 and the squared radius of gyration R_g^2 are both functions that can be expressed as quadratic forms in the coordinates. That means they can be expressed as a function $\mathcal{F}(\mathbf{x}) = \mathbf{x}^\intercal \mathbf{A}\mathbf{x}$ where $\mathbf{A}$ is a matrix weighting the different cross-terms as $\mathcal{F}(\mathbf{x}) = \sum_{i,j} A_{ij} x_i x_j$. Suppose we want to understand the behavior of these quantities as they evolve under the forward process of a diffusion model. Recall that we can write samples from the forward diffusion process as

$$\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\mathbf{R}\mathbf{z}, \qquad \mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$$

So we can write the time-expectation of any quadratic form as

$$\begin{aligned}
\mathbb{E}_{p(\mathbf{x}_t|\mathbf{x}_0)}\left[\mathcal{F}(\mathbf{x})\right] &= \mathbb{E}_{\mathbf{z}}\left[(\sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\mathbf{R}\mathbf{z})^\intercal \mathbf{A}(\sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\mathbf{R}\mathbf{z})\right] \\
&= \mathcal{F}(\sqrt{\alpha_t}\mathbf{x}_0) + \mathbb{E}_{\mathbf{z}}\left[\mathcal{F}(\sqrt{1-\alpha_t}\mathbf{R}\mathbf{z}) + \sqrt{\alpha_t(1-\alpha_t)}\left(\mathbf{x}_0^\intercal \mathbf{R}\mathbf{z} + \mathbf{R}\mathbf{z}^\intercal \mathbf{x}_0\right)\right] \\
&= \alpha_t\ \mathcal{F}(\mathbf{x}_0) + \mathbb{E}_{\mathbf{z}}\left[\mathcal{F}(\sqrt{1-\alpha_t}\mathbf{R}\mathbf{z})\right] \\
&= \alpha_t\ \mathcal{F}(\mathbf{x}_0) + (1-\alpha_t)\ \mathbb{E}_{p_{\text{model}}(\mathbf{x})}\left[\mathcal{F}(\mathbf{x})\right]
\end{aligned}$$

Squared distance is a quadratic form, so diffusion processes will simply linearly interpolate to the behavior of the prior as

$$\mathbb{E}_{p(\mathbf{x}_t|\mathbf{x}_0)}\left[D_{ij}^2(\mathbf{x}_t)\right] = \alpha_t\ D_{ij}^2(\mathbf{x}_0) + (1-\alpha_t)\ \mathbb{E}_{p_{\text{prior}}(\mathbf{x})}\left[D_{ij}^2(\mathbf{x})\right]$$

and squared radius of gyration will similarly evolve under the diffusion as

$$\mathbb{E}_{p(\mathbf{x}_t|\mathbf{x}_0)}\left[R_g^2(\mathbf{x}_t)\right] = \alpha_t\ R_g^2(\mathbf{x}_0) + (1-\alpha_t)\ \mathbb{E}_{p_{\text{prior}}(\mathbf{x})}\left[R_g^2(\mathbf{x})\right]$$

Punchline Because diffusion models will do simple linear interpolations between the average squared distances and R_g of the data distribution and of the prior, we should focus on covariance structures that empirically match these properties as closely as possible. Two primary ways will be in the chain constraint, i.e., that $D_{i,i+1}(\mathbf{x}_t)$ should always be small and match the data distribution, and the size/density constraint of how $R_g^2(\mathbf{x}_t)$ should behave as a function of protein length and typical packing statistics.

C.2 Covariance model #1: Ideal chain

In this section, we introduce one of the simplest covariance models that enforces the chain constraint but ignores the R_g scaling. It will interpolate between the data distribution and the ensemble of unfolded random coils.

Noise process We index our amount of noise with a diffusion time $t \in [0,1]$. Given a denoised structure $\mathbf{x}_0$, a level of noise t, and a noise schedule α_t, we sample perturbed structures from a Multivariate Gaussian distribution $p(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\sqrt{\alpha_t}\mathbf{x}_0, (1-\alpha_t)\Sigma)$ as

$$\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\mathbf{R}\mathbf{z}, \quad \mathbf{z} \sim \mathcal{N}(0,\mathbf{I}),$$

where the covariance matrix enforcing our chain constraint $\Sigma = \mathbf{R}\mathbf{R}^\intercal$ can be expressed in terms of its square root $\mathbf{R}$, which is defined below.

Key to our framework is a matrix $\mathbf{R}$ whose various products, inverse-products, and transpose-products with vectors can be computed in *linear* time. We define the matrix $\mathbf{R}$ in terms of its product with a vector $f(\mathbf{z}) = \mathbf{R}\mathbf{z}$:

$$f(\mathbf{z})_i = \tilde{x}_i + \delta\tilde{x}_1 - \sum_k \frac{\tilde{x}_k}{N}, \quad \text{where} \quad \tilde{x}_i = \gamma\sum_{k=1}^{i} z_k$$

The inverse product $f^{-1}(\mathbf{x}) = \mathbf{R}^{-1}\mathbf{x}$ is then

$$f^{-1}(\mathbf{x})_i = \frac{\tilde{x}_i - \tilde{x}_{i-1}}{\gamma}, \quad \text{where} \quad \tilde{x}_i = x_i - x_1 + \frac{1}{\delta}\sum_k \frac{x_k}{N}$$

This definition of $\mathbf{R}$ induces the following inverse covariance matrix on the noise, which possesses a special structure:

$$\Sigma^{-1} = (\mathbf{R}\mathbf{R}^\intercal)^{-1} = \frac{1}{\gamma^2}\begin{bmatrix} 1 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 1 \end{bmatrix} + \frac{1}{(N\gamma\delta)^2}\mathbf{1}\mathbf{1}^\intercal$$

The parameter γ sets the length scale of the chain and the parameter δ sets the allowed amount of translational noise about the origin. This latter parameter is important for training on complexes where each chain may not have a center of mass at 0.

C.2.1 Covariance model #1 has ideal chain scaling $R_g \propto N^{(1/2)}$

Our ideal-chain model is a simple Brownian motion and so the interatomic residual is Gaussian distributed with zero mean and $\gamma^2|i-j|$ variance, i.e.,

$$\mathbf{r}_{ij} \sim \mathcal{N}(0, \gamma^2|i-j|)$$

The expected squared norm for a Multivariate Normal Distribution (MVN) with spherical covariance is $\|\boldsymbol{\mu}\|_2^2 + k\sigma^2$ where k is the dimensionality, so we have:

$$\mathbb{E}_{p(\mathbf{x}_t|\mathbf{x}_0)}\left[D_{ij}^2(\mathbf{x}_t)\right] = \alpha_t\, D_{ij}^2(\mathbf{x}_0) + (1-\alpha_t)3\gamma^2|i-j|$$

When $\alpha_t = 0$, the expected squared distances are those of the data distribution, while when $\alpha_t = 1$, they are those on an ideal Gaussian chain.

To compute the expected radius of Gyration, we can use the identity that it is simply half of the root mean square of inter-residue distances

$$\begin{aligned}
\frac{1}{2N^2}\sum_{i,j}\mathbb{E}_{p_{\text{prior}}}\left[\|\mathbf{x}_t^j - \mathbf{x}_t^i\|_2^2\right] &= +\frac{1}{2N^2}\sum_{i,j}(1-\alpha)3\gamma^2|i-j| \\
&= 3\gamma^2\frac{1}{2N^2}\sum_{i,j}|i-j| \\
&= 3\gamma^2\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=i}^{N} j-i \\
&= 3\gamma^2\frac{N}{6}\left(\frac{N^2-1}{N^2}\right)
\end{aligned}$$

Therefore, we can also view the mean behavior of the diffusion as linearly interpolating the squared radius of gyration as

$$\mathbb{E}_{p(\mathbf{x}_t|\mathbf{x}_0)}\left[R_g^2(\mathbf{x})\right] = \alpha\left(R_g^{(0)}\right)^2 + (1-\alpha)3\gamma^2\frac{N}{6}\left(\frac{N^2-1}{N^2}\right)$$

When $\alpha \to 0$ and $N \ll 0$, the term $\left(\frac{N^2-1}{N^2}\right) \approx 1$ we recover the well-known scaling for an ideal chain with $\mathbb{E}_{p(\mathbf{x}_t|\mathbf{x}_0)}\left[\left(R_g^2(\mathbf{x}_t)\right)\right] = \frac{Na^2}{6}$ where the segment length is $a = \sqrt{3}\gamma$.

C.3 Covariance model #2: R_g-confined, linear-time Polymer MVNs

In this section we consider how to extend the previous model that also constrains the the scaling of the radius of gyration R_g. We consider a family of two-parameter linear chain models that include the previous model as a special case. Specifically, consider the following linear recurrence

$$\begin{aligned} x_i &= az_i + bx_{i-1} \\ &= a\sum_{j=2}^{i} b^{i-j} z_j + b^{i-1} x_1 \end{aligned}$$

Here, the parameter a is a global scale parameter setting the “segment length” of the polymer and b is a “decay” parameter which sets the memory of the chain to fluctuations. We recover a spherical Gaussian when $b = 0$ and the ideal Gaussian chain when $b = 1$.

This system can also be written in matrix form as $\mathbf{x} = \mathbf{R}\mathbf{z}$ with

$$\mathbf{R} = a \begin{bmatrix} vb^0 & & & & & \\ vb^1 & b^0 & & & & \\ vb^2 & b^1 & b^0 & & & \\ \vdots & & \ddots & \ddots & & \\ vb^{N-2} & & & b^1 & b^0 & \\ vb^{N-1} & & \cdots & b^2 & b^1 & b^0 \end{bmatrix}$$

where $v = \sqrt{\mathrm{Var}(x_1)}$.

We can solve for the equilibrium value of v via the condition $\mathrm{Var}(x_1) = a^2v^2 = \mathrm{Var}(x_i) = \mathrm{Var}(x_{i-1})$. The solution is

$$\begin{aligned} \mathrm{Var}(x_i) &= a^2\,\mathrm{Var}(z_i) + b^2\,\mathrm{Var}(x_i) \\ \mathrm{Var}(x_i)(1-b^2) &= a^2 \\ a^2v^2 &= \frac{a^2}{1-b^2} \\ v &= \frac{1}{\sqrt{1-b^2}} \end{aligned}$$

So our final recurrence is

$$x_i = a\sum_{k=2}^{i} b^{i-k} z_k + a\frac{b^{i-1}}{\sqrt{1-b^2}} z_1$$

C.3.1 Expected R_g^2 as a function of b

To compute the expected Radius of Gyration, we will use the identity $R_g^2(\mathbf{x}) = \frac{1}{2N^2}\sum_{i,j} D_{ij}^2(\mathbf{x})$, which we can compute via the variance of the residual between x_i and x_j. Assuming $j > i$, we have

$$\frac{x_j - x_i}{a} = \sum_{k=i+1}^{j} b^{j-k} z_k + \sum_{k=2}^{i} (b^{j-k} - b^{i-k}) z_k + \frac{b^{j-1} - b^{i-1}}{\sqrt{1-b^2}} z_1$$

and the variance of which is

$$\begin{aligned}\frac{1}{a^2}\mathbb{E}\left[D_{ij}^2(\mathbf{x})\right] &= \frac{1}{a^2}\operatorname{Var}(x_j - x_i)\\ &= \sum_{k=i+1}^{j} b^{2(j-k)} + \sum_{k=2}^{i}(b^{j-k}-b^{i-k})^2 + \frac{(b^{j-1}-b^{i-1})^2}{1-b^2}\\ &= \frac{2(1-b^{j-i})}{1-b^2}\end{aligned}$$

So the expected squared radius of gyration is

$$\begin{aligned}\frac{1}{a^2}\mathbb{E}\left[R_g^2(\mathbf{x})\right] &= \frac{1}{a^2}\mathbb{E}\left[\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=i}^{N} D_{ij}^2(\mathbf{x})\right]\\ &= \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=i}^{N}\frac{1}{a^2}\mathbb{E}\left[D_{ij}^2(\mathbf{x})\right]\\ &= \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=i}^{N}\frac{2(1-b^{j-i})}{1-b^2}\\ &= \frac{2b^{N+1}-b^2N(N+1)+2b(N^2-1)-N(N-1)}{(b-1)^3(b+1)N^2}\\ &\approx \left(\frac{6b}{N}+1-b^2\right)^{-1} \qquad \text{for } b \text{ on } (0,1) \text{ and } N \gg 1\\ &= \frac{N}{6b+N(1-b^2)}\end{aligned}$$

The approximation in the penultimate step works quite well in practice and becomes more accurate with growing N:

$$\forall b \in (0,1) \lim_{N\to\infty} \frac{2b^{N+1}-b^2N(N+1)+2b(N^2-1)-N(N-1)}{(b-1)^3(b+1)N^2}\left(\frac{6b}{N}+1-b^2\right) = 1$$

Additionally, we can verify that this result recovers the expected limiting behavior of an ideal unfolded chain when $b \to 1$

$$\lim_{b\to 1}\frac{1}{a^2}\mathbb{E}\left[R_g^2(\mathbf{x})\right] = \frac{N}{6}$$

and of a standard normal distribution when $b \to 0$

$$\lim_{b\to 0}\frac{1}{a^2}\mathbb{E}\left[R_g^2(\mathbf{x})\right] = 1$$

To finish up, we can add back in our global scaling factor a for our final result:

$$\begin{aligned}\mathbb{E}_{\mathbf{x}\sim\mathbf{p}_{\text{prior}}(\mathbf{x})}\left[R_g^2(\mathbf{x})\right] &= a^2\frac{2b^{N+1}-b^2N(N+1)+2b(N^2-1)-N(N-1)}{(b-1)^3(b+1)N^2}\\ &\approx \frac{Na^2}{6b+N(1-b^2)}\end{aligned}$$

C.3.2 How to implement any R_g^2 scaling

Equipped with a simple dependence of R_g^2 on b, we can solve for the correct spring strength by simply asking for what value of b we achieve our scaling law with Flory coefficient ν

$$\mathbb{E}_{\mathbf{x}\sim\mathbf{p}_{\text{prior}}(\mathbf{x})}\left[R_g^2(\mathbf{x})\right] = (rN^\nu)^2 = \frac{Na^2}{6b+N(1-b^2)}$$

This gives a quadratic equation with the solution

$$b_{\text{effective}}(N,a,r,\nu) = \frac{3}{N} \pm N^{-\nu}\sqrt{N^{2(\nu-1)}(N^2+9)-\frac{a^2}{r^2}}$$

The positive branch is the relevant one to us (the negative branch corresponds to a pathological solutions for small N), giving us the result:

$$b_{\text{effective}}(N,a,r,\nu) = \frac{3}{N} + N^{-\nu}\sqrt{N^{2(\nu-1)}(N^2+9)-\frac{a^2}{r^2}}$$

C.3.3 Standardizing the translational variance

Currently, the above procedure has diverging marginal variance as $b \to 1$. We can arbitrarily retune the *translational variance* of each chain with the following mean-deflation operation which enforces $\sum_k \frac{x_k}{N} = (1-\xi)\sum_k \frac{\tilde{x}_k}{N}$:

$$\begin{aligned}x_i &= \tilde{x}_i - \xi\sum_k \frac{\tilde{x}_k}{N}\\ \tilde{x}_i &= x_i + \frac{\xi}{1-\xi}\sum_k \frac{x_k}{N}\end{aligned}$$

C.3.4 Setting the parameters

We have two procedures for setting the value of ξ, leading to two different named covariance models:

1. **Pile-of-globs covariance**. Set a and b to satisfy within-chain R_g scaling of each chain in a complex based on its length. Set ξ so that the translational variance of each chain is unity. This will cause chains to have a realistic radius of gyration but pile up at the origin.

2. **Glob-of-globs covariance**. Set a and b to satisfy within-chain R_g scaling of each chain in a complex based on its length. Set ξ per chain by solving for the translational variance that also implements the correct whole-complex R_g scaling as a function of the number of residues. This will cause chains to preserve a realistic complex-level radius of gyration and also intra-chain radius of gyration that scales as that of individual globular proteins.

C.3.5 Factorization of the matrix

When also including a centering transform, we can factorize the matrix $\mathbf{R}$ as a product of three matrices, which can be useful for computing inverses and transposes:

$$
\begin{aligned}
\mathbf{R} &= a\mathbf{R}_{\text{center}}\mathbf{R}_{\text{sum}}\mathbf{R}_{\text{init}} \\
&= a\left(\mathbf{I}-\frac{\xi}{N}\mathbf{1}\mathbf{1}^{\intercal}\right)
\begin{bmatrix}
b^0 & & & & & \\
b^1 & b^0 & & & & \\
b^2 & b^1 & b^0 & & & \\
\vdots & & \ddots & \ddots & & \\
b^{N-2} & & & b^1 & b^0 & \\
b^{N-1} & & \cdots & b^2 & b^1 & b^0
\end{bmatrix}
\begin{bmatrix}
\frac{1}{\sqrt{1-b^2}} & & & & & \\
 & 1 & & & & \\
 & & 1 & & & \\
 & & & \ddots & & \\
 & & & & 1 & \\
 & & & & & 1
\end{bmatrix}
\end{aligned}
$$

C.4 Inverse covariance and intuition

Ignoring the translational rescaling and by numerical investigation, it appears that the precision matrix $\Sigma^{-1} = (\mathbf{R}\mathbf{R}^{\intercal})^{-1}$ is

$$
\boldsymbol{\Sigma}^{-1} = \frac{1}{a^2}
\begin{bmatrix}
1 & -b & & & & \\
-b & 1+b^2 & -b & & & \\
 & -b & 1+b^2 & -b & & \\
 & & \ddots & \ddots & \ddots & \\
 & & & -b & 1+b^2 & -b \\
 & & & & -b & 1
\end{bmatrix}
$$

We can decompose this as the sum of terms for the precision of Brownian motion (Chain Laplacian), for a spherical Gaussian (Identity matrix), and some nuisance boundary conditions

$$
\boldsymbol{\Sigma}^{-1} = \frac{1}{a^2}\left(b
\begin{bmatrix}
1 & -1 & & & & \\
-1 & 2 & -1 & & & \\
 & -1 & 2 & -1 & & \\
 & & \ddots & \ddots & \ddots & \\
 & & & -1 & 2 & -1 \\
 & & & & -1 & 1
\end{bmatrix}
+(1-b)^2\mathbf{I}+
\begin{bmatrix}
b(1-b) & & \\
 & & \\
 & & \\
 & & b(1-b)
\end{bmatrix}
\right)
$$

D Random Graph Neural Networks

Prior approaches to predicting or generating protein structure have relied on neural network architectures with $\mathcal{O}(N^2)$ or $\mathcal{O}(N^3)$ computational complexity [Jumper et al., 2021, Anand and Achim, 2022, Trippe et al., 2022], in part motivated by the need to process the structure at multiple length scales simultaneously and/or to reason over triples of particles as is done during distance geometry methods. Here we introduce an effective alternative to these approaches with sub-quadratic complexity by combining Message Passing Neural Network [Gilmer et al., 2017] layers with random graph generation processes. We design random graph sampling methods that reproduce the connectivity statistics of efficient N-body simulation methods, such as the Barnes-Hut algorithm [Barnes and Hut, 1986].

D.1 Background: efficient N-body simulation

One of the principal lessons of computational physics is that N-body simulations involving $\mathcal{O}(N^2)$ dense interactions (e.g. gravitational simulations and molecular physics) can often be effectively simulated with only $\mathcal{O}(N \log N)$-scaling computation. Methods such as Barnes-Hut [Barnes and Hut, 1986] and the Fast Multipole Method take advantage of a common particular property of (and inductive bias for) physical systems that *more distant interactions can be modeled more coarsely for the same level of accuracy*. For example, in cosmological simulations, you can approximate the gravitational forces acting on a star in a distant galaxy by approximating that galaxy as a point at its center of mass.

So far, most *relational* machine learning systems [Battaglia et al., 2018] for protein structure have tended to process information in a manner that is either based on local connectivity (e.g. a k-Nearest Neighbors or cutoff graphs) [Ingraham et al., 2019] or all-vs-all connectivity [Jumper et al., 2021, Anand and Achim, 2022, Trippe et al., 2022]. The former approach is natural for highly spatially localized tasks such as structure-conditioned sequence design and the characterization of residue environments, but it is less clear if local graph-based methods can effectively reason over global structure in a way that is possible with fully connected Graph Neural Networks, such as Transformers [Vaswani et al., 2017]. Here we ask if there might be reasonable ways to add in long-range reasoning while preserving sub-quadratic scaling simply by random graph construction.

Related work Our method evokes similarity to approaches that have been used to scale Transformers to large documents by combining a mixture of local and deterministically [Child et al., 2019] or randomly sampled long-range context [Zaheer et al., 2020]. Distant-dependent density of context has also been explored in multiresolution attention for Vision transformers [Yang et al., 2021] and in dilated convolutional neural networks [van den Oord et al., 2016].

D.2 Random graph generation

We propose to build scalable graph neural networks for molecular systems by sampling random graphs that mix short and long-range connections. We define the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the node set and $\mathcal{E}$ is the edge set. A protein can be represented as a point set $\mathbf{x} \in \mathbb{R}^{N \times 3}$. We define

Algorithm 1 Random graph generation

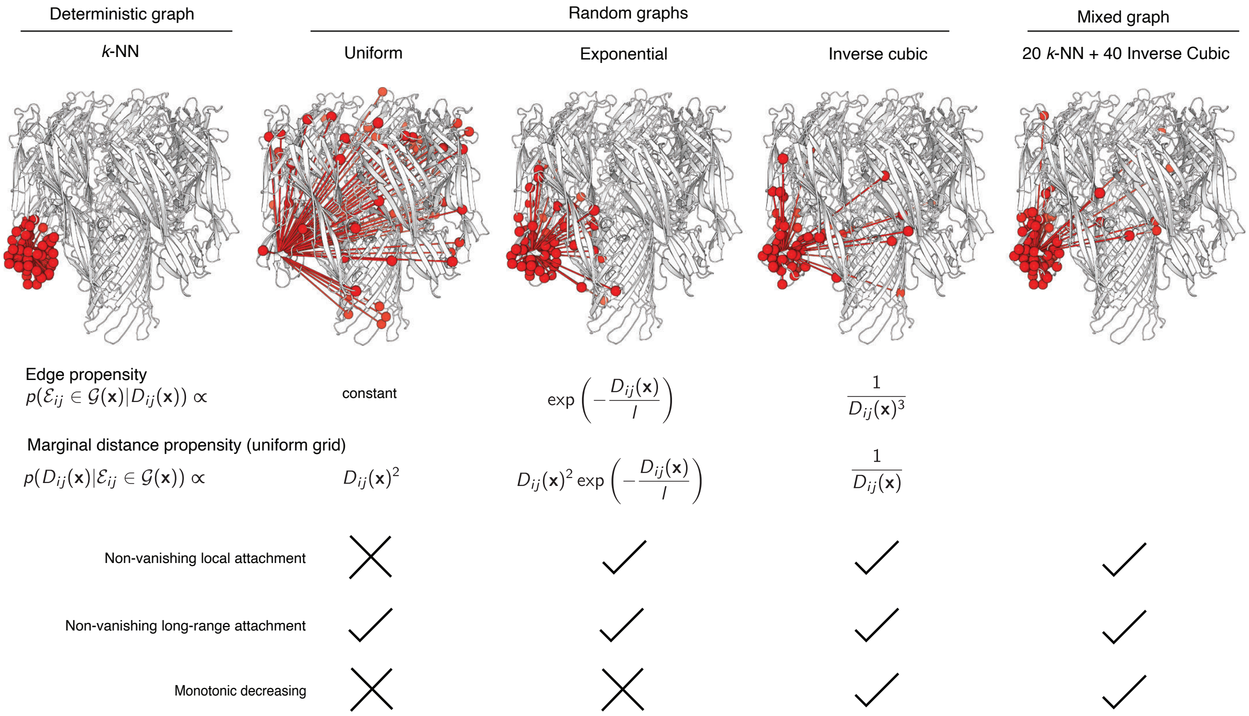
```
Require: Inter-node distances $\{D_{ij}\}_{i,j=1}^{N}$, inverse temperature $\zeta$, attachment propensity
  $\log p((i,j) \in \mathcal{E}(\mathbf{x})|D_{ij}) \propto e^{c(D_{ij})}$, number of edges to sample $k$
  for each $i \in [N]$ do
      for each $j \in [N]$ do
          $U_{ij} \sim \text{Uniform}(0,1)$                         ▷ Sample uniform noise per edge
          $Z_{ij} \leftarrow \zeta\, c(D_{ij}) - \log\big(-\log\big(U_{ij}\big)\big)$   ▷ Perturb log probabilities with Gumbel noise
      end for
      $\mathcal{E} \leftarrow \bigcup_i^N \{(i,j) | j \in \text{TopK}(\mathbf{Z}_i)\}$   ▷ build edge set by sampling top $k$ edges
  end for
```

the process of constructing the geometric graph as $\mathcal{G}(\mathbf{x}) = (\mathcal{V}, \mathcal{E}(\mathbf{x}))$ with $|\mathcal{V}| = N$. Different from the usual graph construction scheme, the edges are generated stochastically, and $\mathcal{E}(\mathbf{x})$ describes the process. We consider schemes in which edges for each node are sampled without replacement from the set of possible edges, weighted by an edge propensity function based on spatial distance (Fig. 3). In practice, we implement this weighted sampling without replacement using Gumbel Top-k sampling Kool et al. [2019] (Algorithm 1). Throughout this work, we use hybrid graphs which include the 20 nearest neighbors per node together with 40 randomly sampled edges under the inverse cubic edge propensity function so that both short-range and long-range interactions are sampled with appropriate rates.

D.3 Computational complexity

Under the inverse cubic attachment model, the cumulative edge propensity as a function of distance will scale as $\int_{D_{\min}}^{D_{\max}} \frac{1}{r^3} r^2 dr = \int_{D_{\min}}^{D_{\max}} \frac{1}{r} dr = \log D_{\max} - \log D_{\min}$. As we increase the total size (radius) of the system by $D_{\max}$, we only need to increase the total number of of edges per node by a factor of $\log D_{\max}$ to keep up with the increase in total edge propensity (and to therefore ensure that increasingly distant parts of the system do not "steal" edge mass from closer parts of the system). This means that, even if we were to scale to extremely large systems such as large, solvated molecular dynamics systems with millions of atoms, the total amount of computation required for a system of N atoms will scale as $\mathcal{O}(N \log N)$. In practice, we found that for protein sizes considered in this work (complexes containing up to 4000 residues[2]) it was sufficient to simply set the number of edges per node to a constant $k = 60$, which means that the graph and associated computation will scale within this bounded size as $\mathcal{O}(N)$. This is a considerable improvement on previous approaches for global learning on protein structure such as methods based on fully connected graph neural networks Trippe et al. [2022] $\mathcal{O}(N^2)$ or Evoformer-based approaches [Jumper et al., 2021] which scale as $\mathcal{O}(N^3)$. These sparse graphs also combine favorably with our method for synthesizing updated protein structures based on predicted inter-residue geometries (Section E).

[2] In some of our symmetry examples we find that models still generalize well to systems larger than they were trained on.


Deterministic graph
Random graphs
Mixed graph
k-NN
Uniform
Exponential
Inverse cubic
20 k-NN + 40 Inverse Cubic
Edge propensity
$p(\mathcal{E}_{ij} \in \mathcal{G}(\mathbf{x}) | D_{ij}(\mathbf{x})) \propto$
constant
$\exp\left(-\frac{D_{ij}(\mathbf{x})}{l}\right)$
$\frac{1}{D_{ij}(\mathbf{x})^3}$
Marginal distance propensity (uniform grid)
$p(D_{ij}(\mathbf{x}) | \mathcal{E}_{ij} \in \mathcal{G}(\mathbf{x})) \propto$
$D_{ij}(\mathbf{x})^2$
$D_{ij}(\mathbf{x})^2 \exp\left(-\frac{D_{ij}(\mathbf{x})}{l}\right)$
$\frac{1}{D_{ij}(\mathbf{x})}$
Non-vanishing local attachment ✗ ✓ ✓ ✓
Non-vanishing long-range attachment ✓ ✓ ✓ ✓
Monotonic decreasing ✗ ✗ ✓ ✓

Supplementary Figure 3: **Random graphs with distance-weighted attachment efficiently capture long-range context.** Contemporary graph neural networks for learning from molecular systems achieve efficiency via spatial locality, e.g. with a spatial k-Nearest Neighbors graphs or cutoff graph (top left, $\mathcal{O}(Nk)$). We propose methods that retain this efficiency while incorporating long-range context through random edge sampling weighted by spatial distance (middle columns). We consider three different graph sampling schemes: (i) Uniformly random sampling (middle left) introduces long-range context but at the expense of vanishing local attachment. (ii) Exponential distance weighting (middle center), which can be related to dilated convolutions [van den Oord et al., 2016], includes both short- and long-range attachment but introduces a *typical* length scale as it induces Gamma-distributed distances. (iii) Inverse cubic distance weighting (middle right), which is the effective connectivity scaling of fast N-body methods such as Barnes-Hut [Barnes and Hut, 1986], retains a balance of both short and long-term distances with a marginal distance propensity that gently and monotonically decays with D. In practice, we combine inverse cubic sampled random graphs with deterministic k-NN graphs to guarantee coverage of the k closest nodes while adding in long-range context (top right).

E Equivariant Consensus Structure from Weighted Inter-residue Geometries

E.1 Background and motivation

Prior neural network layers for generating molecular geometries in proteins have typically relied on either (i) direct prediction of backbone internal coordinates (i.e., dihedral angles) [AlQuraishi, 2019, Wu et al., 2022a], which incurs accumulating errors along the chain in the form of "lever effects" that hinder performance beyond small systems; (ii) prediction of inter-residue geometries followed by offline optimization [Anand and Huang, 2018, Senior et al., 2020], which builds on the successes of predicting protein structure from contacts [Marks et al., 2012] but is difficult to make end-to-end trainable; or (iii) iterative local coordinate updates based on the entire molecular system [Jumper et al., 2021].

In principle, protein structures arise from a balance of competing intra-molecular forces between atoms in the polymer. Indeed, protein structure can be thought of as the solution to a constraint satisfaction problem across multiple length scales and with many kinds of competing interactions. It is therefore natural to think about protein structure prediction as a so-called "Structured Prediction" problem [Belanger and McCallum, 2016] from machine learning, in which predictions are cast as the low-energy configurations of a learned potential function. Structured Prediction models often learn efficiently because it is usually simpler to express a system in terms of its constraints than to directly characterize the solutions to these constraints. This perspective can be leveraged literally for molecular geometries via differentiable optimization or differentiable molecular dynamics [Ingraham et al., 2018, Schoenholz and Cubuk, 2020, Wang et al., 2020], but these approaches are often unstable and can be cumbersome to integrate as part of a larger learning system.

E.2 Equivariant structure updates via convex optimization

Here we introduce a novel framework which realizes the benefits of inter-residue geometry prediction and end-to-end differentiable optimization in an efficient form based on convex optimization. We show how predicting pairwise inter-residue geometries as pairwise roto-translation transformations with anisotropic uncertainty induces a convex optimization problem which can be either locally solved analytically admits a dast iteration scheme for a global consensus configuration. Throughout this section we will refer to the coordinate frames of residues with a notation that is similar to that used in AlphaFold2 [Jumper et al., 2021], but with rotations $\mathbf{R}$ replaced with $\mathbf{O}$ (for **O**rientation) as in Ingraham et al. [2019]. The functions for synthesizing the backbone structure from residue poses (i.e., StructureToTransforms and TransformsToStructure) are those described in the supplementary information of [Jumper et al., 2021].

The key idea of our update is that we ask the network to predict a set of inter-residue geometries $\mathbf{T}_{ij}$ together with confidences w_{ij} (which will initially be simple but can be extended to anisotropic uncertainty) and we then attempt to either fully or approximately solve for the consensus structure that best satisfies this set of pairwise predictions.

Transform preliminaries Let $\mathbf{T} = (\mathbf{t}, \mathbf{O}) \in \mathrm{SE}(3)$ be a transformation consisting of a translation by vector $\mathbf{t} \in \mathbb{R}^3$ followed by a rotation by an orthogonal matrix $\mathbf{O} \in \mathrm{SO}(3)$. These transformations form a group with identity, inverse, and composition given by

$$\begin{aligned}
\mathbf{T}_{id} &= (\mathbf{0},\, \mathrm{I}) \\
\mathbf{T}^{-1} &= \left(-\mathbf{O}^{-1}\mathbf{t},\, \mathbf{O}^{-1}\right) \\
\mathbf{T}_a \circ \mathbf{T}_b &= (\mathbf{t}_a,\, \mathbf{O}_a) \circ (\mathbf{t}_b,\, \mathbf{O}_b) \quad = (\mathbf{t}_a + \mathbf{O}_a \mathbf{t}_b,\, \mathbf{O}_a \mathbf{O}_b)
\end{aligned}$$

We denote the transformation to the frame of each residue a as $\mathbf{T}_a$, and denote the relative transformation from residue a to residue b as

$$\mathbf{T}_{ab} := \mathbf{T}_a^{-1} \circ \mathbf{T}_b \ = \left(\mathbf{O}_a^{-1} \left(\mathbf{t}_b - \mathbf{t}_a \right), \ \mathbf{O}_a^{-1} \mathbf{O}_b \right)$$

These relative transformations satisfy equations

$$\begin{aligned} \mathbf{T}_{ab} \circ \mathbf{T}_{bc} &= \mathbf{T}_{ac} \\ \mathbf{T}_{ba} &= \mathbf{T}_{ab}^{-1} \end{aligned}$$

Convex problem Given a collection of pairwise inter-residue geometry predictions $\mathbf{T}_{ij}$ and confidences $w_{ij} \in \mathbb{R}$, we score a candidate structure $\{\mathbf{T}_i\}_{i=1}^N = \mathrm{StructureToTransforms}(\mathbf{x})$ via a weighted loss U that measures the agreement between the current pose of each residue $\mathbf{T}_i$ and the predicted pose of the residue from the frame of each neighbor $\mathbf{T}_j$:

$$\begin{aligned} U\left(\{\mathbf{T}_i\}; \{w_{ij}, \mathbf{T}_{ij}\}\right) &= \sum_{i,j} w_{ij} \left| \mathbf{T}_i - \mathbf{T}_j \circ \mathbf{T}_{ji} \right|^2 \\ &= \sum_{i,j} w_{ij} \left| \mathbf{t}_i - (\mathbf{t}_j + \mathbf{O}_j \mathbf{t}_{ji}) \right|^2 + w_{ij} \left| \mathbf{O}_i - \mathbf{O}_j \mathbf{O}_{ji} \right|^2 \end{aligned}$$

We wish to optimize each local pose $\mathbf{T}_i$ with neighbors fixed as

$$\mathbf{T}_i^\star \leftarrow \arg\min_{\mathbf{T}_i} U\left(\{\mathbf{T}_i\}; \{w_{ij}, \mathbf{T}_{ij}\}\right)$$

This problem of finding the local “consensus pose” $\mathbf{T}_i^\star$ is a convex optimization problem, the solution to which can be analytically realized as a weighted average with projection,

$$\mathbf{T}_i^\star = \left(\sum_j p_{ij} (\mathbf{t}_j + \mathbf{O}_j \mathbf{t}_{ji}), \ \mathrm{Proj}_{SO(3)} \left(\sum_j p_{ij} \mathbf{O}_j \mathbf{O}_{ji} \right) \right), \quad \text{where } p_{ij} = \frac{w_{ij}}{\sum_j w_{ij}}$$

where the projection operator is accomplished via SVD as in the Kabsch algorithm [Kabsch, 1976] for optimal RMSD superposition. If we iterate this update multiple times to all positions in parallel, we obtain a parallel coordinate descent algorithm. Putting this together, we parameterize our denoising function as

$$\begin{aligned} \{\mathbf{T}_i\}_{i=1}^N &\leftarrow \mathrm{StructureToTransforms}(\mathbf{x}) \\ \{w_{ij}, \mathbf{T}_{ij}\}_{ij \in \mathcal{G}(\mathbf{x})} &\leftarrow \mathrm{InvariantGraphNeuralNetwork}(\mathbf{x}, \mathcal{G}(\mathbf{x}); \boldsymbol{\theta}) \\ \{\mathbf{T}_i\}_{i=1}^N &\leftarrow \mathrm{LocallyOptimizeTransforms}\left(\{\mathbf{T}_i\}_{i=1}^N; \{w_{ij}, \mathbf{T}_{ij}\}_{(i,j) \in \mathcal{G}(\mathbf{x})}\right) \qquad \text{do this } M \text{ times} \\ \hat{\mathbf{x}} &\leftarrow \mathrm{TransformsToStructure}(\{\mathbf{T}_i\}_{i=1}^N) \end{aligned}$$

Extensions to anisotropic uncertainty models The above iteration leverages an *isotropic* uncertainty model in which the error model for the translational component is spherically symmetric and coupled to the uncertainty in the rotational component of the transform. We consider two forms of anisotropic uncertainty: in the first, two-parameter version, we decouple the weight w_{ij} into separate factors for the translational and rotational components of uncertainty, $w^{\intercal}_{ij}$ and $w^{\angle}_{ij}$, respectively. This makes intuitive sense when, for example, the network will possess high confidence about the relative position of another residue but not its orientation.

In a second and more sophisticated form of anisotropic uncertainty, we extend this framework to ellipsoidal error models bespoke to each ij, while retaining a closed form iteration update using approaches from sensor fusion. We parameterized this anisotropic error model by separating this precision term w into three components: $w^{\angle}_{ij}$ for rotational precision and two components for position: $w^{\|}_{ij}$ for radial distance precision, and $w^{\perp}_{ij}$ for lateral precision. The radial and lateral precision terms are each eigenvalues of the full 3x3 precision matrix $\mathbf{P}_{ij}$ for translation errors (i.e., inverse covariance matrix under a multivariate normal error model):

$$\mathbf{P}_{ij} = w^{\|}_{ij}\pi_{ij} + w^{\perp}_{ij}(I - \pi_{ij}), \qquad \pi_{ij} = \frac{(\mathbf{O}_j\mathbf{t}_{ji})(\mathbf{O}_j\mathbf{t}_{ji})^{\intercal}}{(\mathbf{O}_j\mathbf{t}_{ji})^{\intercal}(\mathbf{O}_j\mathbf{t}_{ji})}$$

where π_{ij} is the projection matrix onto the radial direction from $\mathbf{t}_j$ to the predicted position $\mathbf{t}_j + \mathbf{O}_j\mathbf{t}_{ji}$ of $\mathbf{t}_i$, and $\mathrm{I} - \pi_{ij}$ is the projection matrix onto lateral translations (spanned by the remaining two eigenvectors). These anisotropic terms finally combine as

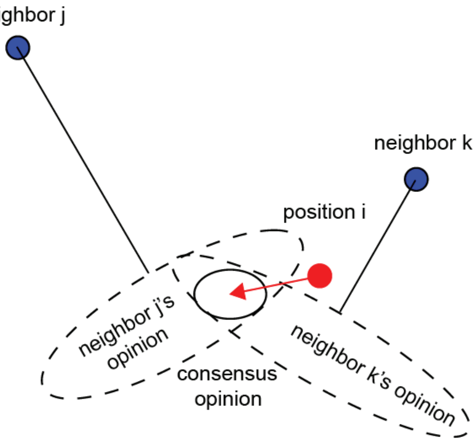
$$\begin{aligned} U\left(\{\mathbf{T}_i\};\{w_{ij},\mathbf{T}_{ij}\}\right) &= \sum_{i,j}\left(\mathbf{t}_j + \mathbf{O}_j\mathbf{t}_{ji} - \mathbf{t}_i\right)^{\intercal}\mathbf{P}_{ij}\left(\mathbf{t}_j + \mathbf{O}_j\mathbf{t}_{ji} - \mathbf{t}_i\right) + w^{\angle}_{ij}\left|\mathbf{O}_i - \mathbf{O}_j\mathbf{O}_{ji}\right|^2 \\ &= \sum_{i,j} w^{\|}_{ij}\left|\pi_{ij}(\mathbf{t}_j + \mathbf{O}_j\mathbf{t}_{ji} - \mathbf{t}_i)\right|^2 \\ &\qquad + w^{\perp}_{ij}\left|(I - \pi_{ij})(\mathbf{t}_j + \mathbf{O}_j\mathbf{t}_{ji} - \mathbf{t}_i)\right|^2 + w^{\angle}_{ij}\left|\mathbf{O}_i - \mathbf{O}_j\mathbf{O}_j\right|^2 \end{aligned}$$

As we expect the radial precision to always exceed the lateral precision, our neural predictor outputs three positive parameters ($w^{\perp}$, $w^{\|} - w^{\perp}$, $w^{\angle}$). Whereas the isotropic objective above is solved by weighted averaging, the anisotropic translation part of this objective is solved by a standard Gaussian product operation from sensor fusion [Murphy, 2007]:

$$\mathbf{t}_i^{\star} = \mathbf{t}_i + \left(\sum_j \mathbf{P}_{ij}\right)^{-1} \sum_j \mathbf{P}_{ij}(t_j + \mathbf{O}_j\mathbf{t}_{ji} - t_i)$$

Supplementary Figure 4 illustrates this anisotropic Gaussian fusion operation.

Extension as a generalization of the AlphaFold Structure Module If an additional "dummy" edge is added that is connected to the current state of $\mathbf{T}_i$, then this transform will serve an identical role to the predicted frame updates in AlphaFold2. Thus, our framework can be cast as a generalization of this family of backbone updates that opens up opportunities for complex fusion of predicted interrresidue geometries.



Supplementary Figure 4: **Anisotropic consensus update.** Position i is forced towards its consensus position which is the mean of a fusion of anisotropic Gaussians. Here we visualize the covariance ellipsese of component the Gaussians, i.e. the inverses of the precision matrices predted by our network.

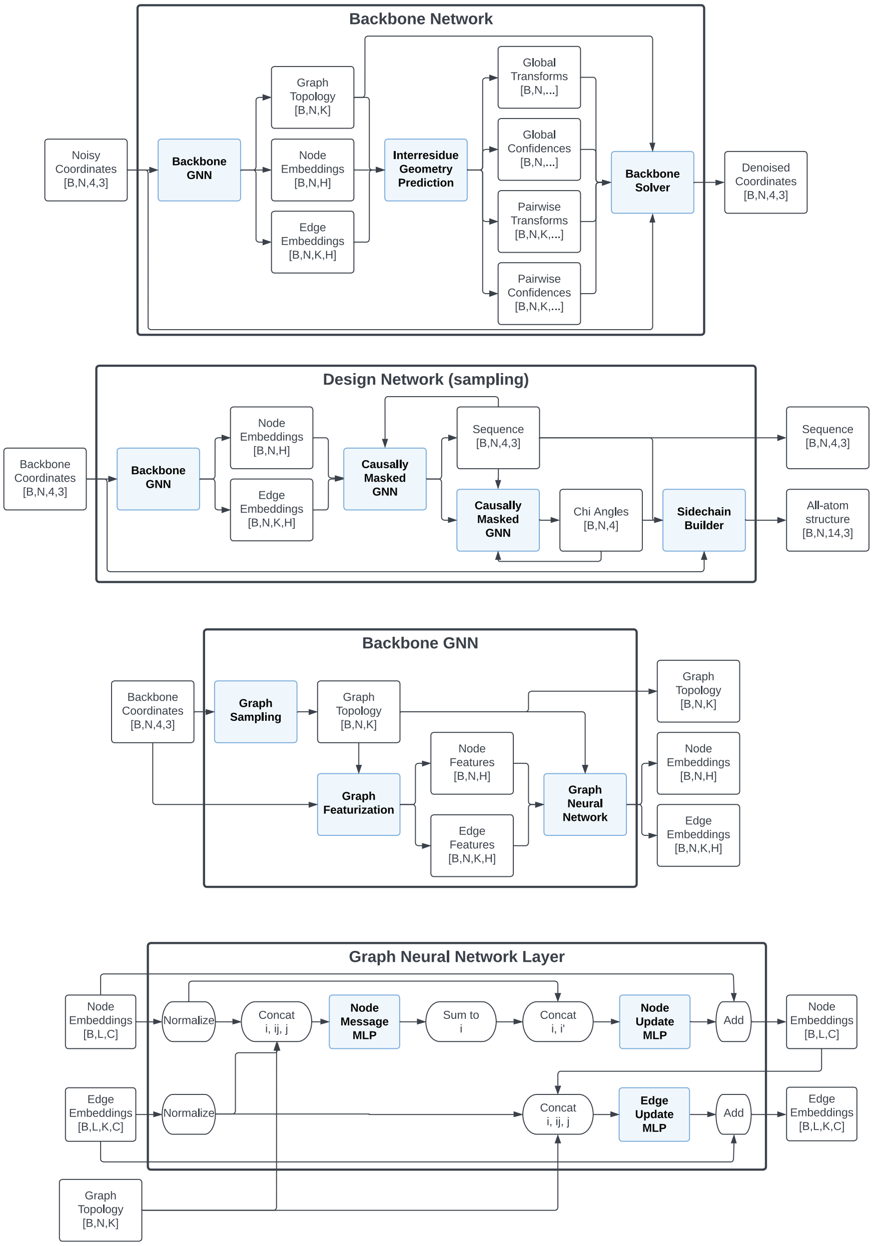
E.3 Extension to equivariant prediction of all backbone atoms

The above updates predict rigid residue poses $\{\mathbf{T}_i\}$ but our diffusion model (Appendix C) requires independent prediction of all backbone heavy atoms. We can straightforwardly augment the above predictions in an equivariant manner by predicting from every node embedding *local coordiates* for each atom position relative to the parent residue pose. To ease learning, we cast these as residual updates from the ideal geometry positions of each backbone heavy atom. To build the final atomic structure, we simply compose these right-compose these local coordinate predictions with the parent poses. These predictions will be equivariant because they are right-composed with the parent residue poses, which are equivariant because they are built from relative, equivariant geometric transformations off of the initial geometry.

F Chroma architecture

Supplementary Figure 5 provides a diagram of the Chroma architecture, which includes the *backbone network*, the *design network*, and the underlying graph neural network components. We list important hyperparameters for the backbone network in Supplementary Table 2 and for the design network in Supplementary Table 3. We design sequences by extending the framework of [Ingraham et al., 2019] and factorizing joint rotamer states autoregressively in space, and then locally autoregressively per side-chain χ angle within a residue as done in [Anand et al., 2022]. For the sequence decoder, we explore both autoregressive decoders of sequence (pictured in fig. 5) and conditional-random field decoding of sequence, which was also explored in concurrent work [Li et al., 2022].

Graph Neural Network All of our neural network models are based on graph neural networks that reason over 3D structures of proteins by transforming them into attributed graphs built from rigid transformations ($SE(3)$) invariant features. This approach has been pursued in several prior



Supplementary Figure 5: **Chroma is composed of graph neural networks for backbone denoising and sidechain design.**

works [Ingraham et al., 2019], and our primary architectural innovations on those models are two-fold:

- We propose *random graph neural networks* that add in long-range connections and reasoning while preserving subquadratic / quasi-linear computational complexity (Appendix D)
- We introduce a method for efficiently and differentiably generating protein structures from predicted inter-residue geometries based on parallel coordinate descent (Appendix E)

Category	**Hyperparameter**	**Value in Backbone Network A**	**Value in Backbone Network B**
Diffusion model	Covariance model	Pile-of-Globs	Glob-of-Globs
	Noise schedule	Log-linear SNR (-7,13.5)	Log-linear SNR (-7,13.5) [Kingma et al., 2021]
Graph Featurization	Node Features	Internal Coordinates	Internal Coordinates
	Edge Features	Distances, Interresidue transforms, $\|i-j\|$, same_chain,	Distances, Interresidue transforms, $\|i-j\|$, same_chain
	Number of edges per node, k	60	60
	Number of kNN edges	20	20
	Number of Inverse Cubic edges	40	40
Graph Neural Network	Number of GNN layers	12	12
	Node Embedding Dimension	512	512
	Edge Embedding Dimension	256	256
	Node MLP hidden dimension	512	2048
	Edge MLP hidden dimension	128	128
	Dropout p	0.1	0.1
Backbone Solver	Uncertainty model	Isotropic (1-parameter)	Decoupled (2-parameter)
	Number of iterations	3	10
Loss Function	Whitened ELBO weight	1	1
	X-space pseudo-ELBO weight	1	1
	X-space units	Nanometers	Nanometers
Total Number of Parameters		18.6M	94.1M

Table 2: **Hyperparameters of backbone network configuration.**

Category	**Hyperparameter**	**Value in Design Network A**	**Value in Design Network B**
Graph Featurization	Node Features	Internal Coordinates	Internal Coordinates
	Edge Features	Distances, $\|i-j\|$, same_chain	Distances, $\|i-j\|$, same_chain
	Number of edges per node, k	30	30
	Number of kNN edges	30	30
	Number of inverse cubic edges	0	0
Graph Neural Network	Number of GNN layers	6	3
	Node embedding dimension	128	128
	Edge embedding dimension	128	128
	Node MLP hidden dimension	512	512
	Edge MLP hidden dimension	128	128
	Dropout p	0.1	0.1
Sequence decoder	Type	Potts model	Autoregressive
Chi decoder	Number of chi bins	N/A	36
Total Number of Parameters		3.7M	9.2M

Table 3: **Hyperparameters of design network configuration.**

G Training

G.1 Dataset

The PDB was queried (on 2022-03-20) for non-membrane X-ray protein structures with a resolution of 2.6 *Å* or better. Structures with homologous sequences were removed by assigning each chain sequence to a cluster ID using USEARCH [Edgar, 2010] at a 50% sequence identity threshold and removing entries with chain cluster ID completely found in another entry. An additional set of 1726 non-redundant antibody structure cluster using 90% sequence identity was added to the reduced set. All 28819 remaining structures were transformed to their biological assembly by favouring assembly ID where the authors and software agreed, followed by authors and finally by software only. Missing side-chain atoms were added with pyRosetta [Chaudhury et al., 2010]. An 80/20/20 train, validation and test splits were generated by minimizing the sequence similarity overlap using entries of PFAM family ID, PFAM clan ID [Mistry et al., 2021], UniProt ID [Bateman et al., 2020] and MMSEQ2 cluster ID at a 30% threshold [Steinegger and Söding, 2017]. A graph pruning method was used to minimize shared label overlap between all splits. Briefly, a graph is built where each PDB entry is represented by a node connected to other entries that share at least one identical annotation. Connected sub-graphs are identified and broken apart by iteratively deleting the most central annotations until there are 50 or fewer connected nodes. Using this procedure, the generated test set had 9%, 59%, 82% and 89% of its entries that did not share any PFAM clan, PFAM family, MMSEQ30 cluster, or Uniprot IDs with the training set, respectively. Whereas a random split would have given 0.1%, 10%, 50% and 70% for the same label types, respectively.

G.2 Optimization

We train the backbone model by optimizing the regularized ELBO loss (Appendix A) with the Adam optimizer [Kingma and Ba, 2014] and leverage data parallelism across 8 GPUs. We train the design networks by optimizing the sequence (pseudo)likelihoods and chi angle likelihoods with Adam on a single GPU.

H Evaluation

H.1 Unconditional samples

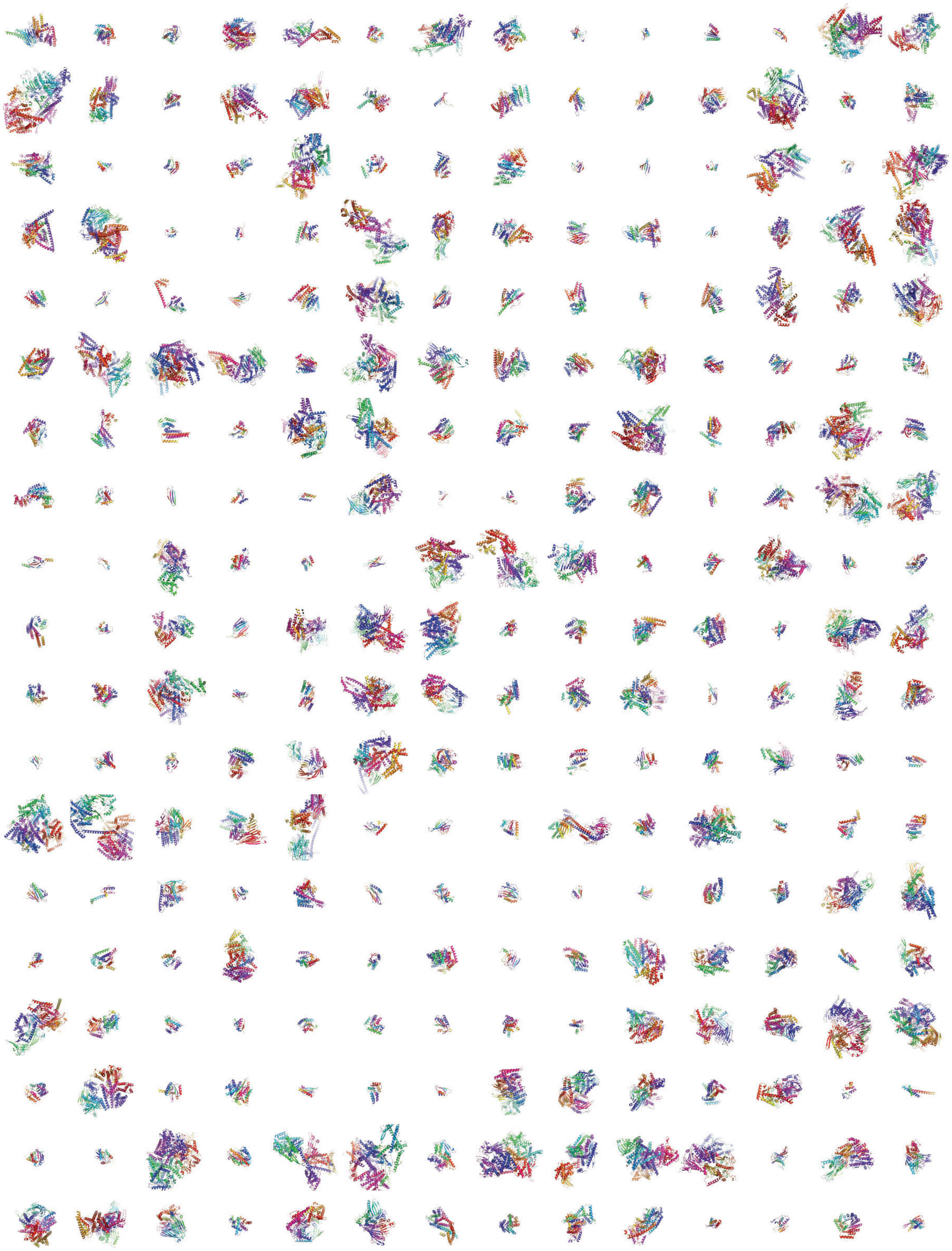
Two sets of unconditional protein samples were generated for display and analysis with Chroma. Both sets used the same parameters: 200 steps, $\lambda_0 = 10$, and $\psi = 2$. Of these two sets, one was comprised of single chain proteins, and the other, of multi-chain proteins. The single chain set contained 50 thousand samples and the lengths were drawn from a ”1/Length” distribution where the probability of a protein chain’s length was proportional to one over its length. The multi-chain set contained 10 thousand samples and length distribution of each chain was determined empirically from the chain length statistics from complexes in the PDB. Specifically, a random protein complex was drawn from the PDB and the number of chains and length of each chain for the random sample was determined from that random PDB complex. In order to show typical non-cherry picked random samples from the model we provide supplementary Figure 6 for single chain and Figure 7 for multi-chain examples.

H.2 Backbone geometry statistics

To evaluate the structural validity of Chroma generated single chain structures, they were characterized based on secondary structures and residue interactions alongside a non-redundant subset of PDB database (**Table 4**). The distribution of secondary structures (α-helix, β-strands, and coil) was evaluated using Stride [Frishman and Argos, 1995]. Residue interaction was determined by any pairwise residue (C-α to C-α) with an Euclidean distance less than 8 $\AA$. Mean and long-range residue contact were computed. Contact order [Ivankov et al., 2003] and radius of gyration [Tanner, 2016] were computed and length normalized according to their corresponding empirical power laws. All metrics except for secondary structures are normalized for Fig. 2b.

H.3 Novelty and structural homology

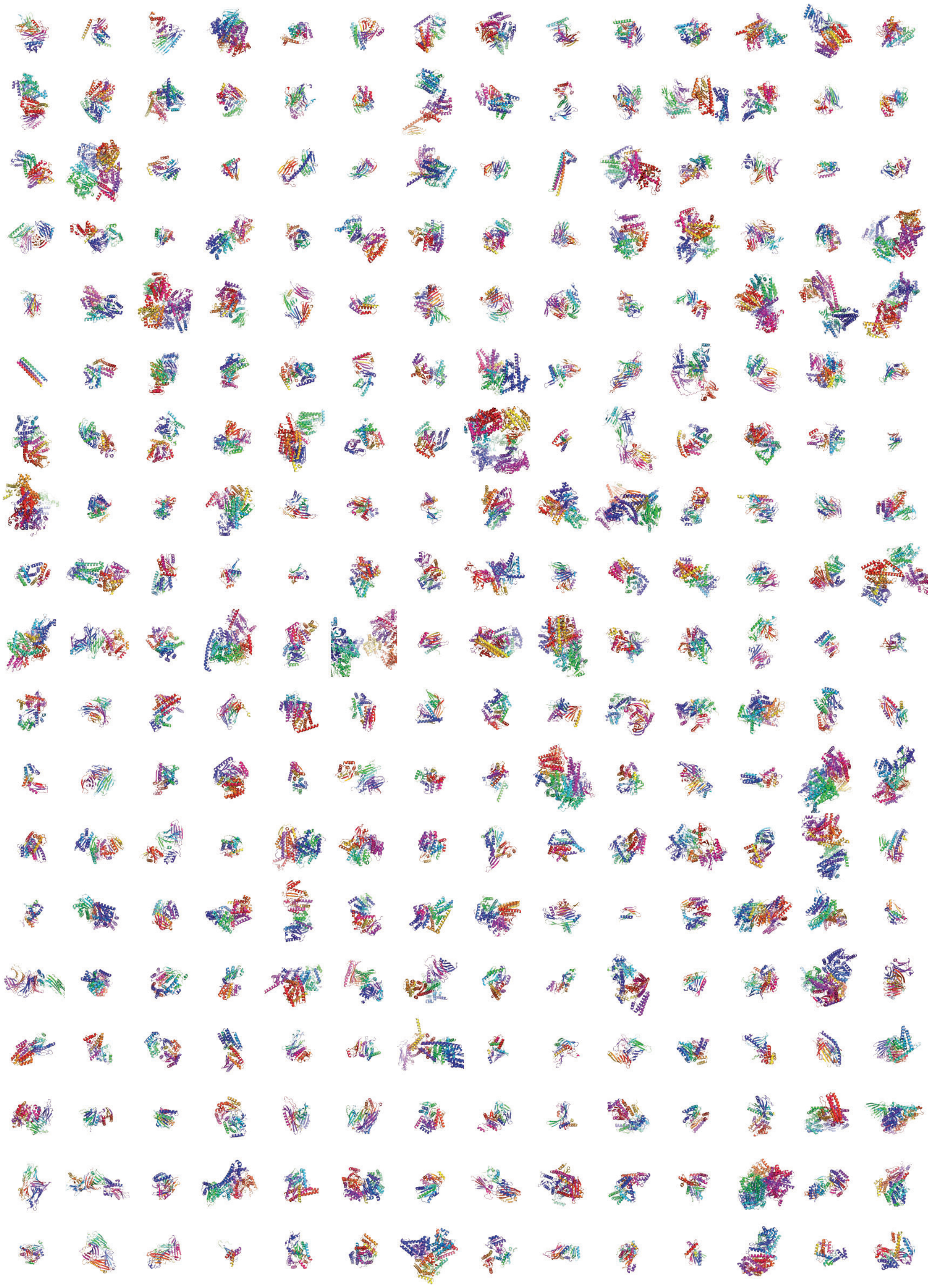
The novelty of Chroma-generated samples was assessed by comparing their structures to natural protein folds. Using TMalign [Zhang and Skolnick, 2005], they were each aligned to the 32k structurally conserved domains from CATHdb S40 set [Sillitoe et al., 2021] and filtered for TMscore greater than 0.5 when normalized by the shortest sequence. The number of domains needed to cover at least 80% of the query was greedily determined by identifying the hits with the highest



Supplementary Figure 6: **Random single chain samples from backbone model A.**

number of residues within 5 *Å* of the query that wasn't already covered. The number of domains required increases with query size given that CATH domains typically have a length ranging between 50 and 200 amino acids. As a baseline, we ran the test set with the same algorithm.

Single chain structures from Chroma and the test set were embedded in 31 Gauss Integral dimen-



Supplementary Figure 7: **Random complex samples from backbone model A.**

sions using the pdb2git program from the Phaistos suite [Harder et al., 2012] [Borg et al., 2009]. It failed to embed structures with chain breaks or with protein lengths greater than 876. The final set of 6492 generated structures and 561 natural folds were projected to a 2 dimensions space using UMAP [McInnes et al., 2018] with default parameters of 25 neighbours and a minimal distance of

Metric	Description	Normalization
Secondary structure content (SS_i)	Distribution of Helix, Strand, Coil for given structure	none
Mean Residue Contact (C_{mean})	Average number of contacts per residue for any given structure	none
Long-range Residue Contact (C_{long})	Number of long-range contacts per residue; long-range residue interaction means a pair of interacting residues separated by 24 or more residues in sequence	none
Contact Order (CO)	Average sequence distance between contacting residues normalized by the total length of the protein; higher contact orders generally indicate longer folding times	$CO/N^{-0.3}$ [Ivankov et al., 2003]
Radius of Gyration (R_g)	Root mean square distance of structure's atomic coordinates from its center of mass	$R_g/N^{0.4}$ [Tanner, 2016]

Table 4: **Metrics to describe backbone geometry of structures**

Sample type	T	λ_0	ψ	**Additional details**
Unconditioned	200	10.	2.	Appendix H
Shape	2000	5.	2.	Appendix J
Infilling	400	5.	N/A	Appendix K
Symmetry	400	16.	2.	Appendix L
Natural language (ProCap)	200	10.	2.	Appendix O, scaled to have norm ≤ 10

Table 5:

0.5.

H.4 Structure prediction-based designability

Structures were generated using Chroma with $\lambda_0 = 10$. Sequences conditioned on generated structures were designed by using our sequence design module with a Potts decoder to create a pairwise sequence-level energy table representing the sequence landscape compatible with the fold. Sequences were sampled from this Potts model using 10 independent cycles of simulated annealing Monte Carlo (MC), each with $200 \cdot n$ steps with n being the length of the protein. The score used in the annealing was the Potts energy plus a flat-bottom restraint energy around the sequence complexity calculated as equation 4 in [Wootton and Federhen, 1993]. The restraining potential linearly penalized sequence complexity dropping below one standard deviation under the mean for a native sequence of equivalent length, and was otherwise zero. For each generated structure, the above MC procedure was run 100 times to produce 100 sequences, each of which were used as input into OmegaFold [Wu et al., 2022b] for structure prediction. For each backbone, the highest obtained TM score was used as the evidence for whether the underlying backbone was designable in our analysis.

H.5 TERM-based designability

RMSD-based search was performed using an in-house implementation of the method FASST available as part of the open-source software package Mosaist (https://github.com/Grigoryanlab/Mosaist). The method is a close relative of the previously published approach MASTER [Zhou and Grigoryan, 2014, 2020a]. The training and test sets for Chroma were used as the search database and the set of native proteins in this analysis, respectively. Although the test and training sets have been split by chain-level sequence homology, we took further care to exclude any apparent homologues of native TERMs from consideration as matches. To this end, we compared the local 31-amino acid sequence windows around each TERM segment and its corresponding match, with any pairings reaching 60% or more sequence identities not being allowed to participate in a match.

I Programmability: Overview

Overview In principle, the set of proteins satisfying a given set of functional constraints can be described using Bayes' Theorem,

$$p(\text{protein}|\text{function}) \propto p(\text{protein}) \times p(\text{function}|\text{protein})$$

where the posterior distribution of proteins $p(\text{protein}|\text{function})$ is proportional to the *likelihood* of satisfying the set of functional constraints $p(\text{function}|\text{protein})$ times the *prior* probability of the protein molecule being functional $p(\text{protein})$. This characterization has been appreciated for several decades [Simons et al., 1997], but leveraging it is challenging in practice for two reasons. First, developing tractable and accurate priors over the space of possible proteins has proven extremely difficult owing to the tremendous complexity in a single protein system (a complex can easily have $> 10^4$ atoms) and the intractabilities of marginalizing out low level details. Secondly, even with an accurate prior, sampling from the space of polypeptide conformations is extremely difficult as it will generally be a rugged landscape under which global optimization is infeasible.

One potential way to make the difficult inverse problem posed by protein design more tractable is given by contemporary methods from machine learning. In particular, diffusion models make conventionally intractable inference and inverse problems tractable by learning to gradually transform a complex data distribution into a simple and tractable distribution [Sohl-Dickstein et al., 2015, Song and Ermon, 2019]. This has enabled transformative applications in text-to-image modeling [Ramesh et al., 2022, Saharia et al., 2022].

The manner in which diffusion models enable Bayesian inversion can be made especially clear in the continuous-time formulation of Diffusion models, where we can take advantage of the fact that the score functions are independent of normalizing constants and we can therefore express the time-dependent *posterior score* $\nabla_{\mathbf{x}} \log p_t(\mathbf{x}|\mathbf{y})$ as the sum of the *prior score* $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ and the *likelihood score* $\nabla_{\mathbf{x}} \log p_t(\mathbf{y}|\mathbf{x})$ because

$$\begin{aligned}\nabla_{\mathbf{x}} \log p_t(\mathbf{x}|\mathbf{y}) &= \nabla_{\mathbf{x}} \log \frac{p_t(\mathbf{x})p_t(\mathbf{y}|\mathbf{x})}{p_t(\mathbf{y})} \\ &= \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) + \nabla_{\mathbf{x}} \log p_t(\mathbf{y}|\mathbf{x}) - \cancel{\nabla_{\mathbf{x}} \log p_t(\mathbf{y})} \\ &= \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) + \nabla_{\mathbf{x}} \log p_t(\mathbf{y}|\mathbf{x}).\end{aligned}$$

We describe how to take advantage of this via the posterior SDE and ODE in Appendix A.

We note that while we primarily focus on classifier conditioning of backbone conditioning throughout this work, it is also very feasible to extend this to tractable *sequence* classifier conditioning with new discrete sampling methods based gradient-based locally-adjusted MCMC proposals [Grathwohl et al., 2021, Rhodes and Gutmann, 2022].

Conditioner	**Dim.**	**Granularity**	**Type**	**Examples and applications**
Sequence	1D	Residue	Learned	Sequence design & sequence conditioning
Domain classifier	1D	Chain	Learned	Pfam, CATH, Taxonomy
Secondary Structure	1D	Residue	Learned	Topological constraints
Distances (contacts)	2D	Atoms	Analytic	Fold constraints, binder design
Sub-structure RMSD	1D	Atoms	Analytic	Scaffolding-based constraint
Sub-structure	1D	Atoms	Analytic	Structural “in filling”
Shape constraint	1D	Atoms	Analytic	Molecular shape control
Symmetry constraint	1D	Atoms	Analytic	Self-assembling oligomers, e.g. capsids
Text caption	1D	Chain, Complex	Learned	Natural language prompting

Table 6: **Conditioners available to Chroma.**

I.1 Example applications of constraint composition

We list a table of composable constraint models in Table 6. Some practical protein design problems that could be realized through composite constraints under this framework are

***De-novo* binders** Combine (i) substructure conditioning on antigen, (ii) optional scaffold constraint on binder, and (iii) contact constraints on epitope/paratope

Enzyme miniaturization Use substructure RMSD to graft an active site into a novel scaffold or known scaffold (via combining with substructure constraints

Nanostructure control Use the shape constraint to sample novel designable folds or complexes satisfying arbtitrary shape constraints

Nanomaterial design Combine nanostructure control with interfacial binding constraints on periodic boundary conditions

J Programmability: Distance-based constraints

J.1 Motivation and problem statement

In some instances, it may be useful to generate diverse protein chain and/or complex structures under the constraint that one or more specific residue pairs be in spatial proximity (i.e., form a "contact"). Such a conditioner could be used, for example, in designing binders, to ensure that the desired binding site is being engaged. Or it could be used to insure some desired topological properties–i.e., the proximity of N- and C-termini (e.g., for ease of circular permutation). Assuming that we are interested in conditioning on a contact between atoms i and j, we are seeking the probability that the distance between these two atoms in the fully denoised structure is below some desired cutoff c, $d_0^{ij} < c$, given a noised sample at time t and the corresponding distance d_t^{ij}.

J.2 Approach

One approach would be to train a time-dependent classifier $p_t(\mathbf{y}|\mathbf{x}(t))$ to classify noisy inputs. For the case of a contact classifier, however, we can directly compute the desired probability analytically. By definition of our forward noise process, the i-th coordinate of our protein at time 0 and t are related to each other by

$$\mathbf{x}_0^{(i)} = \frac{\mathbf{x}_t^{(i)}}{\sqrt{\alpha_t}} - \sqrt{(1-\alpha_t)}[\mathbf{R}\mathbf{z}]_i$$

Below we sketch the derivations of the distribution d_0^{ij} cases of Brownian and globular noise schedules.

J.2.1 Brownian noise

Here we have that

$$[\mathbf{R}\mathbf{z}]_i = \gamma\sum_{k}^{i} z_k - \frac{\gamma}{N}\sum_{j=1}^{N}\sum_{k=1}^{j} z_k + \delta z_1$$

and therefore

$$\mathbf{x}_0^{(j)} - \mathbf{x}_0^{(i)} = \frac{\mathbf{x}_t^{(j)} - \mathbf{x}_t^{(i)}}{\sqrt{\alpha_t}} - \gamma\sqrt{1-\alpha_t}\sum_{k=j}^{i} z_k$$

But as $\sum_{k=j}^{i} z_k \sim \mathcal{N}(0,|i-j|)$ by independence of $\{z_i\}$, we have $\mathbf{x}_0^{(j)} - \mathbf{x}_0^{(i)} \sim \mathcal{N}(\frac{\mathbf{x}_t^{(j)}-\mathbf{x}_t^{(i)}}{\sqrt{\alpha_t}}, \gamma^2(1-\alpha_t)\cdot|i-j|)$, so that:

$$\frac{(d_0^{ij})^2}{|i-j|(1-\alpha_t)\gamma^2} \sim \text{NonCentralChiSquared}\left[\frac{(d_t^{ij})^2}{|i-j|(1-\alpha_t)\alpha_t\gamma^2}, k=3\right]$$

For a contact threshold $c > 1$ we have:

$$d_0^{ij} < c \iff (d_0^{ij})^2 < c^2 \iff \frac{(d_0^{ij})^2}{|i-j|(1-\alpha_t)\gamma^2} < \frac{c^2}{|i-j|(1-\alpha_t)\gamma^2}$$

and so we can conclude that $p_t\left(d_0^{ij} < c|\mathbf{x_t}\right)$ is given exactly by the CDF of the noncentral chi-squared distribution above, evaluated at $c^2\left[|i-j|(1-\alpha_t)\gamma^2\right]^{-1}$.

J.2.2 Globular noise

For the globular chain noise process we instead have that

$$[\mathbf{Rz}]_i = a\sum_{k=2}^{i} b^{i-k} z_k + a\frac{b^{i-1}}{(1-b^2)^{1/2}} z_1$$

By substituting we see

$$\mathbf{x}_0^{(j)} - \mathbf{x}_0^{(i)} = \frac{\mathbf{x}_t^{(j)} - \mathbf{x}_t^{(i)}}{\alpha_t^{1/2}} + (1-\alpha_t)^{1/2}([\mathbf{Rz}]_j - [\mathbf{Rz}]_i)$$

So that $\mathbf{x}_0^{(j)} - \mathbf{x}_0^{(i)} \sim \mathcal{N}(\frac{\mathbf{x}_t^{(j)} - \mathbf{x}_t^{(i)}}{\alpha_t^{1/2}}, (1-\alpha_t)\mathrm{Var}([\mathbf{Rz}]_j - [\mathbf{Rz}]_i))$. But, assuming $j > i$:

$$\mathrm{Var}([\mathbf{Rz}]_j - [\mathbf{Rz}]_i) = \frac{2a^2(1-b^{j-i})}{1-b^2} =: \sigma_{j-i}^2$$

It then follows that

$$\frac{\mathbf{x}_0^{(j)} - x_0^i}{(1-\alpha_t)^{1/2}\sigma_{j-i}} \sim \mathcal{N}(\frac{\mathbf{x}_t^{(j)} - \mathbf{x}_t^{(i)}}{\sigma_{j-i}(1-\alpha_t)^{1/2}\alpha_t^{1/2}}, \mathbf{I})$$

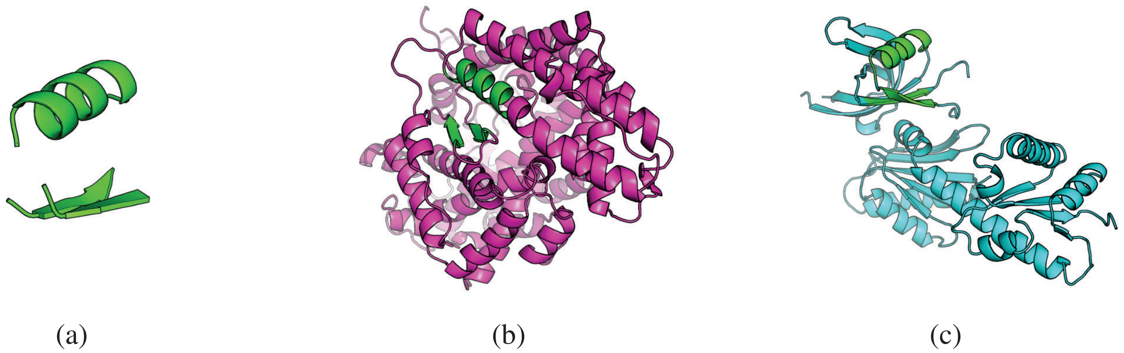
and finally

$$\frac{(d_0^{ij})^2}{(1-\alpha_t)\sigma_{j-i}^2} \sim \mathrm{NonCentralChiSquared}\left[\frac{(d_t^{ij})^2}{\sigma_{j-i}^2\alpha_t(1-\alpha_t)}, k=3\right]$$

K Programmability: sub-structure RMSD

K.1 Motivation and problem statement

It would be very useful for a variety of protein engineering applications to condition structure generation on the presence of a particular structural "motif." By this we mean an arbitrary sub-structure, composed of any number of disjoint backbone segments, that we would like to exist within our final generated structure. In practice, such a motif could represent a functional (e.g., catalytic) constellation of residues or a metal/small-molecule binding site—this could be useful for designing enzymes or other functional proteins, by exploring ideas around a core functional mechanism. In another example, the motif could correspond to a "scaffolding" part of the molecule that we would want to preserve—e.g., the binding scaffold that can admit different loop conformations. Or the motif could represent a desired epitope that we would like to faithfully present on the surface of a generated protein in the context of vaccine design. Fig. 8 shows an example motif and two unrelated native protein structures in which this motif is found with low RMSD.



(a)

(b)

(c)

Supplementary Figure 8: **Motifs can occur in entirely unrelated structural contexts.** **a,** An example motif composed of three disjoint segments. **b,** PDB entry 3NXQ harbors the motif with a backbone RMSD of 0.45 $\AA$. **c,** PDB entry 3OBW harbors the motif with a backbone RMSD of 0.64 $\AA$.

The task of determining whether the pre-specified motif is present in a given structure S is simple–we can, for example, find the substructure of S with the lowest optimal superposition root-mean-squared-deviation (RMSD) to the motif in question and ask whether this RMSD value is below a desired cutoff (this can be done using previously published algorithms [Zhou and Grigoryan, 2015, 2020b]). But what we need for conditional generation is the ability to estimate the probability that the final de-noised structure will harbor the desired motif, given a noisy structure at the current time point in the diffusion.

K.2 An empirical approach

Specifically, if $\mathbf{x}_t \in \mathbb{R}^{N\times 3}$ is our coordinate array and the forward diffusion process is represented by:

$$\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\mathbf{R}\boldsymbol{\epsilon} \qquad\qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

we need to express $p(y|\mathbf{x}_t)$–the probability that $\mathbf{x}_0$ contains the motif given $\mathbf{x}_t$, where y stands for the condition of motif presence (e.g., as defined by RMSD to a template motif below a desired cutoff). If we define the presence of a motif in terms of optimal-alignment best-fit RMSD being below a cutoff, we need to understand how this RMSD behaves (in a probabilistic sense) as a function of noise. Further, as we will generally not be given where within $\mathbf{x}_t$ the motif may be (i.e., we would not know *a priori* the matching between motif atoms and a sub-structure of the target structure), our $p(y|\mathbf{x}_t)$ needs to integrate information for the full structure $\mathbf{x}_t$ to determine possible motif location(s). Achieving this analytically seems non-trivial. For this reason, here we consider an empirical approach to expressing $p(y|\mathbf{x}_t)$.

The goal is to observe the behavior of optimal-alignment best-fit RMSD in practice, as a function of α_t, using a set of reasonable structures and diverse motifs, and find an analytical approximation for its probability distribution. Specifically, given a motif m and a structure represented by $\mathbf{x}_t$, let r_t represent the RMSD of optimal alignment of m onto $\mathbf{x}_t$ (i.e., the lowest RMSD between atoms of m and any sub-structure of $\mathbf{x}_t$), and r_0 represents the RMSD induced by the same matching in the context of structure $\mathbf{x}_0$. We seek to approximate the cumulative distribution function $F(r_0 -$

$r_t|\mathbf{x}_t,\alpha_t)$. With this, we would calculate $p(y|\mathbf{x}_t)$ as $p(y|\mathbf{x}_t) = p(r_0 < \sigma|\mathbf{x}_t) = p(r_0 - r_t < \sigma - r_t|\mathbf{x}_t) = F(\sigma - r_t|\mathbf{x}_t,\alpha_t)$, where σ is the desired RMSD cutoff for classifying the existence of the motif. The procedure in algorithm 2 was run to generate an empirical data set of 10^6 points to describe the behavior of $r_0 - r_t$.

Algorithm 2 Data generation procedure for RMSD classifier fitting

```
Require: $T$–training set of protein complex structures, $N$–number of data points desired
  $i \leftarrow 1$
  while $i \neq N$ do
      $\mathbf{x}_0 \leftarrow$ random protein complex from database                ▷ Noise-free structure
      $\mathbf{x}' \leftarrow$ a different random protein complex from database
      $r \leftarrow$ uniform random number $\in [6;30]$                            ▷ Motif radius
      $R \leftarrow$ random residue from $\mathbf{x}'$
      $m \leftarrow R$ and all residues within $r$ of $R$                          ▷ Sampled motif
      $t \leftarrow$ uniform random number $\in [0;1]$                             ▷ Noise level
      $\mathbf{x}_t \leftarrow \sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\mathbf{R}\epsilon$   ▷ Noised structure
      $A \leftarrow$ optimal alignment of $m$ onto $\mathbf{x}_t$                  ▷ [Zhou and Grigoryan, 2015]
      $r_t \leftarrow$ best-fit RMSD of $m$ aligned onto $\mathbf{x}_t$ according to $A$
      $r_0 \leftarrow$ best-fit RMSD of $m$ aligned onto $\mathbf{x}_0$ according to $A$
      record $r_t$, $r_0$, $t$, $\alpha_t$, $m$
  end while
```

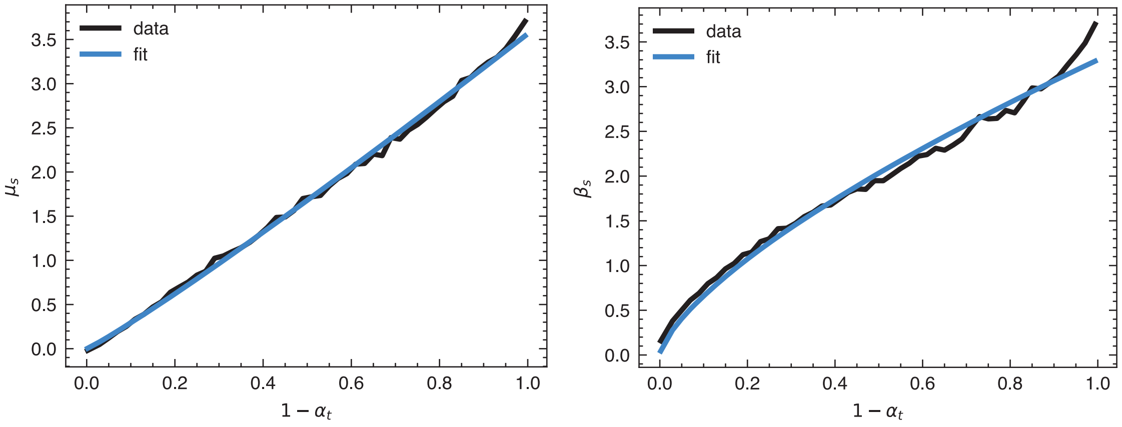
K.2.1 Motif size and complexity dependence

Clearly, the distribution of r_t (and $\Delta r_t = r_0 - r_t$) should depend on α_t. But these distributions should also depend on the size and complexity of the motif. For example, in the extreme case when the motif consists of a single atom, r_t will always be zero. On the other hand, for large and complex motifs, we may expect r_t to increase rapidly with added noise.

The simplest surrogate for motif complexity is its size—i.e., the number of residues it involves. However, under our noise model, the atoms closer to each other in the protein chain will move in a more correlated manner than those that are farther apart. So it should matter whether the motif consists of multiple short disjoint segments matching to far-away (in sequence) portions of the target structure versus a motif consisting of one long contiguous segment. As a purely empirical measure to capture this notion, we propose the following effective length definition:

$$L_e = -\log\left[\frac{2}{n\,(n-1)}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\frac{1}{\sqrt{|i-j|}}C(i,j)\right]$$

where $C(i,j)$ is an indicator function that is 1 if atoms i and j are part of the same chain and 0 otherwise. The motivation for the inverse square root of the index distance is from Brownian motion (displacement distance growing as the square root of time, here the number of atom hops). And the motivation for ignoring atom pairs from different chains is that these move independently under our noise model. In practice, L_e appears to better explain variation of $r_t - r_0$ than just pure number of motif residues L, despite the fact that overall L_e correlates somewhat closely with $\log(L)$.



(a)

(b)

Supplementary Figure 9: **Fitting empirical Gumbel-distribution parameters μ_s and β_s from local α_t-window fits (which capture L_e-dependence of location and scale, respectively) as analytical functions of α_t.** Both the location **(a)** and scale **(b)** parameters vary monotonically with α_t, closely following the functional form $k \cdot (1-\alpha_t)^n$.

K.2.2 Distribution of $r_0 - r_t$

We expect the distribution of $r_0 - r_t$ to depend on α_t and L_e. To get a sense of the general shape of this distribution and its dependence on α_t, one can take slices of the training data with α_t in different narrow ranges. Inspection and fitting of these α_t-window histograms of $r_0 - r_t$ suggested that the Gumbel family of distribution should work reasonably well for describing the observed variations.

The dependence on L_e can be captured defining the parameters of the Gumbel distribution as functions of L_e. Towards defining a reasonable functional form, we consider extremes. The Gumbel distribution has two parameters–location μ and scale β. The latter is solely responsible for the variance (i.e., $\frac{\pi^2}{6}\beta^2$) and the mean is contributed to by both ($\mu + \beta\gamma$, where γ is the Euler–Mascheroni constant, or approximately 0.577). Clearly, for a motif that only has one atom, we expect Δr_t to be a delta function at 0, meaning that both μ and β would be zero. And in general, for small (and simple) motifs we would expect μ and β to be low, while for large (and complex) motifs we would expect it to be high. Thus, both μ and β should be monotonically increasing functions of L_e that pass through the origin. Experimentation with different curve families under these criteria, using the overall data likelihood as the objective metric (see below), we arrived at the simple linear parameterization option as being best–i.e., where $\mu = \mu_s L_e$ and $\beta = \beta_s L_e$ with μ_s and β_s being fitting parameters.

K.3 Fitting procedure

With the parameterization choices above, the fitting approach took the following steps.

K.3.1 Fitting individual α_t windows

For 50 equally-spaced α_t windows, fit the observed $\Delta r_t = r_0 - r_t$ to Gumbel distributions, whose location and scale parameters linearly depend on L_e of each motif, using likelihood maximization. Specifically, the likelihood function being maximized was:

$$\log \mathcal{L} = \sum_{i=1}^{N_D} -\log(\beta_s L_e^i) - \frac{\Delta r_t^i - \mu_s L_e}{\beta_s L_e} - \exp(-\frac{\Delta r_t^i - \mu_s L_e}{\beta_s L_e})$$

were L_e^i and Δr_t^i are the effective motif length and Δr_t is the i-th data point, respectively, and N_D is the number of data points. The result of this procedure then estimates μ_s and β_s parameters specific for the current α_t window.

K.3.2 Fitting parameters as functions of α_t

We next fit μ_s and β_s as functions of α_t analytically. The functional form chosen for both parameters was $k \cdot (1 - \alpha_t)^n$, ensuring that at $\alpha_t = 1$ both parameters necessarily become zero (i.e., as the noise level reaches zero, the Δr distribution should approach a delta function). Both the raw locally-fit values of μ_s and β_s and the corresponding analytical fits are shown in Fig. 9.

K.3.3 Assessment of fit quality

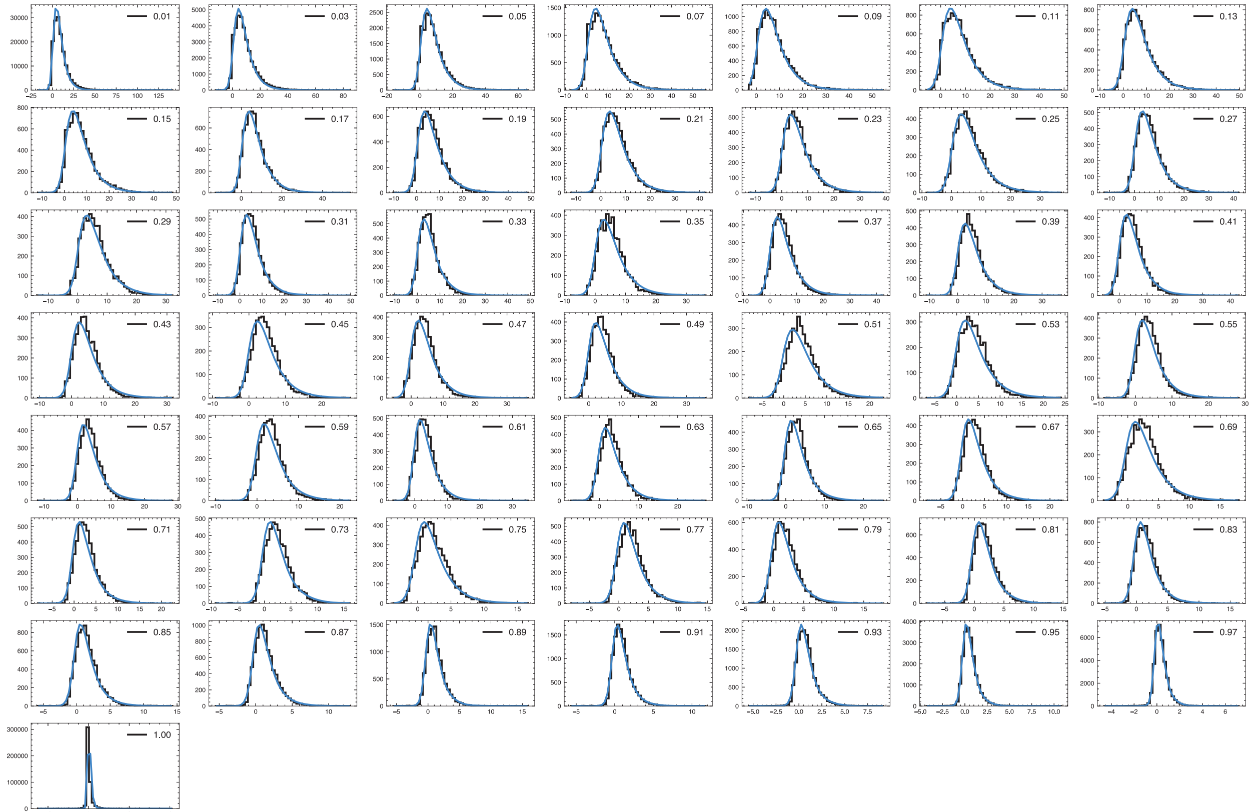
Given the now fully parameterized $p(\Delta r_t | \alpha_t, L_e)$, we integrate over L_e in each α_t window to produce the expected distribution of Δr_t and compare with the corresponding observed distribution to evaluate the overall goodness of fit. The results, shown in Fig. 10, demonstrate an excellent overall fit. This is especially encouraging given the wide range of motif sizes (anywhere from one to over 350 residues) and numbers of disjoint segments present in the training set (one to five).

K.4 Conditioning with pre-registration: structural infilling

In some cases, the residue indices of the desired sub-structure in the context of the larger structure are given a priori, for instance, in the case of imputing missing structural information. let $\mathcal{S}, \mathcal{M} \subset [1, \cdots, N]$ denote the atoms comprising the unknown scaffold and known motif respectively.

K.4.1 Related work

Song et al. [2021] presents a replacement method for drawing approximate conditional samples from $p(\mathbf{x}_0^{\mathcal{S}} | \mathbf{x}_0^{\mathcal{M}})$ in which one samples a sequence of noised motifs $\bar{\mathbf{x}}_{1:T}^{\mathcal{M}} \sim q(\mathbf{x}_{1:T}^{\mathcal{M}} | \mathbf{x}_{(0)}^{\mathcal{M}})$, then running diffusion backwards in time but at each time step replacing $\mathbf{x}_t^{\mathcal{M}} \leftarrow \mathbf{x}_t^{\mathcal{M}}$ before sampling $\mathbf{x}_{t-1} \sim p(\mathbf{x}_{t-1} | \mathbf{x}_t)$. Trippe et al. [2022] demonstrated that this method introduces irreducible error that is exacerbated by the correlation introduced by q and propose a particle-filtering based approach which furnishes arbitrarily accurate conditional samples given sufficient computation. Informally, the error introduced by the replacement method arises from imputing noised motifs that are highly unlikely given the corresponding noised scaffold.



Supplementary Figure 10: **Comparison between expected (based on inferred model parameters) and observed distributions of Δr_t in different windows of α_t.** Observed histograms are show in black and the analytical prediction in blue. Legend indicates the mean α_t value for each window.

K.4.2 Method

Given $\mathbf{x}^{\mathcal{M}}$, we impute for every contiguous missing fragment $\mathcal{S}_i$ we sample $\mathbf{x}^{\mathcal{S}_i}$ from a brownian bridge with endpoints fixed at $a = \mathbf{x}^{\mathcal{S}_i - 1}$ and $b = \mathbf{x}^{\mathcal{S}_i + 1}$ for internal fragments as follows. Let $M = |\mathcal{S}_i|$, we sample $(z_i)_{i=1}^{M} \sim \mathcal{N}(0, \sigma^2)$, where σ^2 was tuned to 4.0.

$$[x^{\mathcal{S}_i}]_k = (1 - \frac{k}{M})a + \sum_{l=1}^{k} z_l - \frac{k}{M}(\sum_{l=1}^{M} z_l - b)$$

If a fragment terminates on the right (e.g there is no right endpoint), instead:

$$[x^{\mathcal{S}_i}]_k = a + \sum_{l=1}^{k} z_l$$

with left-terminating fragments handled similarly.

Once initialized, we integrate backwards in time the conditional probability flow ODE where the conditional $p_t(\mathbf{x}_0^{\mathcal{M}} | \mathbf{x}_t^{\mathcal{M}})$ is taken be $\mathrm{RMSD}(\mathbf{x}_0^{\mathcal{M}}, \mathbf{x}_t^{\mathcal{M}})$ To address issues with clashes and discontinuities, we include terms:$p_{\text{clash}}(\mathbf{x}_0 | \mathbf{x}_t)$ and $p_{\text{violation}}(\mathbf{x}_0 | \mathbf{x}_t)$ where p_{clash} is a $L1$ penalty on all-atom distance matrix of $\mathbf{x}_t$ restricted to non-adjacent residues penalizing distances less than 1.5 $\AA$., and $p_{\text{violation}}$ is given by the violation loss defined in [Jumper et al., 2021].

L Programmability: Symmetry

L.1 Motivation

Built from identical subunit proteins, many protein complexes are assembled symmetrically. Many symmetric complexes such as tube-shaped channel proteins and icosahedral viral capsids are biologically important [Goodsell and Olson, 2000]. Incorporating symmetry in computational protein generation holds promise in designing large functionalized protein complexes [Hsia et al., 2016]. To fully explore the sampling of protein complexes subject to symmetry constraints, we propose a method to symmetrize the underlying ODE/SDE sampling to satisfy any prescribed Euclidean symmetries.

Incorporating group equivariance in machine learning has been an important topic in the machine learning community. [Cohen and Welling, 2016] Incorporating space group symmetries is critical in molecular simulations [Cox and White, 2022, Zabrodsky et al., 1992]. In this work, we proposed a method to incorporate symmetry for diffusion probabilistic models with applications in generating large-scale protein complexes with arbitrary symmetry groups.

L.2 Symmetry breaking in sampling

Let $G = \{g_i\}_{i=0}^{N}$ be a collection of symmetry operations that form a group such as point groups and space groups. For point sets in $\mathbb{R}^3$, these symmetry operations can be represented as a set of orthogonal transformations (rotation/reflection) and translations. The sampling SDE proposed in our work can be generally cast in the following form:

$$\mathrm{d}\mathbf{x} = F(\mathbf{x_t}, t)\,\mathrm{d}t + \sqrt{\beta_t}\ \mathbf{R}\,\mathrm{d}\bar{\mathbf{w}} \tag{5}$$

For synthesizing symmetric protein complexes, we want to sample complexes $\boldsymbol{X}_{t=0} \in \mathbb{R}^{N\times n\times 3}$ which are built from $N = |G|$ identical single-chain proteins $\mathbf{X}^{(\mathbf{i})} \in \mathbb{R}^{n\times 3}$ where n is the number of residues for each subunit. Under the time-dependent noise $\sqrt{\beta_t}\ \mathbf{R}\,\mathrm{d}\bar{\mathbf{W}} \in \mathbb{R}^{N\times n\times 3}$, the samples are generated from:

$$\begin{aligned}\mathbf{X_0} &= \mathtt{SDESOLVE}(\mathbf{X_1}) \\ &= \mathbf{X_1} + \int_{t=1}^{0} F(\mathbf{X_t}, t)\,\mathrm{d}t + \int_{t=1}^{0} \sqrt{\beta_t}\ \mathbf{R}\,\mathrm{d}\bar{\mathbf{W}}\end{aligned} \tag{6}$$

To constrain the sample generation to respect symmetries for an arbitrary group G, the SDE/ODE dynamics need to be G-invariant up to a permutation of subunits. Let $\cdot$ represent the symmetric operations (rotation, reflection, and translation) performed on point sets in $\mathbb{R}^3$, we define the sampling procedure $\mathtt{SDESOLVE}: \mathbb{R}^{|G|\times n\times 3} \to \mathbb{R}^{|G|\times n\times 3}$ with $X_0 = \mathtt{SDESOLVE}(X_1)$ being the desired samples. The sampling procedure needs to follow the following invariance condition:

$$\mathtt{SDESOLVE}(g_i \cdot \mathbf{X_1}) = g_i \cdot \mathtt{SDESOLVE}(\mathbf{X_1}) = \sigma_i\, \mathtt{SDESOLVE}(\mathbf{X_1})\ , \forall g_i \in G \tag{7}$$

where g_i indicates the i-th group element in G and we impose an arbitrary order on G and our method is equivariant to the permutation of subunits. σ_i is the induced permutation operation satisfying the relation: $g_i G = \sigma_i G$, as computed from the group multiplication table (also called the Caley table).

The first equality in eq. (7) is trivially satisfied if $F(\cdot)$ is $E(3)$ equivariant, as G consists of only orthogonal transformations and translations. However, the second equality is generally not satisfied. For molecular simulations where the Hamiltonian dynamics is used, the second equality can be satisfied if (i) the energy function is E(3) invariant, and (ii) the initial $\mathbf{X_1}$ and $\frac{d\mathbf{X_1}}{dt}$ are symmetric, i.e $g_i \cdot \mathbf{X_1} = \sigma_i \mathbf{X_1}, g_i \cdot \frac{d\mathbf{X_1}}{dt} = \sigma_i \frac{d\mathbf{X_1}}{dt}$. At each successive time step, X_t automatically satisfies the prescribed G-symmetry. This approach confines both the position and momentum update to ensure the sampled configurations remain symmetric.

However, this is not the case with SDE/ODE sampling in our framework. We list three origins of symmetry-breaking error if eq. (6) is used: (i) $F(\mathbf{X_t}, t)$ uses distances as features and is automatically $E(3)$ equivariant. However, because the protein feature graphs are generated probabilistically, $F(g_i \cdot \mathbf{X_t}, t) \neq g_i \cdot F(\mathbf{X_t}, t)$ with each subunit protein x^i having different geometric graphs, albeit being symmetric. (ii) Our polymer structured noise is randomly sampled from $\mathcal{N}(x^i; \mu, \mathbf{R}\mathbf{R}^\intercal)$, so each subunit protein has different chain noises, i.e. $g_i \cdot \mathbf{R}\,\mathrm{d}\bar{\mathbf{W}} \neq \mathbf{R}\,\mathrm{d}\bar{\mathbf{W}}$. (iii) The sampling procedure requires solving an ODE/SDE which is vulnerable to accumulated integration error. Integration error can induce unwanted geometric drifts such as rotation and translation[Harvey et al., 1998], and be a substantial symmetry-breaking force.

L.3 Symmetric sampling

By leveraging the desired symmetry, we averaged the SDE/ODE update from symmetric subunit proteins and broadcast with symmetry operation G. By performing the symmetric broadcasting, we remove the symmetry-breaking error for each symmetric subunit. The goal is to construct sampling protocols that satisfy eq. (7) which also capture interactions between subunits.

L.3.1 Symmetric initialization

For samples to remain G-invariant throughout the sampling process, it is necessary to have the initial noised structure $\mathbf{X_1}$ symmetrized. We define the following symmetrized "copying" operation:

$$\mathbf{X_1} = G \cdot \mathbf{x_1} = \{g_i \cdot \mathbf{x_1} | \, g_i \in G\}, \mathbf{x_1} \sim \mathcal{N}(\mathbf{x_1}; \mu, \mathbf{R}\mathbf{R}^\intercal) \tag{8}$$

By construction, the generated structures are symmetric under G, i.e.

$$g_i \cdot \mathbf{X_1} = (g_i G) \cdot \mathbf{x_1} = \sigma_i \mathbf{X_1}, \; \forall g_i \in G \tag{9}$$

L.3.2 Symmetrized SDE

Trivial symmetrization Given symmetrically initialized $\mathbf{X_1}$, a trivial construction is to set

$$F_{sym.}(\mathbf{X_t}, t) = (G g_i^{-1}) \cdot F(\mathbf{X_t^{(i)}}, t), \tag{10}$$

where i is the index to any subunit that is generated with g_i and we dropped t-dependence in F for a more compact notation. Note that one does not need to make a special choice of $\mathbf{x_t^{(i)}}$ and g_i as all subunits are symmetric. However, it is more convenient to select i where g_i corresponds to the identity transformation. The method only requires performing an update on a single subunit, followed by symmetrized broadcasting. However, it satisfies eq. (7), because it fails to capture any subunit-subunit interactions which are important in capturing protein complexes.

Symmetric broadcasting To incorporate subunit interactions, we use the entire symmetric complex as input for $F(\cdot,\cdot)$ so that the subunit interactions are captured, and our backbone GNN is also designed to capture large protein complexes. Similar to the trivial construction described above, we select a particular g_i and compute:

$$\underbrace{F_{sym.}(\mathbf{X}_t, t)}_{|G| \times n \times 3} = \underbrace{(G g_i^{-1})}_{|G| \times 3 \times 3} \cdot \underbrace{[F(\mathbf{X_t}, t)]_i}_{n \times 3} \tag{11}$$

For noise, we simply sample polymer structured noise and broadcast:

$$\mathbf{R} \, \mathrm{d}\bar{\mathbf{W}}_{\mathbf{sym}} = (G g_i^{-1}) \cdot \mathbf{R}[\mathrm{d}\bar{\mathbf{W}}]_i \tag{12}$$

where $[F(\mathbf{X_t})]_i$ retrieves the gradient update for subunit i. Intuitively, the method computes the gradient update globally and broadcasts the update vector to all symmetric positions.

For simulating large complexes where certain long-range interactions are not as critical, we can instead focus on updating a subset $\mathcal{S} \subset [1, ..., |G|]$ of subunits in $\mathbf{X_t}$ to save memory and time. Given a chosen subunit i, we deduce $\mathcal{S}$ by choosing the k-nearest neighbor ((k-NN)) subunits based on the distances between the subunits' geometric centers, so that short-range subunit interactions are included. We select K subunits in this way, and K is a hyperparameter.

Symmetric averaging An alternative symmetrization method is to average the $F(\boldsymbol{X}_t, t)$ at symmetric coordinates over all possible symmetry operations in G.

$$[F_{sym}(\mathbf{X_t}, t)]_i = \underbrace{\frac{1}{|G|}(g_i G^{-1})}_{|G|\times 3\times 3} \cdot \underbrace{F(\boldsymbol{X}_t, t)}_{|G|\times n\times 3} \quad = \frac{1}{|G|}\sum_i g_i \cdot (\sigma_i^{-1} F(\boldsymbol{X}_t, t)) \tag{13}$$

where $(g_i G^{-1})$ aligns the $[F(\mathbf{X}, t)]_j$ onto the symmetric subunit i and σ_i^{-1} equivalently permute symmetric gradient update to argree with order used to generate the seed geometry x_1. Then we average the symmetric $F(x, t)$ contribution together to obtain $[F(\mathbf{X}, t)]_i$. In practice, one can just pick i to compute $[F(\mathbf{X}, t)]_i$ and broadcast to symmetric subunits via G. Similarly, the noise can also be symmetrized by averaging:

$$[\mathbf{R}\,\mathrm{d}\bar{\mathbf{W}}_{\mathbf{sym}}]_i = \frac{\sqrt{|G|}}{|G|}(g_i G^{-1}) \cdot \mathbf{R}\,\mathrm{d}\bar{\mathbf{W}} \tag{14}$$

where $\sqrt{|G|}$ is used to correct the shrinking in the noise covariance because chain noise is obtained by averaging $|G|$ symmetric subunits.

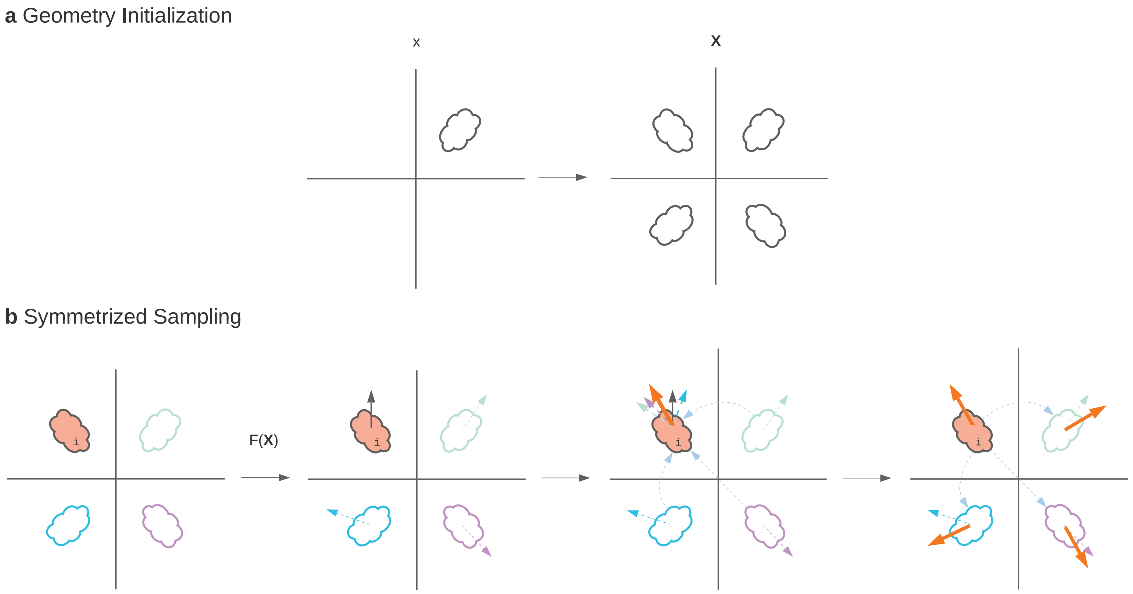
In summary, we proposed two methods of constructing $F_{sym.}(\cdot)$ and $\mathbf{R}dW$ and replace the sampling operation described in eq. (6) with the following modified SDE:

$$\begin{aligned} \mathbf{X_0} &= \mathtt{SDESOLVE}_{sym.}(\mathbf{X_1}) \\ &= \mathbf{X_1} + \int_{t=1}^{0} F_{sym}(\mathbf{X_t}, t)\,\mathrm{d}t + \int_{t=1}^{0} \sqrt{\beta_t}\,\mathrm{d}\bar{\mathbf{W}}_{\mathbf{sym}} \end{aligned} \tag{15}$$

The high-level algorithm is described in Figure 11 with an example that illustrates the C_4 symmetric sampling.

L.4 Additional symmetric samples

We include more generated samples for selected point groups including C_n (cyclic symmetry), D_n (dihedral symmetry), T (tetrahedral symmetry), O (octahedral symmetry), I (icosahedral symmetry). For each all the samples we use $\lambda_0 = 16$ and $\phi = 2$ with the Heun SDE solver that integrates from 1 to 0 for 400 steps. We use subunit k-NN sampling with $K = 6$. When $K > |G|$, we set $K = |G|$. We provide additional samples categorized by the imposed symmetry group in Figure 12 with a range of sequence lengths per subunit. Our method strictly imposes symmetries. However, the sampled geometries can sometimes show poor contact while still being symmetric. We provide such samples in Figure 13.



Supplementary Figure 11: (**a**) The protein complexes are initialized by performing symmetry operations on an initial protein. (**b**) $F(\mathbf{X})$ and $\mathbf{R}\,\mathrm{d}\bar{\mathbf{W}}$ are symmetrized by averaging $[F(\mathbf{X})]_j$ at symmetric positions.

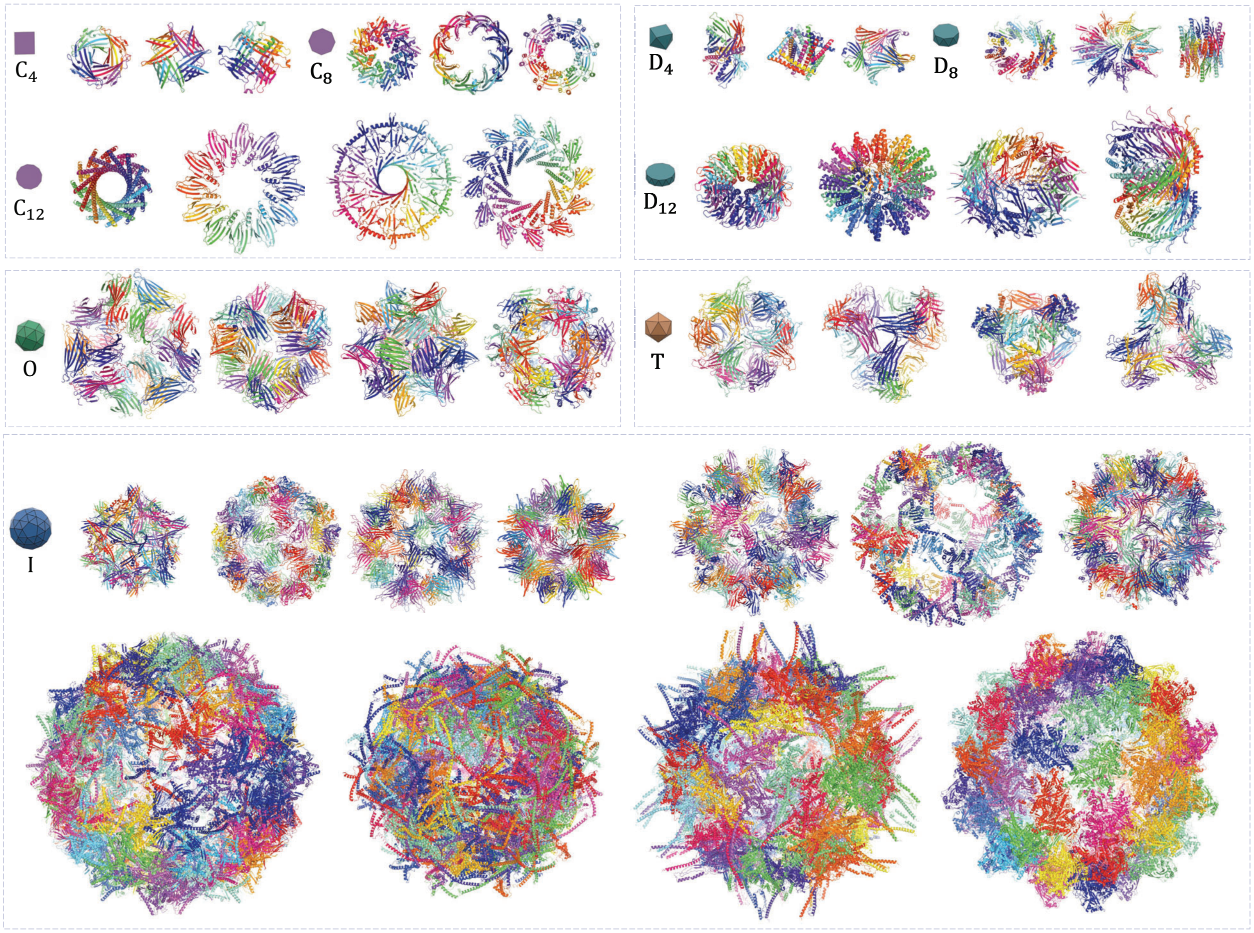
M Programmability: Shape

M.1 Motivation

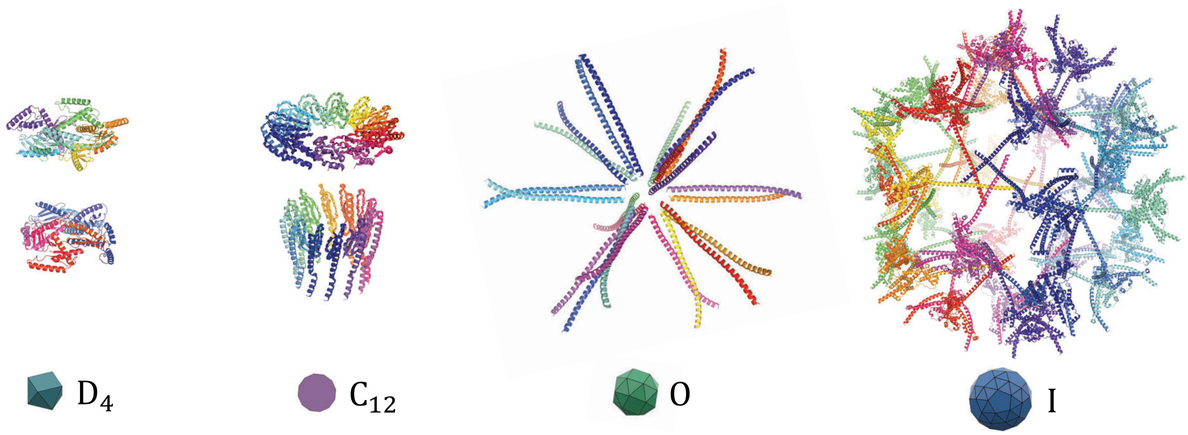
Proteins often realize particular functions through particular shapes, and consequently being able to sample proteins subject to generic shape constraints would seem to be an important tool for fully realizing the potential of protein design. Pores allow molecules to pass through biological membranes via a doughnut shape, scaffolding proteins spatially organize molecular events across the cell with precise spacing and interlocking assemblies, and receptors on the surfaces of cells interact with the surrounding world through precise geometries. Here we aim to explore and test generalized tools for conditioning on volumetric shape specifications with Chroma.

M.2 Approach

Our shape conditioning approach is based on Optimal Transport [Peyré et al., 2019], which provides tools for identifying correspondences and geometric distances between objects, such as the atoms in a protein backbone and a point cloud sampled from a target shape. We leverage two metrics from the optimal transport theory: (i) the Wasserstein distance [Peyré et al., 2019], which can measure the correspondence between point clouds in absolute 3D space and (ii) the Gromov-Wasserstein distance, which can measure the correspondences between objects in different domains by comparing their intra-domain distances or dissimilarities. Because it leverages *relational* comparisons, Gromov-Wasserstein can measure correspondences between unaligned objects with different sturctures and dimensionalities such as a skeleton graph and a 3D surface [Solomon et al.,



Supplementary Figure 12: Additional generated complexes grouped based on imposed symmetry groups.



Supplementary Figure 13: The generated complexes can form a poor protein-protein interface while still respecting imposed symmetry.

2016] or even between unsupervised word embeddings in two different languages [Alvarez-Melis and Jaakkola, 2018].

We initially experimented with adding heuristic gradients to the diffusion based on just the Wasserstein distance (estimated with the Sinkhorn algorithm [Peyré et al., 2019]), but found that the huge degeneracy in potential volume-filling conformations would often lead to jammed or high-contact-order solutions. While long-run Langevin sampling might help to allow gentle annealing into a satisfactory configuration in principle , we sought to accelerate convergence by breaking this degeneracy with a very coarse "space-filling plan" for how the fold should map into the target point cloud, which the prior can then realize with a specific protein backbone.

Mapping 1D to 3D We can leverage Gromov-Wasserstein (GW) optimal transport to answer the question "How would a protein with *ideal* distance scaling and a given length fill space in a target 3D volume?". To do so, we (i) built an idealized distance matrix for a protein based on the scaling law[3] $D_{ij} = 7.21 \times |i-j|^{0.32}$, (ii) compute the distance matrix for our target shape, and (iii) solve for the Gromov-Wasserstein optimal transport given these two distance matrices [Peyré et al., 2019] yielding a coupling matrix $K_{\text{GromovWasserstein}}$ with dimensionality $N_{\text{atoms}} \times N_{\text{points}}$. This coupling map sums to unity and captures the correspondence between each atom in the abstract protein chain and each point in the target point cloud. We use a small amount of entropy regularization to solve the optimal transport problem.

Optimal Transport loss In the inner loop of sampling, we can combine the GromovWasserstein coupling with simple Wasserstein couplings as a form of regularization towards our fold "plan". Our final loss is then

$$\text{ShapeLoss}(\mathbf{x},\mathbf{r}) = \sum_{i,j} \left(K_{ij}^{\text{GW}} + K_{ij}^{\text{W}}(\mathbf{x},\mathbf{r}) \right) \|\mathbf{x}_i - \mathbf{r}_j\|$$

where we compute the Wasserstein optimal couplings K_{ij}^{W} with the Sinkhorn algorithm [Peyré et al., 2019]. This yields a fast, differentiable loss that can be used directly for sampling.

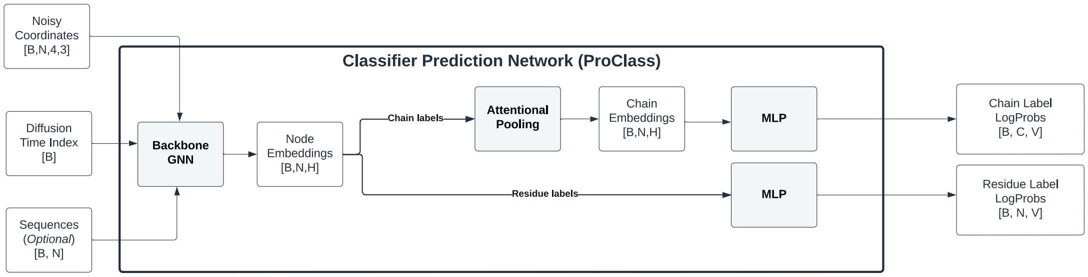
Generating 3D shapes We rendered letters and numbers from the English alphabet in the Liberation Sans font, extruded these 2D images into 3D volumes, and then sampled isotropic point clouds from these volumes.

N Programmability: Residue, Domain, and Complex-level Classification

N.1 Model Inputs

Noised backbone coordinates obtained from the PDB are passed as input to the model, along with a scalar $0 < t < 1$ denoting the time during diffusion (indexed between zero and one) that the noise was sampled at. The model optionally can consume sequence information if available.

[3] This scaling law was fit on a large single-domain protein 6HYP.


Noisy Coordinates [B,N,4,3]
Diffusion Time Index [B]
Sequences (Optional) [B, N]
Classifier Prediction Network (ProClass)
Backbone GNN
Node Embeddings [B,N,H]
Chain labels
Attentional Pooling
Chain Embeddings [B,N,H]
MLP
Residue labels
MLP
Chain Label LogProbs [B, C, V]
Residue Label LogProbs [B, N, V]

Supplementary Figure 14: **ProClass model architecture.**

N.2 Featurization

The time component is encoded with a random fourier featurization (e.g., see Tancik et al. [2020]). Provided sequence is encoded with a learnable embedding layer of amino acid identity. Backbone coordinates are passed to our `ProteinFeatureGraph` that extracts 2-mer and chain-based distances and orientations. These components are summed and passed to the neural network.

N.3 Architecture

The encoder is a message passing neural network. The graph is formed by taking K=20 nearest neighbors and sampling additional neighbors from a distribution according to a random exponential method.

Node and edge embeddings are passed to each layer, with each node being updated by a scaled sum of messages passed from neighbors. The message passed from node i to node j is obtained by stacking the embeddings at node i, those at node j, and $\mathcal{E}$, and passing these to a multi-layer perceptron (1 hidden layer). Edges are updated similarly. Each layer also applies layer normalization (along the channel dimension) and dropout (dropout probability=0.1).

After processing by the MPNN, node embeddings are passed to a different classification head for each label. If a head corresponds to a chain-level label, residues from each chain are pooled using an attentional pooling layer. The resulting embeddings are then passed to an MLP with 1 hidden layer to output logits for each label.

N.4 Labels and loss functions

The model is trained to predict the following labels: CATH, PFAM, Funfam, Organism, Secondary Structure, Interfacial Residue. The loss for predicting each label is quantified using cross entropy loss, and all components are summed and weighted equally.

N.5 Training

The model is trained for 50 epochs with an Adam optimizer [Kingma and Ba, 2014] with default momentum settings (betas=(0.9,0.999)), the learning rate is linearly annealed from 0 up to 0.0001 over the first 10,000 steps then kept constant. During training, first a time stamp $0 < t < 1$ is sampled uniformly, then noise is sampled from the globular covariance distribution, injected into the backbone coordinates, and fed to the model. Next, label predictions are made, loss are computed, and parameters are updated with the Adam optimizer.

N.6 Hyperparameters

The classification model has 4 layers, the size of node feature dimension is 512 and the edge feature dimension is 192, node update MLP has hidden dimension 256 with 2 hidden layers, and edge update MLP has hidden dimension 128 with 2 hidden layers.

O Programmability: Natural Language Annotations

O.1 Motivation

Recent advances in text-to-image diffusion models such as DALL-E 2 [Ramesh et al., 2022] and Imagen [Saharia et al., 2022] have produced qualitatively impressive results using a natural language interface. Given the open availability of pre-trained language models and a corpus of protein captions form large scientific databases such as the PDB [Berman, 2000] and UniProt [Consortium, 2020], we explore the possibility of creating a natural language interface to protein backbone generation. To do this, we build a protein captioning model (ProCap), which predicts $p(y|\mathbf{x}_t)$, where y is a text description of a protein and $\mathbf{x}_t$ is a noised protein backbone. This conditional model, when used in conjunction with the structural diffusion model presented in the main text, can be used as a text to protein backbone generative model.

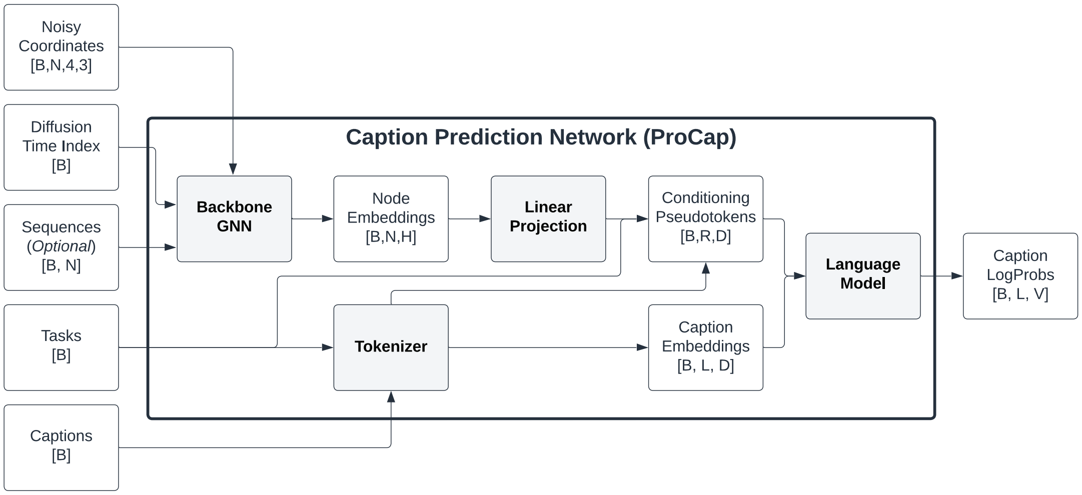
O.2 Dataset curation

To build a caption model, we begin by curating a paired dataset of protein structures and captions from both the PDB and UniProt databases. Caption information is collected for the structures used for the backbone diffusion model training, as well as the individual chains within these structures. For each structure, we use the PDB descriptive text as an overall caption. For each chain in a structure, we obtain a caption by concatenating all available functional comments from UniProt. Structures containing more than 1000 residues are not used, corresponding to a minority (10%) of all structures. The final set used to train and validate the caption model contains approximately 45 thousand captions, including those from both PDB and UniProt. Unlike for the backbone model, the splits used for training are completely random. The small size of the dataset constrained architecture choices to those with relatively few free parameters.

O.3 Model architecture

O.3.1 Architecture overview

To predict captions given noised structures, we construct ProCap using a pretrained language model and a pretrained protein encoder. The pretrained language model is the GPT-Neo 125 million parameter model [Black et al., 2021]. GPT-Neo was trained on the Pile [Gao et al., 2021]


Noisy Coordinates [B,N,4,3]
Diffusion Time Index [B]
Sequences (*Optional*) [B, N]
Tasks [B]
Captions [B]
Caption Prediction Network (ProCap)
Backbone GNN
Node Embeddings [B,N,H]
Linear Projection
Conditioning Pseudotokens [B,R,D]
Tokenizer
Caption Embeddings [B, L, D]
Language Model
Caption LogProbs [B, L, V]

Supplementary Figure 15: **ProCap model architecture.** ProCap connects a pretrained graph neural network encoder to an autoregressive language model trained on a large data corpus including scientific documents. We use the 125M parameter GPT-Neo as the language model, with internal dimension $D = 768$. Conditioning is achieved with pseudotokens generated from encodings of protein complex 3D backbone coordinates (batch size B, number of residues N, embedding dimension H) and a task token indicating whether a caption describes the whole complex or a single chain. The R relevant pseudotokens for each caption, consisting of the chain/structure residue tokens and the task token, are passed to the language model along with the caption. When used in the forward mode, ProCap can describe the protein backbone by outputting the probabilities of each word in the language model's vocabulary of size V for each of the L tokens of a caption. When used in conjunction with the prior model, it can be used for text to protein backbone synthesis. In training, ProCap uses a masked cross entropy loss applied only to the caption logits.

which contains articles from arXiv and Pubfed. Its choice is motivated to maximize the chance that the model would begin training with some understanding of protein-related text. We also use the pretrained graph neural network encoder from ProClass, the protein structure classification model introduced above, to encode protein backbones. Analogously to the choice of the language model, the purpose of the structure encoder is to start ProCap with semantic knowledge of protein structure. To condition the autoregressive language model, GPT-Neo, pseudotokens are formed from structures using the ProClass encoder and prepended to the caption as context, similar to [Lester et al., 2021].

O.3.2 Data embedding

Here, we describe the embedding of task, caption, and structure data into a shared tensor representation for input to the language model. Captions and task tokens are encoded using a modified version of the GPT-Neo tokenizer, whose vocabulary we augment with a special token to distinguish between prediction tasks involving single chains and those relating to entire structures. Structure inputs are converted into pseudotokens with the same shape as text embeddings through the graph neural network encoder of the pre-trained ProClass model. The task, structure, and caption embeddings are concatenated into a representation that is passed to the language model to obtain logits representing the probabilities of caption tokens. The model is trained on a standard masked cross entropy loss of the caption. The overall architectural flow is detailed in Fig. 15. We proceed to discuss the details of the embedding procedure.

Structure encoding in ProCap relies on a pretrained ProClass model. This classifier model consists of a GNN with multiple heads to extract different class information, as described previously. The GNN portion of the classifier network is used to obtain embeddings of each residue in the latent space of the classifier, with the intent that the pre-trained classifier weights should help ProCap learn the relationship between structures and captions. Besides the 3D information of the atoms in each structure, the diffusion timestep (noise level) is input to the GNN via a Fourier featurization layer which converts the diffusion time to a vector with the same dimension as the GNN node embedding space using randomly chosen frequencies between 0 and 16. To allow for ProCap to learn the optional use of sequence information, in 25% of the training data sequences are randomly passed along with structures. In these cases, the amino acid information for each residue is converted through a single embedding layer with output size equal to that of the GNN node embedding space dimension, then added to the time step vector.

Task tokens are added to the model to allow for captions of both single chain and full complex captions. For the prediction of UniProt captions describing single chains within structures, only the embeddings of the residues in the relevant chain are passed to the language model. For the prediction of the PDB captions related to entire structures, all residue embeddings are passed. In addition, a linear layer is added after the ProClass embeddings to go between the ProClass latent space and the embedding space of the language model, which are of different dimensionality. Finally, in order to help the model distinguish between PDB and UniProt prediction tasks, the encodings of the entire structures are each prepended with an embedding vector of a newly defined PDB marker token. We normalize the components of all structure vectors such that each one has zero mean and unit variance.

In summary, the ProCap architecture consists of a pre-trained GNN model for structure embedding and a pre-trained language model for caption embedding, with a learnable linear layer to interface between the two and a learnable language model head to convert the raw language model outputs to token probabilities.

O.4 Model training

We train ProCap to be compatible with conditional generation using the structural diffusion prior model. Like the other conditional models in this paper, each structure is noised according to the schedule of the structural diffusion model. During ProCap training, the graph neural network encoder weights from the pre-trained ProClass model are frozen. In addition, the internal weights of the GPT-Neo language model are also frozen, except for the head whose parameters are allowed to train. We choose to freeze these model parameters because of the relatively small training data size compared to that which was used to pre-train the language model. The language model head is allowed to learn, both to improve the embeddings of its usual tokens, as well as to optimize the encoding of new tokens. In training, we add a `<|PDB|>` task token to the GPT-Neo vocabulary to cue the model to predict whole complex captions from the PDB.

Training is conducted on a single V100 with a constant learning rate of 5×10^{-5} and the Adam optimizer with hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.999$. We evaluate loss on a validation set after every 2000 training examples. During training, we set an early stopping patience of 40 iterations, which was triggered after approximately 20 epochs at a cross entropy loss of 3.29.

O.5 Performance

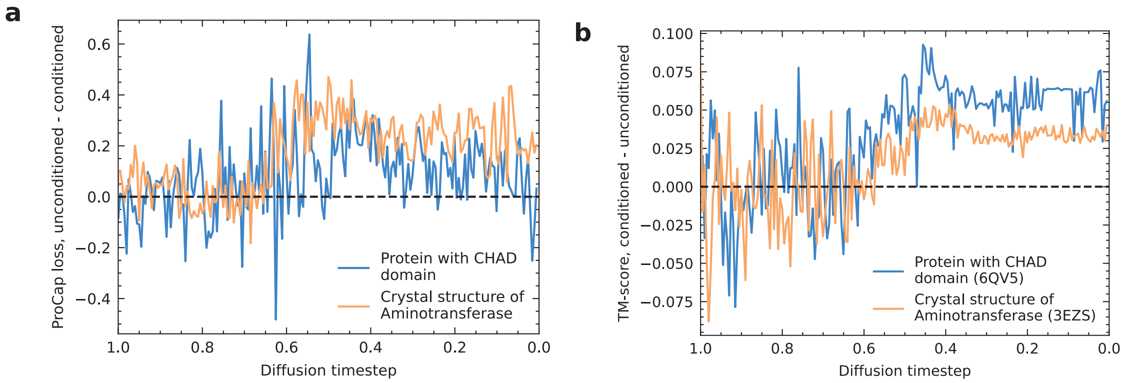
In order to test ProCap as a generative model, we draw high-quality conditional and corresponding unconditional low-temperature samples from the model. To that end, we employ a structural denoising approach in a similar fashion to the method described in [Song et al., 2021]. Specifically, the hybrid Langevin-reverse time SDE of Appendix B is used to evolve noisy random sample structures drawn from the diffusion model prior, with gradients of the ProCap loss with respect to structure added to the gradients of the structure diffusion model. When the size of the ProCap gradients is too small relative to those from the prior model, there is little appreciable difference between a caption-conditioned sample and an unconditional sample drawn from the same seed. We thus scale the ProCap gradients up by a factor of 100 and find that the resulting samples are better conditioned, analogously to previous work on classifier guidance [Dhariwal and Nichol, 2021]. Simultaneously, we observe that the sample quality decreases as ProCap gradients are scaled up further, resulting in the loss of secondary structure and even breakdown of backbone bond length constraints. To mitigate this effect, we limit the size of the gradient,

$$\nabla_{\mathbf{x}} \log p_t(\mathbf{y}|\mathbf{x}) \to \frac{\nabla_{\mathbf{x}} \log p_t(\mathbf{y}|\mathbf{x})}{|\nabla_{\mathbf{x}} \log p_t(\mathbf{y}|\mathbf{x})\,|} \min(|\nabla_{\mathbf{x}} \log p_t(\mathbf{y}|\mathbf{x})\,|, c_{\max})$$

with the choice $c_{\max} = 10$.

Examples of our generated samples are presented in the right two columns of Fig. 5. To evaluate ProCap model performance, we measure the improvement in caption loss during the SDE evolution between the unconditioned and conditioned samples. As an independent check, we also examine the gain in the TM-score between our sample (conditioned over unconditioned) and a target PDB structure which exemplifies the caption being used for conditioning. Finally, we analyze the generated structures visually for structural coherence. Qualitatively, starting from the same noisy random structure, the diffusion model yields denoised structures which demonstrate desirable characteristics including secondary structure elements, both with and without guidance from the caption model.

The caption loss and TM-score metrics for the sampling trajectories leading to the structures in Fig. 5 are shown in Fig. 16. Both are initially quite noisy, and the conditioned and unconditioned samples are equally likely at high t to have lower ProCap loss and/or better alignment with the target structure. However, over the course of the reverse diffusion, the effect of the conditioning is demonstrated in both panels. It is particularly notable that the TM-score is relatively stable at low t, indicating a regime where the SDE evolution is fine-tuning structural details rather than making significant changes.



Supplementary Figure 16: **ProCap evaluation metrics show effect of natural language conditioning compared to unconditioned samples from the same noised seed structure.** Panel **a** shows the caption model cross-entropy loss as a function of diffusion timestep, for two sample trajectories with and without the use of caption gradients. Panel **b** shows the TM-score between sampled structures and example structures from the PDB corresponding to the captions used for conditioning.