ProxelGen: Generating Proteins as 3D Densities

Felix Faltings [*1] **Hannes Stark** [*1] **Regina Barzilay** [1] **Tommi Jaakkola** [1]

Abstract

We develop ProxelGen, a protein structure generative model that operates on 3D densities as opposed to the prevailing 3D point cloud representations. Representing proteins as voxelized densities, or *proxels*, enables new tasks, conditioning capabilities, and a straightforward path for employing convolutional model architectures with different inductive biases than previous generative models. We generate proteins encoded as proxels via a 3D CNN-based VAE in conjunction with a diffusion model operating on its latent space. Compared to state-of-the-art models, ProxelGen's samples achieve higher novelty and better FID scores while maintaining designability of the training set. ProxelGen's advantages are demonstrated in a standard motif scaffolding benchmark, and we show how 3D density-based generation allows for more flexible shape conditioning.

1. Introduction

Recent advances in protein structure generation have explored a wide array of modeling choices, ranging from architectures to generative processes (Yim et al., 2024b). The focus of our work, however, is on equally fundamental yet unexplored aspect of structure generation: *protein representation*. Currently, the predominant representation is to express protein as sequences of geometric features, such as frames (Yim et al., 2023b;a; Bose et al., 2024), three-dimensional coordinates of the Cα (Lin et al., 2024; Geffner et al., 2025), or all atoms (Chu et al., 2023). This convergence on atomistic representations influences many downstream modeling choices ultimately limiting models' performance and capabilities.

In this paper, we explore an alternative representation of protein structure based on 3D densities called *proxels* (short for protein elements). Proxels are three-dimensional arrays, where each entry is the value of a function at a point in a three-dimensional grid, in the same way a pixel is the measured color intensity at a point in a two-dimensional grid. The proxels have multiple channels, each corresponding to a different function that encodes different information. For example, three of the channels sample Gaussian densities concentrated around the C, Cα, and N atoms in order to represent the protein backbone. To leverage this representation, we introduce ProxelGen, a latent diffusion model that directly operates on proxels. As shown in Figure 1, ProxelGen samples proteins encoded as proxels via a 3D CNN-based VAE in conjunction with a diffusion model operating on its latent space.

ProxelGen takes advantage of several features of proxels that distinguish it from previous work, enabling new tasks and conditioning capabilities:

- **Coarse-Graining** Proxels can be naturally coarse-grained by averaging across spatial positions. We leverage this by training ProxelGen in the spatially compressed latent space of a VAE, achieving a 512$\times$ compression in dimensions.
- **Convolutional Architecture** The VAE and flow model in ProxelGen use spatial convolution, imbuing them with inductive biases that are novel to the protein structure generation field. In comparison, most atomistic models use the same overarching transformer architectures operating either on frames, residues, atoms, or pairs of residues, and thus share many of the same inductive biases and disadvantages such as quadratic or cubic scaling in the length of the protein. On the other hand, ProxelGen scales with the resolution of the proxel grid, rather than the number of residues or atoms in the protein.
- **Spatial Conditioning** ProxelGen can naturally condition on spatial information, such as shapes or motifs specified as voxels or proxels, by directly concatenating to the model's input. For instance, proteins can be inpainted simply by masking out a region in space. For the same task, previous conditional generative models used for motif scaffolding not only require manual prespecification of the size of the protein, but also of how

[*]Equal contribution [1]CSAIL, Massachusetts Institute of Technology. Correspondence to: Felix Faltings <faltings@mit.edu>, Hannes Stark <hstark@mit.edu>.

Proceedings of the Workshop on Generative AI for Biology at the 42nd International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

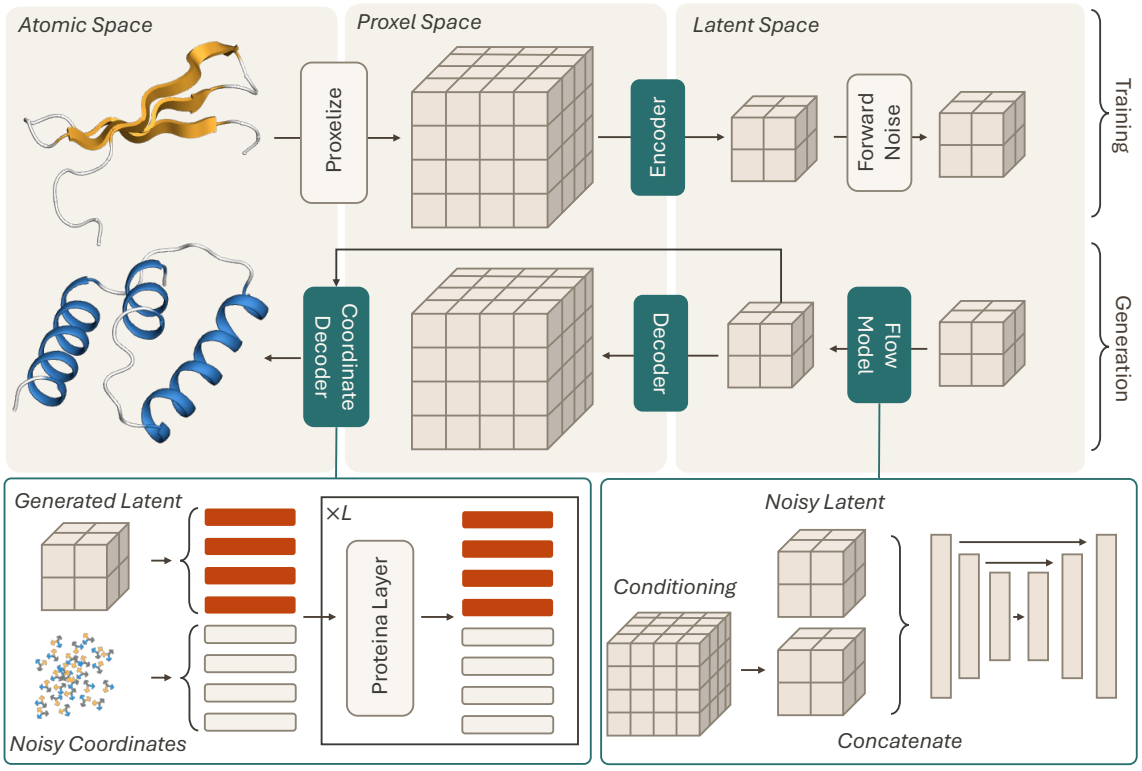

Figure 1. ProxelGen trains 3 separate models. First, an *autoencoder* that spatially compresses our proxel-based protein representation into latents. Second, a latent flow model trained to generate latent protein representations, which are obtained from the encoder that compresses a protein's proxel representation (encoder weights remain frozen while the flow is trained). Third, a coordinate decoder flow model that learns to decode latent voxels to 3D point clouds for applications where a point cloud protein representation is desired. At inference time, the two flow models are composed to sample protein structures by first generating latent proxel representations and then decoding them to 3D point clouds via the coordinate decoder flow model.

the designed residues map to the motif residues.

- **Fixed Dimensions** The dimensions of the proxels do not need to change as a function of the number of residues or chains in the protein as opposed to atomistic representations. ProxelGen thus does not require any specification of the length of the protein in advance. In contrast, the dimension of the atomistic representation is inherently tied to the size and composition of the protein, requiring the prespecification of the number of residues and chains. This issue only becomes worse in all-atom generative models, where the number of atoms is tied to the identity of the residues in the protein.

Empirically, ProxelGen's samples are more novel and show a better alignment to the training distribution than recent state-of-the-art models such as Proteina (Geffner et al., 2025), while maintaining the same level of designability as the proteins from the AFDB training data. Moreover, we demonstrate the advantages of ProxelGen's spatial voxel-based conditioning abilities. ProxelGen is able to generate structures based on arbitrarily specified 3D shapes, and in a standard motif scaffolding task, ProxelGen improves upon baselines in tasks where the motif topology is more complicated than a simple contiguous sequence. For example, ProxelGen finds 12 unique successful designs on a 4 segment motif, as opposed to a single successful design for all other methods.

2. Background

Protein Structure Generation. Deep learning approaches for protein structure generation aim to aid biological discovery by proposing solutions to conditional design tasks such as motif scaffolding (Yim et al., 2024a) and binder design (Watson et al., 2023). Toward this, a crucial benchmark has been unconditional protein structure generation, based on which several axes of model design have been explored. These include architectures (Lin and AlQuraishi, 2023; Lin et al., 2024; Geffner et al., 2025), generative modeling paradigms and processes (Yim et al., 2023a; Bose et al., 2024), and which geometric priors should be captured in the architecture (Wagner et al., 2024; Geffner et al., 2025)

or generative process (Yim et al., 2023b).

Meanwhile, alternative representations for protein structures remain largely unexplored. The main extent to which variations of protein representation have been considered are variations of 3D point clouds: geometric frames (Yim et al., 2023b), α-carbons (Lin et al., 2024; Geffner et al., 2025), and all-atom representations (Chu et al., 2023). We argue that considering a *density-based* protein representation opens the door for new types of conditioning and controllable generation, favorable inductive biases, and employing optimized and scalable architectures from image generation literature.

Density-based Generative modeling. With 3D density-based modeling, we refer to representing proteins via one or multiple functions $f : \mathbb{R}^3 \mapsto \mathbb{R}$ that assign a density to each point in 3D space. Such fields could, e.g., be the electron density or an artificially constructed density to capture the protein's shape and properties. To represent them computationally, we can discretize them into 3D grids of voxels. Operating on densities introduces a new set of possibly advantageous prior knowledge, similar to how operating on 3D point cloud representations imbues models with inductive biases that may be helpful for reasoning about biomolecules (a bias toward reasoning about distances and underlying atomic interactions). Reasoning about densities may be a further step in this direction, in that modeling electron densities can be considered closer to modeling the underlying physical interactions than reasoning about atoms.

Employing this richer density representation as a generative modeling target has also proven fruitful in small molecule generation (Pinheiro et al., 2024a; Zhang et al., 2024; Pinheiro et al., 2024b; Dumitrescu et al., 2025). In contrast, ProxelGen explores the use of such representations for proteins for the first time. Compared to small molecules, proteins present unique challenges due to their larger size and their chain-like nature which forms an important implicit constraint on the densities.

Latent Diffusion and Flow Models. Diffusion or flow models (Song et al., 2020; Liu et al., 2022; Lipman et al., 2022; Albergo and Vanden-Eijnden, 2022; Albergo et al., 2023) are a class of generative models that parameterize a time-dependent vector field $v_{\theta,t}$, which can be integrated to evolve noise from a chosen prior distribution $\mathbf{x}_0 \sim p_0$ to samples from the desired data distribution $\mathbf{x}_1 \sim p_1$. Latent diffusion approaches train two models and compose them at inference time to obtain a generative model: a variational autoencoder (VAE) (Kingma and Welling, 2022) and a flow model trained to generate objects from the VAE's latent space.

Protein structure generation has struggled to obtain benefits from latent diffusion or flow approaches (Fu et al., 2023; McPartlon et al., 2024; Lu et al., 2025a; Yim et al., 2025), possibly due to their use of 3D atomistic representations that change their dimensionality with the number of atoms and are hard to compress into a favorable fixed-size latent space. Operating on densities discretized as 3D voxel grids instead allows ProxelGen to leverage modern, highly optimized deep learning machinery developed for 2D pixel grids (images).

3. Method

3.1. Proxel Representation

We represent proteins as *proxel* (protein element) arrays: multi-channel 3D arrays $P \in \mathbb{R}^{C\times H\times W\times D}$, which can be seen as a discrete sampling of C continuous functions on a regular grid of points in 3D. We use the different channels to represent different information about the protein.

Backbone Channels We use 3 channels to encode the positions of the C, Cα and N backbone atoms. Taking the Cα channels as an example, let $x_1, \ldots, x_N \in \mathbb{R}^3$ be the atom coordinates. This is represented as a singular density,

$$f(x) = \sum_{i=1}^{N} \delta_{x_i}(x). \tag{1}$$

In order to expand the support of this density, it is convolved with a Gaussian kernel to give,

$$g(x) = (G * f)(x) = \sum_{i=1}^{N} G(x - x_i), \tag{2}$$

which is then sampled on a discrete grid,

$$P^{\mathrm{C}\alpha}_{ijk} = g(x_{ijk}), \tag{3}$$

with grid points given by,

$$x_{ijk} = ie_1 + je_2 + ke_3 + x_0, \quad i, j, k \in [H] \times [W] \times [D] \tag{4}$$

where x_0 is the lower left corner of the grid and e_1, e_2, e_3 are the standard basis vectors,

Bond Channel The fourth channel encodes the backbone bonds. For each pair of bonded backbone atoms $(x_i^1, x_i^2) \in \mathbb{R}^3 \times \mathbb{R}^3$, we place density at the midpoints $\bar{x}_i = (x_i^1 + x_i^2)/2$, convolve with a Gaussian, and sample on the grid as above:

$$P^{\mathrm{bond}}_{ijk} = g(x_{ijk}), \quad g(x) = \sum_i G(x - \bar{x}_i). \tag{5}$$

Chain Flow In protein design practice, the final produced protein descriptor is its sequence that can be synthesized and

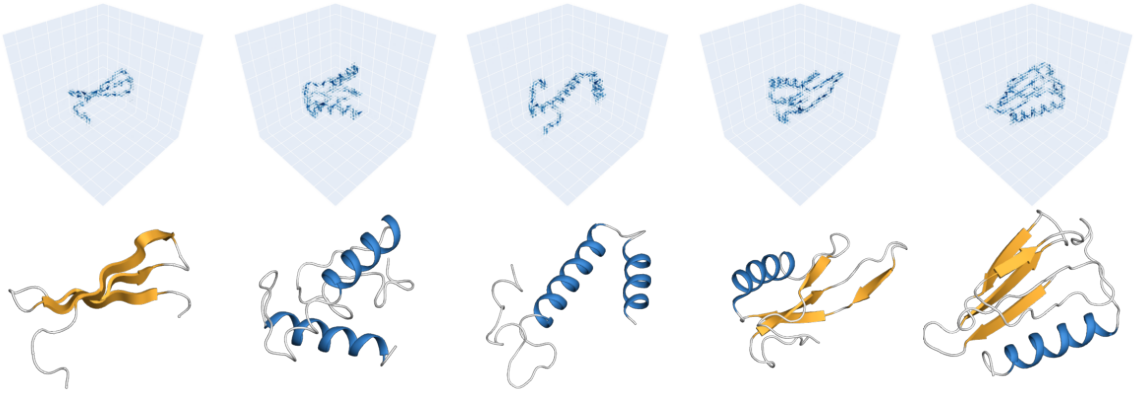

Figure 2. **Unconditional Samples**. Top row: chain flow of generated proxels. Bottom row: decoded atomic structures.

experimentally tested. For this purpose, we need to be able to decode our proxel representations to an *ordered* sequence of amino acids. Hence, proxels and their latent features need to carry information which we term the *chain flow* - information for each part of the represented density about which density follows it in the protein's chain ordering.

We encode this as a vector field that flows along the backbone from the N terminus to the C terminus. Concretely, let $v_1, \ldots, v_{N-1} \in \mathbb{R}^3$ denote the sequence of vectors connecting successive Cα positions in the backbone, $v_i = x_{i+1} - x_i$. Furthermore, let $\bar{x}_i = (x_{i+1} + x_i)/2$ denote the midpoints between Cα's. The vector field is then,

$$v(x) = \sum_{i=1}^{N} G(x - x_i) \cdot v_i. \tag{6}$$

Because the vector field takes values in $\mathbb{R}^3$, this accounts for three channels in the voxel representation.

Finally, note that as a practical consideration, this proxelization process can be done very fast. Each density is a sum of terms that can be processed in parallel, taking advantage of Single Instruction Multiple Data (SIMD). More details are given in the appendix.

3.2. Latent Flow Model

In order to handle large proxel arrays, we follow the approach in latent diffusion (Rombach et al., 2022; Vahdat et al., 2021) and learn a spatially compressed representation of the proxels. Concretely, we learn latents $L \in \mathbb{R}^{c \times h \times w \times d}$, where $h = H/f$, $w = W/f$, $d = D/f$ for some downsampling factor f and $c \neq C$ in general. We learn L using a β-VAE with an encoder E_ϕ which predicts the parameters of a Gaussian in latent space given proxels as input and a decoder D_ψ which produces proxels given a latent. Letting μ denote the data distribution of proxels P, the objective is

$$\mathcal{L}_{\text{AE}} = \mathbb{E}_{L \sim E_\phi(P), P \sim \mu}[\|D_\psi(L) - P\|_2^2] + \beta KL(E_\phi(P) \| \mathcal{N}(0, I)). \tag{7}$$

In practice, β is chosen to be very small in order to achieve a good reconstruction loss. Because of the sparsity of the proxels, we also found that the variances predicted by the encoder varied heavily between empty regions and regions occupied by the protein structure. This introduces an irregular geometry in the latent space, where the embedded data distribution is concentrated very tightly in some dimensions (regions occupied by the structure) and very diffuse in other dimensions (empty regions). To alleviate this, we force the distribution predicted by E_ϕ to have an identity covariance matrix. Note that this effectively reduces the KL penalty to an L_2 penalty on the predicted latents, preventing the latent distribution from spreading out too much, which can hinder downstream generative modeling.

After training the VAE, its weights are fixed, and we train a flow model to generate the latents L. We use a linear stochastic interpolant (Albergo et al., 2023) and train a model to predict the velocity,

$$\mathcal{L}_{\text{Flow}} = \mathbb{E}_{P \sim \mu, Z \sim \mathcal{N}, t \sim \nu}[\|s_\theta(Z_t) - (E_\phi(P) - Z)\|^2], \tag{8}$$

where $Z_t = t \cdot E_\phi(P) + (1 - t)Z$ and ν is a timestep distribution supported on $[0, 1]$. In practice, we found it helpful to bias this distribution towards earlier times. See appendix App. A.1 for details.

3.3. Conditional Generation

We also consider conditioning on spatially structured inputs, such as masked proxels or voxelized volumes. Concretely, let $I \in \mathbb{R}^{H \times W \times D}$ be the conditioning input. Because our

generative model operates in a spatially compressed latent space, we first map the input down to a similarly spatially compressed input I_L, using a condition encoder with the same architecture as the VAE encoder E_ϕ. The compressed input is then fed into the generative model by channel-wise concatenation, so that the model sees the part of the input relevant to each spatial location. The condition encoder is trained concurrently with the generative model, in contrast to the VAE encoder, which is pretrained separately.

3.4. Structure Recovery

Because atomic or frame representations of proteins are ubiquitous, we also consider how to decode from proxels back into a backbone representation. This allows us to, for example, use inverse folding models like ProteinMPNN to design sequences based on our generated structures and evaluate self-consistency RMSDs for comparison with previous methods.

To achieve this, we fine-tune an atomistic flow model to generate structures conditioned on proxels. Specifically, we finetune a small Proteina (Geffner et al., 2025) model to generate structures conditioned on generated proxels. To condition the model, we first tokenize the proxels by taking spatial patches, in the same way that images are tokenized in vision transformers. However, in order to reduce the number of tokens used to represent the proxels, we only tokenize the *latent* representation of the proxels. We also add a 3D sinusoidal positional embedding to the proxel tokens.

These latent proxel tokens are then injected into each layer of the model by concatenating them to the regular tokens that Proteina operates on, where we also allow the proxel tokens to be updated by each layer of the model. This method avoids the need to add any additional layers into the model, which we finetune from a unconditional Proteina checkpoint.

In order to avoid the need to finetune a much larger and more computationally demanding Proteina architecture, we opt to finetune a small 60M parameter version and then use a larger pretrained unconditional 200M parameter Proteina model to refine the decoded structures. Empirically, we found that this refinement step helps designability with only slight changes to the structures. We hypothesize that the small model may not fully respect local geometry such as bond lengths and angles leading in turn to poorly designed sequences due to ProteinMPNN's sensitivity to backbone geometry.

3.5. Self-Supervised Proxel Representations

In order to directly evaluate the proxels generated by our model, we opt to use the Frechet Inception Distance (Heusel et al., 2017), which compares the similarity of two distributions in a latent space (see Sec. 4.1 for more details). By comparing the samples generated by the model to the training set, we can assess how well it has learned the desired distribution. This metric has also recently been applied to evaluate protein generative models (Geffner et al., 2025; Faltings et al., 2025; Lu et al., 2025b). In order to adapt the FID to proxels, we learn self-supervised representations of proxels using contrastive learning. Specifically, we train a 3D ResNet using SimCLR (Chen et al., 2020), which we refer to as ProxCLR. See App. A.2 for more details.

4. Experiments

The goals of our experiments are to (1) demonstrate that generating realistic and diverse protein structures is possible using our proxel representation and (2) demonstrate new capabilities enabled by the representation. We thus first evaluate unconditional generation, followed by two applications to motif scaffolding and shape-conditioned generation.

4.1. Experimental Details

We first give general experimental details on inference settings and evaluation metrics. See Appendix A.1 for details on data processing, model architectures, and training.

Sampling Many protein structure generative models use some form of low-temperature sampling to optimize for designability, such as using a larger step scaling when integrating the flow field (Geffner et al., 2025), using a lower noise scale in the sampling SDE (Watson et al., 2023), or annealing rotations faster than translations in frame-based models (Bose et al., 2024). However, previous work (Faltings et al., 2025) suggests that optimizing for designability may not be desirable. Moreover, better unconditional designability may not necessarily translate to better performance on relevant tasks such as motif scaffolding. Instead, we tune our model towards the FID and find that moderate step size inflation improves the FID, but overly large ones (i.e., low temperatures) hurt the FID, likely because the model could drop parts of the distribution at low temperatures. Based on our previous observation in Sec. 3.2 that the early time steps are important for generation quality, we integrate the vector field with a smaller step size at early time steps so that more function evaluations are spent in the initial parts of generation.

Evaluation Metrics We use several metrics for evaluating generated proxels and protein structures.

- **FID** We evaluate the quality of generated proxels using the Frechet Inception Distance based on the self-supervised representations from our ProxCLR model Φ. Given a sample of proxels $P_1, \ldots, P_N$, and a

Sample	Designability ↑	Diversity ↑	Novelty ↓	FID ↓	α/β	Contacts
Native	39.45	232	0.71	1.89	34.31/14.99	29.19
Native Proxels	50.39	233	0.74	2.93	40.80/16.82	29.55
Proteina 200M (C)	97.77	127	0.87	20.47	69.11/7.08	15.77
Proteina 200M (H)	22.27	243	0.69	10.53	33.25/11.90	15.55
Proteina 400M (C)	97.66	114	0.88	16.68	66.45/8.99	21.07
Proteina 400M (H)	45.70	239	0.74	7.25	48.00/12.98	18.77
ProxelGen	53.13	233	0.73	6.05	41.56/16.05	24.54

Table 1. **Unconditional generation results** Low temperature Proteina samples marked (C) and regular temperature samples marked (H). α/β give the fraction of residues in alpha-helices/beta-sheets. Contacts gives the number of contacts of order greater than 0. Results computed on 256 samples.

reference set $\tilde{P}_1, \ldots, \tilde{P}_M$, we compute embeddings $\Phi(P_1), \ldots, \Phi(P_N)$ and $\Phi(\tilde{P}_1), \ldots, \Phi(\tilde{P}_M)$. The FID is the $\mathcal{W}_2$ distance between Gaussian approximations of the two sets of embedded proxels,

$$\text{FID} = \mathcal{W}_2(\mathcal{N}(\mu_1, \Sigma_1), \mathcal{N}(\mu_2, \Sigma_2), \tag{9}$$

where μ_1, Σ_1 and μ_2, Σ_2 are the mean and covariance matrices of $\Phi(P_1), \ldots, \Phi(P_N)$, and $\Phi(\tilde{P}_1), \ldots, \Phi(\tilde{P}_M)$ respectively.

- **Designability** For a given protein structure, we sample 8 sequences using ProteinMPNN, which are subsequently folded with ESMFold. The structure is considered designable if the folded structure of any one of the designed sequences has an RMSD to the original structure under 2Å.
- **Diversity** We compute the diversity of a set of structures by counting the number of foldseek clusters using a threshold of 0.5.
- **Novelty** The novelty of a structure is reported as the highest TMScore (Zhang and Skolnick, 2005) between the structure and any structure in the training set. Note that this implies that a lower novelty is better.
- **Higher Order Contacts** We also report the number of higher-order contacts in the structures as a measure of structural complexity. Two backbone positions in a structure form a contact if their Cα atoms are within 5Åof each other. The order of a contact is taken as the number of intervening secondary structures between the two residues. For example, a contact within the same secondary structure, such as between consecutive residues in an alpha helix, has an order of 0. We report the number of contacts of order greater than 0.
- **Secondary Structure Statistics** We also report the percentage of residues that occur in alpha helices and beta sheets.

4.2. Unconditional Generation

We evaluate the unconditional proxel samples from ProxelGen, as well as the structures reconstructed from the generated proxels. We compare our model against native structures and structures recovered from native proxels. As a baseline, we compare against the SOTA Proteina model (Geffner et al., 2025) using both high and low temperature sampling and two model sizes. For each method, we consider 256 samples. We see from the results in Table 4 that, compared to Proteina at low temperatures, ProxelGen achieves better FID, novelty, secondary structure distributions, and number of high-order contacts, while maintaining a designability close to the training data. Sampling from Proteina at regular temperatures improves novelty and FID, but leads to worse designability compared to ProxelGen. Overall, we see that ProxelGen is able to generate realistic proteins.

4.3. Inpainting and Motif Scaffolding

We next consider how ProxelGen can be used for motif scaffolding. Similar to previous methods, we frame this as an inpainting task. However, contrary to atomistic models, the input to the model is a masked region of space, rather than a masked region of the protein sequence as in atomistic models.

We finetune a model starting from an unconditional checkpoint using the strategy from (Lin et al., 2024) where we randomly crop protein sequences to extract motifs that are then proxelized. When generating scaffolds, we first conditionally generate proxels and then decode them into a structure as before. When refining the structure, we take advantage of a conditional Proteina model to more strictly preserve the structure of the motif region.

We evaluate on the RFdiffusion motif scaffolding benchmark from (Watson et al., 2023), consisting of 24 scaffolding tasks compiled from the protein design literature. For each of the 24 motifs, we design 1000 scaffolds. Each design is then scored by (1) generating 8 sequences with

		Free	Prespecified				Lengths	
Task Name	**Segments**	**ProxelGen**	**Proteina**	**Genie2**	**RFDiffusion**	**FrameFlow**	**Free**	**Prespecified**
6E6R	1	34					66.76	
long			713	415	381	110		108
medium			417	272	151	99		78
short			56	26	23	25		48
6EXZ	1	66					81.39	
long			290	326	167	403		110
medium			43	54	25	110		80
short			3	2	1	3		50
5TRV	1	33					92.94	
long			179	97	23	77		116
medium			22	23	10	21		86
short			1	3	1	1		56
7MRX	1	2					68.50	
128			51	27	66	35		128
85			31	23	13	22		85
60			2	5	1	1		60
1YCR	1	73	**249**	137	7	149	75.82	40-100
3IXT	1	**17**	8	14	3	8	73.88	50-75
5TPN	1	1	4	**8**	5	6	52.00	50-75
4ZYP	1	**22**	11	3	6	4	77.36	30-50
5WN9	1	1	2	1	0	**3**	23.00	35-50
1PRW	2	**2**	1	1	1	1	88.00	60-105
5IUS	2	0	1	1	1	0		57-142
2KL8	2	1	1	1	1	1	103.00	79
4JHW	2	0	0	0	0	0		60-90
1QJG	3	**51**	3	5	1	18	84.80	53-103
5YUI	3	2	**5**	3	1	1	116.50	50-100
1BCF	4	**12**	1	1	1	1	120.33	96-152

Table 2. **Unique Successes on 24 motif scaffolding tasks**. *Free* methods do not require the specification of where the motif appears in the design, as opposed to *Prespecified* ones. Lengths for free methods are averages of the successes. Lengths for prespecified methods are length ranges in the specification. Segments give the number of sequence-contiguous segments in the motif. The first four tasks contain the same motif with different length specifications.

ProteinMPNN and (2) folding the resulting sequences with ESMFold. The designed sequences preserve the original sequence of the motif. A design is considered a success if any of the folded structures of the 8 designed sequences have (1) a backbone RMSD to the original design less than 2Å (2) a motif RMSD less than 1Å (3) a pLDDT greater than 70 and (4) a pAE less than 5. Successes are clustered based on a TM Score cutoff of 0.6 to give the final number of unique successes. As baselines, we compare against Proteina, Genie2 (Lin et al., 2024), RFDiffusion (Watson et al., 2023), and FrameFlow (Yim et al., 2023a).

The results are presented in Table 4.1. Because our method treats scaffolding differently, two distinctions are in order. First, *free* methods denote ones that do not require a specification of where the motif should appear in the designed protein, as opposed to *prespecified* methods. Second, some tasks use the same motif but with different length specifications (top half), as opposed to some tasks that only give one length specification (bottom half). For the former type of task, we only report one number for ProxelGen.

ProxelGen delivers competitive performance in motif scaffolding, achieving many more unique successes in tasks that all previous methods do poorly on, such as 1BCF or 1QJG. It is not surprising that these two motifs are composed of

Subset	Precision ↑	Recall ↑	F1 ↑	TMScore ↓	Novelty ↓
All	88.55	88.69	88.60	60.21	78.28
Designable	89.26	89.36	89.28	64.14	81.77

Table 3. **Shape Conditioning Results** Precision, Recall, and F1 are computed based on the input shape and the shape of the generated proteins. TMScore is computed between the generated structure and the original structure used to create the input shape.

several disconnected segments, where the need to manually specify where each segment appears in the design is especially limiting for previous methods.

On the other hand, ProxelGen achieves fewer unique successes on other tasks, such as 6E6R. One possible explanation for the differences is the length of the designs, since longer designs are naturally more diverse, leading to more *unique* successes. Indeed, the results on the first four tasks are more comparable between rows with similar lengths. However, the success of ProxelGen on 1QJG and 1BCF cannot be explained by length. In Appendix B.1, we also discuss how the lack of motif-awareness in the atomic structure decoder also limits ProxelGen's performance, particularly on motifs.

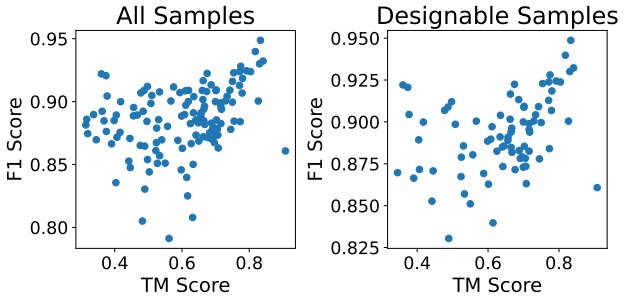

Figure 3. **Shape conditioning**: Samples with the highest TM Scores to the original structures have better shape adherence, but there are also many samples with good F1 but low TM score. Designable samples are also evenly distributed with no evident bias.

4.4. Shape Conditioning

In addition to motif scaffolding, we explore other forms of spatial conditioning enabled by proxels. Here we consider conditioning on arbitrary shapes, specified as binary voxel masks. This allows us to take arbitrary geometries as input, such as protein surfaces, and produce corresponding proteins.

We create training pairs of shapes and structures by generating coarse protein surfaces. We generate the surfaces by smoothing the proxels' Cα channel. We then sample this density on the same grid as the proxels and threshold at a single standard deviation to obtain a binary mask. As in motif scaffolding, we fine-tune a model starting from pretrained unconditional weights.

In order to evaluate the model, we consider how well it is able to generate novel proteins that conform to the same shape as existing proteins in the validation set. We evaluate a design by looking at the precision, recall, and F1 scores between the shape of the generated protein and the shape given as input. We also report the novelty and the TMScore to the original structure. As a baseline, we compared against Chroma, a structure-based model that is also able to condition on shapes.

In Table 4.3 we report ProxelGen's results, including both all samples as well as only the designable subset. We obtained relatively poor shape adherence using Chroma (F1 0.67), but given the difference between Chroma's shape conditioning and ours, the results may not be fully comparable, and we therefore omitted them from the table.

We see that ProxelGen generates proteins that adhere well to the input shape while being different from the original structures. In Fig. 3 we show that the shape adherence does not depend much on the TMScore to the original structure, indicating that ProxelGen is not simply recovering the original structures from their shapes. It is, however, possible that the model may have seen structures during training with very similar shapes despite having dissimilar structures. This is reflected in the relatively high TMScores to the training set (Novelty).

5. Discussion

We identified an axis of protein structure generative model development along which there is a distinct lack of exploration: determining the fundamentally important question of which protein representation is the most fit for generative modeling and which benefits may arise from alternatives to the unquestionedly prevailing 3D coordinate protein representations. We argue for a density-based protein representation that is more true-to-nature than 3D point clouds in that it can correspond to modeling the underlying density rather than a coarse-grained atom representation. Operating on discretized voxel grids of such densities admitted using strong, established generative modeling machinery from the image generation literature, resulting in our ProxelGen model. Evaluating ProxelGen in a series of unconditional and conditional generation benchmarks showed its superior performance along several metrics while demonstrating

strengths orthogonal to coordinate-based methods and enabling new types of conditioning.

For the future of density-based protein structure generation, we envision training directly on electron densities instead of employing custom-constructed densities that are derived from coordinate representations. Such electron densities are more information-rich and closer to experiment than coordinate representations, which are only derived from electron densities in an additional model-building step. Such models, as well as the herein proposed ProxelGen, are most likely to be used for protein engineering campaigns with positive societal impacts.

References

Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

Michael Samuel Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *The Eleventh International Conference on Learning Representations*, 2022.

Inigo Barrio-Hernandez, Jingi Yeo, Jürgen Jänes, Milot Mirdita, Cameron L. M. Gilchrist, Tanita Wein, Mihaly Varadi, Sameer Velankar, Pedro Beltrao, and Martin Steinegger. Clustering predicted structures at the scale of the known protein universe. *Nature*, 622 (7983):637–645, Oct 2023. ISSN 1476-4687. doi: 10.1038/s41586-023-06510-w. URL https://doi.org/10.1038/s41586-023-06510-w.

Avishek Joey Bose, Tara Akhound-Sadegh, Guillaume Huguet, Kilian Fatras, Jarrid Rector-Brooks, Cheng-Hao Liu, Andrei Cristian Nica, Maksym Korablyov, Michael Bronstein, and Alexander Tong. Se(3)-stochastic flow matching for protein backbone generation, 2024. URL https://arxiv.org/abs/2310.02391.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PmLR, 2020.

Alexander E. Chu, Lucy Cheng, Gina El Nesr, Minkai Xu, and Po-Ssu Huang. An all-atom protein generative model. *bioRxiv*, 2023. doi: 10.1101/2023.05.24.542194. URL https://www.biorxiv.org/content/early/2023/05/25/2023.05.24.542194.

Alexandru Dumitrescu, Dani Korpela, Markus Heinonen, Yogesh Verma, Valerii Iakovlev, Vikas Garg, and Harri Lähdesmäki. E(3)-equivariant models cannot learn chirality: Field-based molecular generation, 2025. URL https://arxiv.org/abs/2402.15864.

Felix Faltings, Hannes Stark, Tommi Jaakkola, and Regina Barzilay. Protein fid: Improved evaluation of protein structure generative models, 2025. URL https://arxiv.org/abs/2505.08041.

Cong Fu, Keqiang Yan, Limei Wang, Wing Yee Au, Michael McThrow, Tao Komikado, Koji Maruhashi, Kanji Uchino, Xiaoning Qian, and Shuiwang Ji. A latent diffusion model for protein structure generation, 2023. URL https://arxiv.org/abs/2305.04120.

Tomas Geffner, Kieran Didi, Zuobai Zhang, Danny Reidenbach, Zhonglin Cao, Jason Yim, Mario Geiger, Christian Dallago, Emine Kucukbenli, Arash Vahdat, and Karsten Kreis. Proteina: Scaling flow-based protein structure generative models, 2025. URL https://arxiv.org/abs/2503.00710.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022. URL https://arxiv.org/abs/1312.6114.

Yeqing Lin and Mohammed AlQuraishi. Generating novel, designable, and diverse protein structures by equivariantly diffusing oriented residue clouds, 2023. URL https://arxiv.org/abs/2301.12485.

Yeqing Lin, Minji Lee, Zhao Zhang, and Mohammed AlQuraishi. Out of many, one: Designing and scaffolding proteins at the scale of the structural universe with genie 2, 2024. URL https://arxiv.org/abs/2405.15489.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Amy X. Lu, Wilson Yan, Sarah A. Robinson, Simon Kelow, Kevin K. Yang, Vladimir Gligorijevic, Kyunghyun Cho, Richard Bonneau, Pieter Abbeel, and Nathan C. Frey. All-atom protein generation with latent diffusion. *bioRxiv*, 2025a. doi: 10.1101/2024.12.02.626353. URL https://www.biorxiv.org/content/early/2025/02/13/2024.12.02.626353.

Tianyu Lu, Melissa Liu, Yilin Chen, Jinho Kim, and Po-Ssu Huang. Assessing generative model coverage of protein structures with shapes. *bioRxiv*, pages 2025–01, 2025b.

Matt McPartlon, Céline Marquet, Tomas Geffner, Daniel Kovtun, Alexander Goncearenco, Zachary Carpenter, Luca Naef, Michael M. Bronstein, and Jinbo Xu. Bridging sequence and structure: Latent diffusion for conditional protein generation, 2024. URL https://openreview.net/forum?id=DP4NkPZOpD.

Pedro O. Pinheiro, Arian Jamasb, Omar Mahmood, Vishnu Sresht, and Saeed Saremi. Structure-based drug design by denoising voxel grids, 2024a. URL https://arxiv.org/abs/2405.03961.

Pedro O. Pinheiro, Joshua Rackers, Joseph Kleinhenz, Michael Maser, Omar Mahmood, Andrew Martin Watkins, Stephen Ra, Vishnu Sresht, and Saeed Saremi. 3d molecule generation by denoising voxel grids, 2024b. URL https://arxiv.org/abs/2306.07473.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022. URL https://arxiv.org/abs/2112.10752.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space, 2021. URL https://arxiv.org/abs/2106.05931.

Mihaly Varadi, Damian Bertoni, Paulyna Magana, Urmila Paramval, Ivanna Pidruchna, Malarvizhi Radhakrishnan, Maxim Tsenkov, Sreenath Nair, Milot Mirdita, Jingi Yeo, Oleg Kovalevskiy, Kathryn Tunyasuvunakool, Agata Laydon, Augustin Žídek, Hamish Tomlinson, Dhavanthi Hariharan, Josh Abrahamson, Tim Green, John Jumper, Ewan Birney, Martin Steinegger, Demis Hassabis, and Sameer Velankar. Alphafold protein structure database in 2024: providing structure coverage for over 214 million protein sequences. *Nucleic Acids Research*, 52(D1):D368–D375, 11 2023. ISSN 0305-1048. doi: 10.1093/nar/gkad1011. URL https://doi.org/10.1093/nar/gkad1011.

Simon Wagner, Leif Seute, Vsevolod Viliuga, Nicolas Wolf, Frauke Gräter, and Jan Stühmer. Generating highly designable proteins with geometric algebra flow matching, 2024. URL https://arxiv.org/abs/2411.05238.

Joseph L. Watson, David Juergens, Nathaniel R. Bennett, Brian L. Trippe, Jason Yim, Helen E. Eisenach, Woody Ahern, Andrew J. Borst, Robert J. Ragotte, Lukas F. Milles, Basile I. M. Wicky, Nikita Hanikel, Samuel J. Pellock, Alexis Courbet, William Sheffler, Jue Wang, Preetham Venkatesh, Isaac Sappington, Susana Vázquez Torres, Anna Lauko, Valentin De Bortoli, Emile Mathieu, Sergey Ovchinnikov, Regina Barzilay, Tommi S. Jaakkola, Frank DiMaio, Minkyung Baek, and David Baker. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976): 1089–1100, Aug 2023. ISSN 1476-4687. doi: 10.1038/

s41586-023-06415-8. URL https://doi.org/10.1038/s41586-023-06415-8.

Jason Yim, Andrew Campbell, Andrew Y. K. Foong, Michael Gastegger, José Jiménez-Luna, Sarah Lewis, Victor Garcia Satorras, Bastiaan S. Veeling, Regina Barzilay, Tommi Jaakkola, and Frank Noé. Fast protein backbone generation with se(3) flow matching, 2023a. URL https://arxiv.org/abs/2310.05297.

Jason Yim, Brian L. Trippe, Valentin De Bortoli, Emile Mathieu, Arnaud Doucet, Regina Barzilay, and Tommi Jaakkola. Se(3) diffusion model with application to protein backbone generation, 2023b. URL https://arxiv.org/abs/2302.02277.

Jason Yim, Andrew Campbell, Emile Mathieu, Andrew Y. K. Foong, Michael Gastegger, José Jiménez-Luna, Sarah Lewis, Victor Garcia Satorras, Bastiaan S. Veeling, Frank Noé, Regina Barzilay, and Tommi S. Jaakkola. Improved motif-scaffolding with se(3) flow matching, 2024a. URL https://arxiv.org/abs/2401.04082.

Jason Yim, Hannes Stärk, Gabriele Corso, Bowen Jing, Regina Barzilay, and Tommi S. Jaakkola. Diffusion models in protein structure and docking. *WIREs Computational Molecular Science*, 14(2): e1711, 2024b. doi: https://doi.org/10.1002/wcms.1711. URL https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wcms.1711.

Jason Yim, Marouane Jaakik, Ge Liu, Jacob Gershon, Karsten Kreis, David Baker, Regina Barzilay, and Tommi Jaakkola. Hierarchical protein backbone generation with latent and structure diffusion, 2025. URL https://arxiv.org/abs/2504.09374.

Odin Zhang, Jieyu Jin, Zhenxing Wu, Jintu Zhang, Po Yuan, Haitao Lin, Haiyang Zhong, Xujun Zhang, Chenqing Hua, Weibo Zhao, Zhengshuo Zhang, Kejun Ying, Yufei Huang, Huifeng Zhao, Yuntao Yu, Yu Kang, Peichen Pan, Jike Wang, Dong Guo, Shuangjia Zheng, Chang-Yu Hsieh, and Tingjun Hou. Ecloudgen: Leveraging electron clouds as a latent variable to scale up structure-based molecular design. *bioRxiv*, 2024. doi: 10.1101/2024.06.03.597263. URL https://www.biorxiv.org/content/early/2024/12/26/2024.06.03.597263.

Yang Zhang and Jeffrey Skolnick. Tm-align: a protein structure alignment algorithm based on the tm-score. *Nucleic Acids Research*, 33(7):2302–2309, 01 2005. ISSN 0305-1048. doi: 10.1093/nar/gki524. URL https://doi.org/10.1093/nar/gki524.

A. Additional Technical Details

A.1. Additional Experiment Details

Timestep Training Distribution We empirically probed the importance of different timesteps by taking data samples, flowing them to a time t and then denoising them with our generative model from that point. We found a sharp drop off in FID around $t = 0.3$ indicating that the model is easily able to generate good samples provided the intermediate samples at $t = 0.3$ are good. We therefore oversampled timesteps around $t = 0.3$ by sampling from a mixture of a uniform distribution on $[0, 1]$ and a truncated normal centered at $t = 0.2$ with standard deviation $\sigma = 0.1$.

Training Details We use $32 \times 32 \times 32$ proxel arrays with the 7 channels described above. The discrete sampling points of the proxel grid are spaced 1.5Å apart. Given that the average distance between two consecutive Cα atoms is around 3Å this ensures that two atoms are rarely placed in the same proxel cell. The Gaussian kernel we use when constructing the proxels has a standard deviation of 1Å.

We train on the FoldSeek (Barrio-Hernandez et al., 2023) clustered AlphaFoldDB (Varadi et al., 2023) dataset used in (Lin et al., 2024), keeping only structures that fit completely on our proxel grid, leaving 98,584 structures. We randomly split this into 93,654 training structures and 4,930 validation structures.

Given the mixed advantages and disadvantages of rotation-equivariant convolutional architectures, we opt to use regular non-equivariant convolutions. We instead axis-align all the structures we train on so that their principal components align to the axes of a fixed coordinate system centered at their centers of mass. In order to avoid learning spurious biases from this procedure, we compute the alignment using perturbed structures so that, for example, nearly spherical structures are randomly aligned.

The VAE is implemented using a 3D convolutional architecture with self-attention based on the architecture used in (Rombach et al., 2022). The VAE was trained with a KL weight of 10^{-6} a learning rate of 10^{-4} and a weight decay of 10^{-4}. The generative model is implemented as a 3D UNet architecture, also with self-attention, based on the architecture from (Song et al., 2020). We also tried a transformer architecture but the performance was much worse. The model was trained with a learning rate of 10^{-4}. Models were trained on single NVIDIA A6000 GPUs until relevant metrics converged (validation loss for the VAE, FID for the generative model).

A.2. Contrastive Learning Details

Let Φ denote our embedding model. Let $P_1, \ldots, P_B$ denote a batch of proxelized proteins, and let z_{2i}, z_{2i-1} denote two augmented views of P_i. In our case, we use random cropping, rotation, and translation as augmentations. Given the cosine similarities,

$$s_{i,j} = \frac{\Phi(z_i)^T \Phi(z_j)}{\|\Phi(z_i)\| \|\Phi(z_j)\|}, \tag{10}$$

we define the loss for pair i, j

$$l(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{B} \mathbb{I}_{k \neq i} \exp(s_{i,k}/\tau)}. \tag{11}$$

The total loss for a batch is then,

$$\mathcal{L} = \frac{1}{2B} \sum_{k=1}^{B} [l(2k, 2k-1) + l(2k-1, 2k)]. \tag{12}$$

We learn Φ using stochasic gradient descent following gradients of $\mathcal{L}$.

B. Additional Results

B.1. Motif Scaffolding

Table 4 gives a further breakdown of ProxelGen's performance on the motif scaffolding tasks. We note that because the coordinate decoder model is not conditioned on the motif it can fail to respect the motif topology and thread the protein chain through the motif in a different way. We thus perform a first filtering of the 1000 designs from ProxelGen that checks that the motif is properly preserved. We see that this is actually a big limiter on ProxelGen's performance, particularly on

larger motifs where the chance of violating the motif topology is greater. Thus performance could be greatly improved by improving the atomic decoding to be aware of the motif, though we leave this for future work.

Table 4. Additional motif scaffolding details

motif	segments	mean len	motif len	unique successes	total successes	prefilter successes
6e6r	1	66.76	13	34	37	311
6exz	1	81.39	15	66	93	315
5trv	1	92.94	21	33	46	204
7mrx	1	68.50	22	2	5	94
1ycr	1	75.82	9	73	77	208
3ixt	1	73.88	24	17	25	184
5tpn	1	52.00	19	1	4	39
4zyp	1	77.36	15	22	26	132
5wn9	1	23.00	20	1	5	133
1prw	2	88.00	40	2	5	9
5ius	2	NaN	42	0	0	0
2kl8	2	103.00	59	1	4	15
4jhw	2	NaN	24	0	0	116
1qjg	3	84.80	3	51	54	565
5yui	3	116.50	11	2	5	173
1bcf	4	120.33	32	12	91	510