ENZYSHELL: DYNAMIC SHELL-GUIDED ENZYME BACKBONE GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Enzyme design requires generating protein backbones that scaffold catalytic motifs and accommodate substrates. However, directly generating full backbones from these functional constraints involves two distinct requirements: inferring a suitable local protein environment around the catalytic motif and substrate, and generating the full backbone to support this environment. To address this challenge, we propose EnzyShell, a dynamic shell-guided framework for enzyme backbone generation. We explicitly model this local environment as a functional shell comprising the motif and residues surrounding either the motif or the substrate. We then train two complementary flow models for full-backbone generation, conditioned on the substrate together with either the motif or the functional shell. During sampling, we dynamically construct the functional shell from backbone predictions of the motif-conditioned model, provide it to the shell-conditioned model, and combine their predicted vector fields to guide enzyme backbone generation. Evaluated on the EnzyBind benchmark spanning six EC classes, EnzyShell improves backbone designability, EC match rate, and predicted k_{cat} over the evaluated baselines, while maintaining comparable catalytic-motif geometry.

1 INTRODUCTION

Enzymes recognize substrates and catalyze reactions through precisely organized three-dimensional environments supported by the surrounding protein structure (Chu et al., 2024; Albanese et al., 2025). Designing such enzymes requires satisfying functional constraints while maintaining structural integrity (Vaissier Welborn & Head-Gordon, 2019). In particular, precise catalytic geometry is critical for function, as demonstrated experimentally in designed Kemp eliminases (Blomberg et al., 2013). Computational enzyme backbone design therefore requires preserving catalytic-motif geometry, accommodating the target substrate, and generating designable backbones (Braun et al., 2026; Ahern et al., 2026).

Diffusion- and flow-based models have substantially advanced protein backbone generation and motif scaffolding (Watson et al., 2023; Yim et al., 2023; Geffner et al., 2025; Didi et al., 2026). Recent methods have extended these generative approaches to enzyme design by incorporating catalytic reactions, substrates, side-chain conformations, and active-site information (Hua et al., 2024; Song et al., 2024; 2025; Butcher et al., 2025; Ahern et al., 2026). However, directly generating a full enzyme backbone from such functional constraints involves two distinct requirements: inferring a suitable local protein environment around the catalytic motif and substrate, and generating the full backbone around this environment. Residues in this local environment can play important roles in organizing catalytic sites and supporting substrate interactions (Deng & Cui, 2023). This motivates explicitly representing this local environment and using it to guide full-backbone generation.

For an enzyme backbone, we define a functional shell comprising the catalytic motif and residues surrounding either the motif or the substrate. The shell captures structural context around both the substrate and catalytic residues, including motif regions that may lie away from the substrate, and provides an explicit structural condition for full-backbone generation.

Building on this representation, we introduce **EnzyShell**, a dynamic shell-guided framework for motif- and substrate-conditioned enzyme backbone generation. We train two flow models for full-backbone generation: one is conditioned on the prescribed motif and substrate, and the other on a functional shell and the substrate. The motif-conditioned model infers the local protein environment

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

while generating the full backbone, whereas the shell-conditioned model uses this environment as an explicit structural condition to guide full-backbone generation. During sampling, the functional shell is dynamically constructed from backbone predictions of the motif-conditioned model and used to condition the shell-conditioned model. The two predicted vector fields are then combined to update a shared backbone trajectory.

We evaluate EnzyShell on 190 enzyme–substrate pairs from EnzyBind (Song et al., 2025), spanning six EC classes. Under a consistent evaluation setting, EnzyShell achieves 75% backbone designability, compared with 52% for the strongest evaluated external baseline, while maintaining comparable catalytic-motif geometry and generating structurally diverse successful designs. It also achieves a higher EC match rate and predicted k_{cat}, with ablations showing that dynamic shell-guided sampling outperforms either model alone and sequential two-model sampling.

Our contributions are summarized as follows:

- We propose EnzyShell, a dynamic shell-guided framework for motif- and substrate-conditioned enzyme backbone generation that explicitly represents the local protein environment around the catalytic motif and substrate as a functional shell.
- We introduce a dynamic shell-guided sampling strategy that combines two complementary full-backbone flow models, in which predictions from the motif-conditioned model dynamically construct the structural condition for the shell-conditioned model and their vector fields jointly guide a shared sampling trajectory.
- We demonstrate improved backbone designability, EC match rate, and predicted k_{cat} on EnzyBind while maintaining comparable catalytic-motif geometry, with ablations validating the contributions of dynamic shell-guided sampling.

2 Related Work

2.1 Computational Enzyme Design

Recent generative models incorporate structural and functional information into enzyme design. EnzyGen jointly generates enzyme sequences and structures conditioned on functionally important sites and substrates (Song et al., 2024). EnzyControl extends a pretrained motif-scaffolding model with substrate conditioning for enzyme backbone generation (Song et al., 2025). RFdiffusion2 (Ahern et al., 2026) and RFdiffusion3 (Butcher et al., 2025) enable atom-level active-site scaffolding and all-atom biomolecular design, respectively. Proteína-Complexa performs all-atom design in a learned latent space with inference-time optimization (Didi et al., 2026). Related methods have also incorporated local pocket environments into protein and enzyme design. PocketGen co-designs ligand-binding pocket sequences and structures (Zhang et al., 2024). GENzyme follows a staged pipeline that generates a catalytic pocket from a given reaction, builds the full enzyme conditioned on the pocket, and subsequently optimizes the enzyme–substrate complex (Hua et al., 2024). EnzyPGM jointly generates enzymes and substrate-binding pockets by modeling pocket-substrate interactions (Lin et al., 2026).

In contrast, EnzyShell explicitly represents the local environment around both the motif and substrate as a functional shell that is dynamically constructed during full-backbone generation.

2.2 Iterative Conditioning and Model Composition

Iterative generative models can incorporate intermediate predictions into subsequent sampling steps. Self-conditioning feeds previous predictions back into the model and has been widely used in diffusion-based generation, including in RFdiffusion for protein structure design (Chen et al., 2022; Watson et al., 2023). More recently, Dynamic Guidance (Triaridis et al., 2025) adapts the guidance target at each sampling step based on the current sample. A complementary line of work combines multiple model predictions during sampling. Classifier-free guidance (Ho & Salimans, 2022) combines conditional and unconditional predictions, with related guidance formulations developed for flow-matching models (Zheng et al., 2023). Composable Diffusion Models (Liu et al., 2022) combine separately conditioned diffusion models through an energy-based formulation for compositional

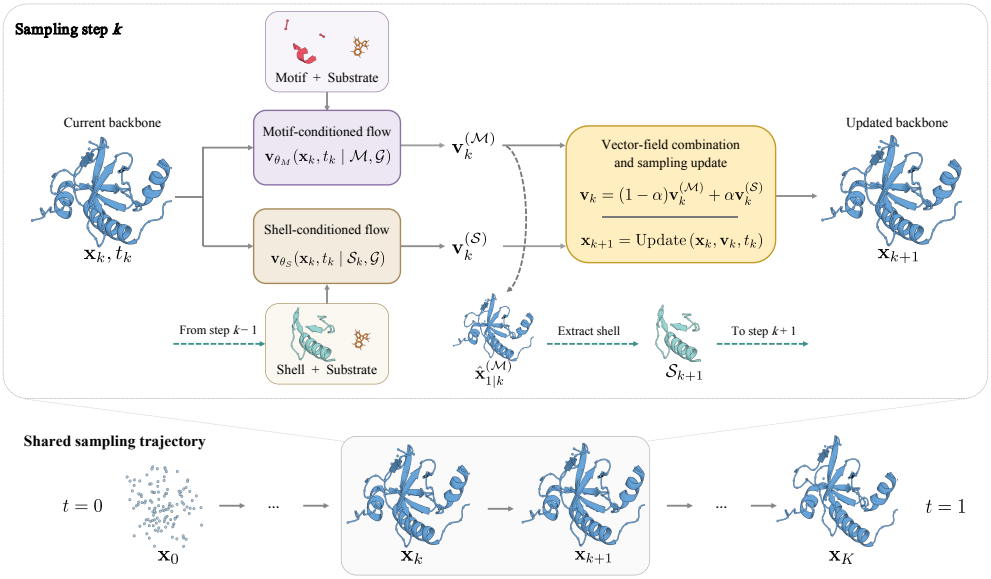

Figure 1: **Dynamic shell-guided sampling in EnzyShell. Top:** At sampling step k, the motif- and shell-conditioned flow models operate on the shared noisy backbone $\mathbf{x}_k$. The motif-conditioned backbone prediction is used to construct the functional shell for the next step, while the two predicted vector fields are combined to update the shared backbone trajectory. The shell used at step k is constructed from the preceding motif-conditioned prediction, with $\mathcal{S}_0 = \mathcal{M}$. Teal dashed arrows indicate shell transfer across sampling steps. **Bottom:** Repeating this procedure generates a full backbone from Gaussian noise along a single sampling trajectory.

generation, while CompFlow (Miglior et al., 2026) extends compositional inference to flow matching by combining conditional vector fields.

In EnzyShell, intermediate backbone predictions from the motif-conditioned flow dynamically construct the structural condition for the shell-conditioned flow, while both models predict full-backbone vector fields that jointly guide a shared sampling trajectory.

3 Method

As illustrated in Figure 1, EnzyShell is a dual-model framework that dynamically couples two conditional flow models during backbone generation. We introduce a functional shell to explicitly represent the local protein environment and train the two full-backbone flow models with different conditioning: one on the motif and substrate, and the other on the functional shell and substrate. We refer to these as the motif-conditioned model and the shell-conditioned model, respectively. During sampling, we use the vector field predicted by the motif-conditioned model to derive a clean backbone prediction, from which we dynamically construct the functional shell. We then provide the shell to the shell-conditioned model and combine the two predicted vector fields to guide a shared sampling trajectory.

3.1 Problem Formulation and Preliminaries

Problem Formulation. We represent an enzyme backbone of length N by its ordered C_α coordinates $\mathbf{x} = (x_1, \ldots, x_N) \in \mathbb{R}^{N\times 3}$, where $x_i \in \mathbb{R}^3$ denotes the C_α coordinate of the i-th residue. A catalytic motif consisting of N_m residues is specified by a set of residue positions $\mathcal{R}^m \subseteq \{1, \ldots, N\}$, and their target coordinates $\mathbf{x}^m = (x_1^m, \ldots, x_{N_m}^m) \in \mathbb{R}^{N_m\times 3}$. We denote the motif condition as $\mathcal{M} = (\mathcal{R}^m, \mathbf{x}^m)$. A substrate consisting of N_g atoms is represented as $\mathcal{G} = (\mathbf{x}^g, \mathbf{f}, \mathbf{e})$, where $\mathbf{x}^g = (x_1^g, \ldots, x_{N_g}^g) \in \mathbb{R}^{N_g\times 3}$ denotes its atomic coordinates, while $\mathbf{f}$ and $\mathbf{e}$ denote atom features

and bond information, respectively. Our goal is to model the conditional distribution $p(\mathbf{x} \mid \mathcal{M}, \mathcal{G})$ over enzyme backbones conditioned on a catalytic motif and a substrate.

Preliminaries. We adopt conditional flow matching (CFM) (Lipman et al., 2022; Tong et al., 2023) for enzyme backbone generation. The generative process follows the ordinary differential equation (ODE)

$$d\mathbf{x}_t = \mathbf{v}_\theta(\mathbf{x}_t, t \mid C)\, dt, \qquad t \in [0, 1], \tag{1}$$

where $\mathbf{v}_\theta$ is a learnable vector field and C denotes the conditioning information. Starting from Gaussian noise at $t = 0$, the ODE transports $\mathbf{x}_t$ toward the enzyme backbone data distribution at $t = 1$.

For training, we construct a linear interpolation between Gaussian noise $\mathbf{x}_0$ and a native enzyme backbone $\mathbf{x}_1$:

$$\mathbf{x}_t = (1 - t)\mathbf{x}_0 + t\mathbf{x}_1, \tag{2}$$

where $\mathbf{x}_0 \sim \mathcal{N}(0, \sigma_0^2 I)$ and $t \sim \mathcal{U}(0, 1)$. The corresponding conditional flow-matching objective is

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{\mathbf{x}_1, \mathbf{x}_0, t}\left[\|\mathbf{v}_\theta(\mathbf{x}_t, t \mid C) - (\mathbf{x}_1 - \mathbf{x}_0)\|_2^2\right]. \tag{3}$$

3.2 Functional Shell Representation

Generating a full backbone from the catalytic motif and substrate requires inferring a suitable local protein environment around them. We therefore introduce the functional shell as the explicit structural representation of this local environment. The shell includes the catalytic motif itself, together with residues surrounding either the motif or the substrate (Figure 2). Specifically, given an enzyme backbone $\mathbf{x} = (x_1, \ldots, x_N)$, we define the substrate and motif neighborhoods as

$$\mathcal{B}^g(\mathbf{x}, \mathcal{G}) = \left\{ i \in \{1, \ldots, N\} : \min_{1 \le a \le N_g} \|x_i - x_a^g\|_2 \le r_g \right\}, \tag{4}$$

$$\mathcal{B}^m(\mathbf{x}, \mathcal{M}) = \left\{ i \in \{1, \ldots, N\} : \min_{j \in \mathcal{R}^m} \|x_i - x_j\|_2 \le r_m \right\}. \tag{5}$$

Here, we set the substrate and motif neighborhood radii to $r_g = 8$ Å and $r_m = 6$ Å, respectively. The resulting residue set defines the functional shell.

$$\mathcal{R}^s = \mathcal{R}^m \cup \mathcal{B}^m(\mathbf{x}, \mathcal{M}) \cup \mathcal{B}^g(\mathbf{x}, \mathcal{G}). \tag{6}$$

Let $N_s = |\mathcal{R}^s|$ denote the number of residues in the shell. We represent the shell geometry by C_α coordinates $\mathbf{x}^s \in \mathbb{R}^{N_s \times 3}$, and define the functional shell condition as $\mathcal{S} = (\mathcal{R}^s, \mathbf{x}^s)$.

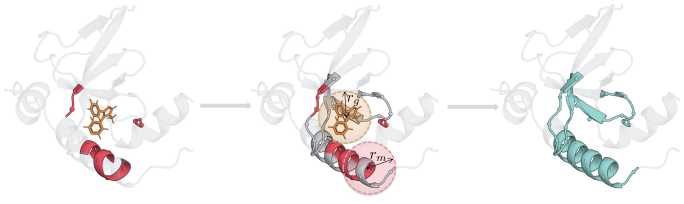

Figure 2: **Functional shell construction.** Left: Catalytic motif (red) and substrate (orange). Middle: Residues are selected within $r_g = 8$ Å of any substrate atom or $r_m = 6$ Å of any motif C_α atom, using residue C_α coordinates. Dashed regions illustrate the cutoffs schematically. Right: These neighborhoods and all motif residues form the functional shell (teal).

During training, the shell is constructed from the native enzyme backbone, whereas during sampling it is constructed entirely from a predicted one, including the coordinates at motif positions rather than using the prescribed motif coordinates $\mathbf{x}^m$.

3.3 DUAL CONDITIONAL FLOW MODELS

Model Architecture. Both conditional flow models build on the non-equivariant C_α transformer of Proteina (Geffner et al., 2025), with the same architecture but separate parameters. At each flow time t, the model predicts a vector field over the noisy backbone $\mathbf{x}_t$, conditioned on either the catalytic motif $\mathcal{M}$ or functional shell $\mathcal{S}$, together with the substrate $\mathcal{G}$. We extend Proteina with substrate atom tokens and protein–substrate pair features to incorporate substrate information, including atomic features, pairwise distances, and molecular connectivity. Further architecture and implementation details are provided in Appendix A.3.

Motif-conditioned Flow Model. The motif-conditioned model predicts a full-backbone vector field conditioned on the catalytic motif and substrate:

$$\mathbf{v}_{\theta_M}(\mathbf{x}_t, t \mid \mathcal{M}, \mathcal{G}). \tag{7}$$

Conditioned on the prescribed motif and substrate, the model must infer a suitable local protein environment while generating the full backbone.

Shell-conditioned Flow Model. The shell-conditioned model instead predicts a full-backbone vector field conditioned on the functional shell and substrate:

$$\mathbf{v}_{\theta_S}(\mathbf{x}_t, t \mid \mathcal{S}, \mathcal{G}). \tag{8}$$

The functional shell explicitly provides the local protein environment around the catalytic motif and substrate. The two models thus play complementary roles: the motif-conditioned model infers this environment while generating the full backbone, whereas the shell-conditioned model uses it as an explicit structural condition for full-backbone generation.

During sampling, however, the shell is constructed from backbone predictions of the motif-conditioned model, introducing a mismatch between training and inference. To improve robustness to imperfect shell conditions, we add Gaussian noise to the shell coordinates during training (Ho et al., 2022), with details provided in Appendix A.1.

Training Objectives. Both models are trained using the following weighted flow-matching objective:

$$\mathcal{L}_{\mathrm{FM}} = \frac{1}{3N}\sum_{i=1}^{N} \ell_i + \frac{w_s}{3N_s}\sum_{i\in\mathcal{R}^s} \ell_i, \tag{9}$$

where the per-residue flow-matching loss is

$$\ell_i = \|v_{\theta,i}(\mathbf{x}_t, t \mid C) - (x_{1,i} - x_{0,i})\|_2^2 . \tag{10}$$

Here, C denotes the conditioning information, with $C = (\mathcal{M}, \mathcal{G})$ for the motif-conditioned model and $C = (\mathcal{S}, \mathcal{G})$ for the shell-conditioned model. The full-backbone and shell terms are normalized separately by their respective numbers of coordinate components. We set $w_s = 5$, giving residues in the functional shell greater weight during flow-matching training.

In addition to the flow-matching loss, we use auxiliary losses to supervise protein–substrate and residue–residue relationships. Specifically, $\mathcal{L}_{\mathrm{sub}}$ and $\mathcal{L}_{\mathrm{contact}}$ supervise protein–substrate distances and contacts, respectively, while $\mathcal{L}_{\mathrm{distogram}}$ supervises pairwise residue distances. The shared objective is then

$$\mathcal{L}_{\mathrm{shared}} = \mathcal{L}_{\mathrm{FM}} + \mathcal{L}_{\mathrm{sub}} + \mathcal{L}_{\mathrm{contact}} + \mathbf{1}_{\{t>0.3\}}\mathcal{L}_{\mathrm{distogram}}, \tag{11}$$

where the indicator $\mathbf{1}_{\{t>0.3\}}$ restricts the distogram loss to $t > 0.3$.

The motif-conditioned model is trained with

$$\mathcal{L}_{\mathcal{M}} = \mathcal{L}_{\mathrm{shared}}, \tag{12}$$

whereas the shell-conditioned model uses an additional auxiliary interface loss:

$$\mathcal{L}_{\mathcal{S}} = \mathcal{L}_{\mathrm{shared}} + \mathcal{L}_{\mathrm{interface}}. \tag{13}$$

The interface loss supervises pairwise distances between neighboring shell and non-shell residues. Detailed definitions of the auxiliary losses and training details are provided in Appendices A.2 and A.3, respectively.

3.4 DYNAMIC SHELL-GUIDED SAMPLING

At inference, the two models jointly generate a single enzyme backbone trajectory through sampling steps $0 = t_0 < \cdots < t_K = 1$. Let $\mathbf{x}_k$ denote the backbone at time t_k. Since the functional shell is not given at inference, it must be inferred during backbone generation. Starting from Gaussian noise, we dynamically construct the functional shell from the motif-conditioned model's predictions, then combine the two predicted vector fields to update the backbone at each sampling step.

Dynamic shell conditioning. Related to self-conditioning in diffusion and flow-based generative models, which reuses intermediate model predictions during sampling, we use predictions of the motif-conditioned model to dynamically construct the shell condition for the shell-conditioned model. The motif-conditioned model is conditioned on the prescribed motif and substrate throughout sampling. At step k, it predicts the vector field

$$\mathbf{v}_k^{(\mathcal{M})} = \mathbf{v}_{\theta_M}(\mathbf{x}_k, t_k \mid \mathcal{M}, \mathcal{G}), \tag{14}$$

from which we obtain the corresponding clean-backbone prediction

$$\hat{\mathbf{x}}_{1|k}^{(\mathcal{M})} = \mathbf{x}_k + (1 - t_k)\mathbf{v}_k^{(\mathcal{M})}. \tag{15}$$

The shell condition is initialized with the prescribed motif and is subsequently constructed from the clean prediction of the motif-conditioned model at the preceding sampling step. Specifically, we define

$$\mathcal{S}_k = \begin{cases} (\mathcal{R}^m, \mathbf{x}^m), & k = 0, \\ (\mathcal{R}_k^s, \mathbf{x}_k^s), & k \geq 1, \end{cases} \tag{16}$$

where, for $k \geq 1$, the shell residue set is determined from $\hat{\mathbf{x}}_{1|k-1}^{(\mathcal{M})}$ according to the functional-shell definition in Section 3.2:

$$\mathcal{R}_k^s = \mathcal{R}^m \cup \mathcal{B}^m\left(\hat{\mathbf{x}}_{1|k-1}^{(\mathcal{M})}, \mathcal{M}\right) \cup \mathcal{B}^g\left(\hat{\mathbf{x}}_{1|k-1}^{(\mathcal{M})}, \mathcal{G}\right), \tag{17}$$

and the corresponding shell coordinates $\mathbf{x}_k^s$ are extracted from $\hat{\mathbf{x}}_{1|k-1}^{(\mathcal{M})}$ at residue positions $\mathcal{R}_k^s$.

Following RFdiffusion2 (Ahern et al., 2026), we center the conditioning coordinates during training to prevent the model from exploiting their absolute position relative to the origin. We apply the same procedure during sampling. At step k, we compute the centroid of the current shell and use it to center both the shell and substrate coordinates:

$$c_k = \frac{1}{N_{s,k}} \sum_{j=1}^{N_{s,k}} x_{k,j}^s, \qquad \bar{\mathbf{x}}_k^s = \mathbf{x}_k^s - c_k, \qquad \bar{\mathbf{x}}_k^g = \mathbf{x}^g - c_k, \tag{18}$$

where $N_{s,k} = |\mathcal{R}_k^s|$. This gives the centered conditions $\bar{\mathcal{S}}_k = (\mathcal{R}_k^s, \bar{\mathbf{x}}_k^s)$ and $\bar{\mathcal{G}}_k = (\bar{\mathbf{x}}_k^g, \mathbf{f}, \mathbf{e})$. Under the linear interpolation $\mathbf{x}_t = (1 - t)\mathbf{x}_0 + t\mathbf{x}_1$, centering the clean backbone while retaining the same reference noise gives $\bar{\mathbf{x}}_k = \mathbf{x}_k - t_k c_k$. The shell-conditioned vector field is evaluated in this representation and converted back to the shared frame:

$$\mathbf{v}_k^{(\mathcal{S})} = \mathbf{v}_{\theta_S}\left(\bar{\mathbf{x}}_k, t_k \mid \bar{\mathcal{S}}_k, \bar{\mathcal{G}}_k\right) + c_k. \tag{19}$$

The centroid and coordinate transformations are recomputed at each sampling step.

Vector-Field Combination. At each sampling step, we combine the vector fields predicted by the two models as

$$\mathbf{v}_k = (1 - \alpha)\mathbf{v}_k^{(\mathcal{M})} + \alpha \mathbf{v}_k^{(\mathcal{S})}, \tag{20}$$

where $\alpha \in [0, 1]$ controls the contribution of the shell-conditioned model. Following the stochastic sampling procedure of Proteina (Geffner et al., 2025), we compute the corresponding score as

$$\mathbf{s}_k = \frac{t_k \mathbf{v}_k - \mathbf{x}_k}{1 - t_k}, \tag{21}$$

then, the noisy backbone is updated as

$$\mathbf{x}_{k+1} = \mathbf{x}_k + (\mathbf{v}_k + g_k \mathbf{s}_k)\,\Delta t_k + \sqrt{2 g_k \gamma \Delta t_k}\,\epsilon_k, \qquad \epsilon_k \sim \mathcal{N}(0, I), \tag{22}$$

where $\Delta t_k = t_{k+1} - t_k$, $g_k = g(t_k)$ controls the stochastic Langevin component, and γ controls the noise scale. We set $g_k = 0$ near the end of sampling. Algorithm 1 summarizes the complete sampling procedure. Sampling details are provided in Appendix A.3.

Algorithm 1 Dynamic Shell-Guided Sampling in EnzyShell

Input: Motif-conditioned model $\mathbf{v}_{\theta_M}$, shell-conditioned model $\mathbf{v}_{\theta_S}$, motif $\mathcal{M} = (\mathcal{R}^m, \mathbf{x}^m)$, substrate $\mathcal{G} = (\mathbf{x}^g, \mathbf{f}, \mathbf{e})$, backbone length N, shell radii r_g, r_m, reference scale σ_0, combination weight $\alpha \in [0, 1]$, sampling grid $0 = t_0 < \cdots < t_K = 1$.
 # Initialize the backbone and shell condition
$\mathbf{x}_0 \sim \mathcal{N}(0, \sigma_0^2 I) \in \mathbb{R}^{N\times 3}$
$\mathcal{R}_0^s \leftarrow \mathcal{R}^m$, $\mathbf{x}_0^s \leftarrow \mathbf{x}^m$, $\mathcal{S}_0 \leftarrow (\mathcal{R}_0^s, \mathbf{x}_0^s)$
for $k = 0, \ldots, K-1$ **do**
  # Compute the motif-conditioned vector field
  $\mathbf{v}_k^{(\mathcal{M})} \leftarrow \mathbf{v}_{\theta_M}(\mathbf{x}_k, t_k \mid \mathcal{M}, \mathcal{G})$
  # Predict the clean backbone from the motif-conditioned model
  $\hat{\mathbf{x}}_{1|k}^{(\mathcal{M})} \leftarrow \mathbf{x}_k + (1 - t_k)\mathbf{v}_k^{(\mathcal{M})}$
  # Center the shell and substrate conditions
  $N_{s,k} \leftarrow |\mathcal{R}_k^s|$, $c_k \leftarrow \dfrac{1}{N_{s,k}} \displaystyle\sum_{j=1}^{N_{s,k}} x_{k,j}^s$
  $\bar{\mathbf{x}}_k^s \leftarrow \mathbf{x}_k^s - c_k, \quad \bar{\mathbf{x}}_k^g \leftarrow \mathbf{x}^g - c_k$
  $\bar{\mathcal{S}}_k \leftarrow (\mathcal{R}_k^s, \bar{\mathbf{x}}_k^s), \quad \bar{\mathcal{G}}_k \leftarrow (\bar{\mathbf{x}}_k^g, \mathbf{f}, \mathbf{e})$
  # Transform the noisy backbone for the shell-conditioned model
  $\bar{\mathbf{x}}_k \leftarrow \mathbf{x}_k - t_k c_k$
  # Compute the shell-conditioned vector field
  $\mathbf{v}_k^{(\mathcal{S})} \leftarrow \mathbf{v}_{\theta_S}\left(\bar{\mathbf{x}}_k, t_k \mid \bar{\mathcal{S}}_k, \bar{\mathcal{G}}_k\right) + c_k$
  # Combine the motif- and shell-conditioned vector fields
  $\mathbf{v}_k \leftarrow (1 - \alpha)\mathbf{v}_k^{(\mathcal{M})} + \alpha \mathbf{v}_k^{(\mathcal{S})}$
  # Advance the shared backbone trajectory
  $\mathbf{x}_{k+1} \leftarrow \mathrm{SamplerStep}(\mathbf{x}_k, \mathbf{v}_k, t_k, t_{k+1})$
  if $k < K - 1$ **then**
    # Construct the shell condition for the next sampling step
    $\mathcal{R}_{k+1}^s \leftarrow \mathcal{R}^m \cup \mathcal{B}^m\left(\hat{\mathbf{x}}_{1|k}^{(\mathcal{M})}, \mathcal{M}\right) \cup \mathcal{B}^g\left(\hat{\mathbf{x}}_{1|k}^{(\mathcal{M})}, \mathcal{G}\right)$
    $\mathbf{x}_{k+1}^s \leftarrow \hat{\mathbf{x}}_{1|k}^{(\mathcal{M})}[\mathcal{R}_{k+1}^s]$
    $\mathcal{S}_{k+1} \leftarrow (\mathcal{R}_{k+1}^s, \mathbf{x}_{k+1}^s)$
  end if
end for
return $\mathbf{x}_K$

4 Experiments

4.1 Experimental Setup

Training Sets and Benchmark. We follow EnzyControl (Song et al., 2025) and use its EnzyBind dataset and data splits for training and evaluation. EnzyBind contains 11,100 enzyme-substrate complexes curated from PDBbind (Wang et al., 2005). Enzyme sequences are clustered using CD-HIT (Fu et al., 2012), with clusters assigned to disjoint training and test sets to prevent sequence overlap across the two splits. Catalytic motifs are defined as evolutionarily conserved residues identified by multiple sequence alignment within the same second-level EC family. We evaluate EnzyShell on the EnzyBind test benchmark, which contains 190 enzyme–substrate pairs spanning EC classes 1–6.

Baseline Methods. We compare EnzyShell with Proteína (Geffner et al., 2025), EnzyControl (Song et al., 2025), RFdiffusion2 (Ahern et al., 2026), and RFdiffusion3 (Butcher et al., 2025). EnzyShell is conditioned on both the prescribed catalytic motif and substrate, while each baseline uses the conditioning inputs supported by its original formulation. Further details on the baseline configurations are provided in Appendix B.2.

Table 1: **Overall structural and functional performance on the EnzyBind benchmark.** Best and second-best results are highlighted in **bold** and underline, respectively.

Model	Designability ↑	Motif RMSD (Å) ↓	# Succ. Clusters ↑	EC Match Rate↑	k_{cat} ↑	Gnina docking score ↓
Proteina	0.191	5.11	97	0.171	2.63	-6.45
EnzyControl	0.498	2.65	153	0.232	2.90	-6.53
RFdiffusion2	0.234	3.54	129	0.240	2.90	**-6.61**
RFdiffusion3	0.517	**1.77**	164	0.395	3.31	-6.60
EnzyShell	**0.752**	1.82	**171**	**0.435**	**3.38**	-6.50

Table 2: **Ablation of dynamic shell-guided sampling.** The default sampling budget is $N_{\text{step}} = 400$ steps per model unless otherwise specified. Best and second-best results are shown in **bold** and underline, respectively.

Sampling Strategy	Designability ↑	Motif RMSD (Å) ↓	# Succ. Clusters ↑
Standalone single-model sampling			
Motif-model sampling	0.689	1.92	167
Motif-model sampling ($2 \times N_{\text{step}}$)	0.700	1.88	167
Shell-model sampling	0.455	2.83	149
Shell-model sampling ($2 \times N_{\text{step}}$)	0.468	2.78	158
Two-model sampling with shell conditioning			
Sequential two-model sampling	0.726	1.94	169
Dynamic shell-guided sampling (EnzyShell)	**0.752**	**1.82**	**171**

Evaluation Protocol and Metrics. We evaluate all methods using a consistent pipeline in which sequences are designed for each generated backbone with ProteinMPNN (Dauparas et al., 2022) and subsequently refolded with ESMFold (Lin et al., 2023). For structural evaluation, we report **Designability** to measure sequence–structure consistency, **Motif RMSD** to measure preservation of the prescribed catalytic motif, and **# Succ. Clusters** to measure structural diversity among successful designs. For functional evaluation, we report **EC Match Rate** using CLEAN (Yu et al., 2023b), predicted k_{cat} using UniKP (Yu et al., 2023a), and the **Gnina docking score** (McNutt et al., 2021) for predicted enzyme–substrate binding. Full evaluation details are provided in Appendix B.1.

4.2 Main Results

Table 1 compares EnzyShell with representative baselines across structural and predicted functional metrics. For structural quality, EnzyShell achieves the highest designability of 0.752, compared with 0.517 for RFdiffusion3, while maintaining a comparable motif RMSD (1.82 Å vs. 1.77 Å). Among designs satisfying both the designability criterion and motif RMSD < 1.5 Å, EnzyShell also yields the largest number of Foldseek clusters (171 vs. 164).

For predicted functional metrics, EnzyShell achieves the highest EC match rate (0.435) and predicted k_{cat} (3.38), compared with 0.395 and 3.31 for RFdiffusion3, respectively. Its Gnina docking score is -6.50, compared with -6.61 for RFdiffusion2, -6.60 for RFdiffusion3, and -6.53 for EnzyControl.

4.3 Ablation of Dynamic Shell-Guided Sampling

We compare standalone, sequential, and dynamic shell-guided sampling to examine how the two conditional flow models contribute during sampling (Table 2). Standalone sampling uses either model independently. Sequential sampling constructs a fixed functional shell from the final motif-conditioned prediction and uses it throughout a subsequent shell-conditioned trajectory. EnzyShell instead dynamically updates the shell from motif-conditioned backbone predictions (Eq. 16) and combines the two vector fields along a shared trajectory (Eq. 20).

Limited Gains from Additional Sampling. We double the sampling budget of each standalone model to control for the additional model evaluations used by the two-model strategies. For standalone shell-model sampling, the shell is initialized with the prescribed motif, $\mathcal{S}_0 = \mathcal{M}$, without dynamically inferred shell residues. Doubling the sampling budget provides only limited improvements for either

Table 3: **Effect of the vector-field combination weight α.** Larger α assigns greater weight to the shell-conditioned vector field. All other inference settings are fixed.

Combination Weight α	Designability ↑	Motif RMSD (Å) ↓
0.00	0.689	1.92
0.25	0.719	1.88
0.50	0.745	1.83
0.65	**0.752**	**1.82**
0.75	0.744	1.84
0.85	0.744	1.89
1.00	0.736	1.91

model (Table 2), indicating that additional sampling alone does not account for the gains of EnzyShell. The lower standalone performance of the shell-conditioned model is consistent with its reliance on an explicit functional shell during training.

Benefits of Dynamic Shell-Guided Sampling. Under the same sampling budget, EnzyShell achieves higher designability, lower motif RMSD, and more successful structural clusters than sequential sampling. This improvement shows that dynamically updating the shell and combining the two vector fields along a shared trajectory is more effective than passing a fixed shell between two sequential trajectories.

4.4 Effect of Vector-Field Weighting

We vary the combination weight α in Eq. 20 while keeping all other inference settings fixed. Larger values of α assign greater weight to the shell-conditioned vector field. At $\alpha = 0$ and $\alpha = 1$, backbone updates use only the motif- and shell-conditioned fields, respectively; the motif-conditioned model is still used to construct the dynamic shell at $\alpha = 1$.

As shown in Table 3, intermediate weights consistently reduce motif RMSD relative to both endpoints, while weights between 0.50 and 0.85 also improve designability. The best observed performance is obtained at $\alpha = 0.65$. Performance remains stable over $\alpha = 0.50$–0.75, indicating that the improvement does not depend on a narrowly tuned combination weight. These results demonstrate the benefit of combining both vector fields during shared-trajectory sampling.

5 Conclusion

We propose EnzyShell, a dynamic shell-guided framework for motif- and substrate-conditioned enzyme backbone generation. EnzyShell explicitly represents the local protein environment around catalytic motifs and substrates as a functional shell and dynamically uses backbone predictions from the motif-conditioned flow to construct the structural condition for the shell-conditioned flow. Experiments on EnzyBind demonstrate improved backbone designability, EC match rate, and predicted k_{cat} over the evaluated baselines while maintaining comparable catalytic-motif geometry.

Several limitations remain. First, our evaluation remains computational and requires experimental validation. Second, the functional metrics rely on predictive models and may not fully reflect experimental outcomes. Future work could extend EnzyShell to incorporate richer structural information, including side-chain and all-atom constraints.

AI USE STATEMENT

We used generative AI tools to assist with editing and improving the readability of the manuscript. All AI-assisted text was reviewed and revised by the authors. The authors take full responsibility for the final content of this work.

REFERENCES

Woody Ahern, Jason Yim, Doug Tischer, Saman Salike, Seth M Woodbury, Donghyo Kim, Indrek Kalvet, Yakov Kipnis, Brian Coventry, Han Raut Altae-Tran, et al. Atom-level enzyme active site scaffolding using rfdiffusion2. *Nature Methods*, 23(1):96–105, 2026.

Katherine I Albanese, Sophie Barbe, Shunsuke Tagami, Derek N Woolfson, and Thomas Schiex. Computational protein design. *Nature Reviews Methods Primers*, 5(1):13, 2025.

Rebecca Blomberg, Hajo Kries, Daniel M Pinkas, Peer RE Mittl, Markus G Grütter, Heidi K Privett, Stephen L Mayo, and Donald Hilvert. Precision is essential for efficient catalysis in an evolved kemp eliminase. *Nature*, 503(7476):418–421, 2013.

Markus Braun, Adrian Tripp, Morakot Chakatok, Sigrid Kaltenbrunner, Celina Fischer, David Stoll, Aleksandar Bijelic, Wael Elaily, Massimo G Totaro, Melanie Moser, et al. Computational enzyme design by catalytic motif scaffolding. *Nature*, 649(8095):237–245, 2026.

Jasper Butcher, Rohith Krishna, Raktim Mitra, Rafael I Brent, Yanjing Li, Nathaniel Corley, Paul T Kim, Jonathan Funk, Simon Mathis, Saman Salike, et al. De novo design of all-atom biomolecular interactions with rfdiffusion3. *bioRxiv*, 2025.

Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. *arXiv preprint arXiv:2208.04202*, 2022.

Alexander E Chu, Tianyu Lu, and Po-Ssu Huang. Sparks of function by de novo protein design. *Nature biotechnology*, 42(2):203–215, 2024.

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.

Jiahua Deng and Qiang Cui. Second-shell residues contribute to catalysis by predominately pre-organizing the apo state in pafa. *Journal of the American Chemical Society*, 145(20):11333, 2023.

Kieran Didi, Zuobai Zhang, Guoqing Zhou, Danny Reidenbach, Zhonglin Cao, Sooyoung Cha, Tomas Geffner, Christian Dallago, Jian Tang, Michael M. Bronstein, Martin Steinegger, Emine Kucukbenli, Arash Vahdat, and Karsten Kreis. Scaling atomistic protein binder design with generative pretraining and test-time compute. In *International Conference on Learning Representations*, 2026.

Limin Fu, Beifang Niu, Zhengwei Zhu, Sitao Wu, and Weizhong Li. Cd-hit: accelerated for clustering the next-generation sequencing data. *Bioinformatics*, 28(23):3150–3152, 2012.

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. In *International Conference on Learning Representations*, 2025.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research*, 23(47):1–33, 2022.

Chenqing Hua, Jiarui Lu, Yong Liu, Odin Zhang, Jian Tang, Rex Ying, Wengong Jin, Guy Wolf, Doina Precup, and Shuangjia Zheng. Reaction-conditioned de novo enzyme design with genzyme. *arXiv preprint arXiv:2411.16694*, 2024.

Zefeng Lin, Zhihang Zhang, Weirong Zhu, Tongchang Han, Xianyong Fang, Tianfan Fu, and Xiaohua Xu. Enzypgm: pocket-conditioned generative model for substrate-specific enzyme design. *arXiv preprint arXiv:2601.19205*, 2026.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023.

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

Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B Tenenbaum. Compositional visual generation with composable diffusion models. In *European Conference on Computer Vision*, pp. 423–439. Springer, 2022.

Andrew T McNutt, Paul Francoeur, Rishal Aggarwal, Tomohide Masuda, Rocco Meli, Matthew Ragoza, Jocelyn Sunseri, and David Ryan Koes. Gnina 1.0: molecular docking with deep learning. *Journal of cheminformatics*, 13(1):43, 2021.

Luca Miglior, Vincenzo Gervasi, and Davide Bacciu. Compflow: Composing velocity fields for multi-condition generation. In *2nd Workshop on Compositional Learning: Safety, Interpretability, and Agents*, 2026.

Chao Song, Zhiyuan Liu, Han Huang, Liang Wang, Qiong Wang, Jianyu Shi, Hui Yu, Yihang Zhou, and Yang Zhang. EnzyControl: Adding functional and substrate-specific control for enzyme backbone generation. In *Advances in Neural Information Processing Systems*, 2025.

Zhenqiao Song, Yunlong Zhao, Wenxian Shi, Wengong Jin, Yang Yang, and Lei Li. Generative enzyme design guided by functionally important sites and small-molecule substrates. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 46259–46279. PMLR, 2024.

Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *arXiv preprint arXiv:2302.00482*, 2023.

Kostas Triaridis, Alexandros Graikos, Aggelina Chatziagapi, Grigorios G Chrysos, and Dimitris Samaras. Mitigating diffusion model hallucinations with dynamic guidance. *arXiv preprint arXiv:2510.05356*, 2025.

Valerie Vaissier Welborn and Teresa Head-Gordon. Computational design of synthetic enzymes. *Chemical reviews*, 119(11):6613–6630, 2019.

Michel Van Kempen, Stephanie S Kim, Charlotte Tumescheit, Milot Mirdita, Jeongjae Lee, Cameron LM Gilchrist, Johannes Söding, and Martin Steinegger. Fast and accurate protein structure search with foldseek. *Nature biotechnology*, 42(2):243–246, 2024.

Renxiao Wang, Xueliang Fang, Yipin Lu, Chao-Yie Yang, and Shaomeng Wang. The pdbbind database: methodologies and updates. *Journal of medicinal chemistry*, 48(12):4111–4119, 2005.

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, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, 2023.

Jason Yim, Andrew Campbell, Andrew YK Foong, Michael Gastegger, José Jiménez-Luna, Sarah Lewis, Victor Garcia Satorras, Bastiaan S Veeling, Regina Barzilay, Tommi Jaakkola, et al. Fast protein backbone generation with se (3) flow matching. *arXiv preprint arXiv:2310.05297*, 2023.

Han Yu, Huaxiang Deng, Jiahui He, Jay D Keasling, and Xiaozhou Luo. Unikp: a unified framework for the prediction of enzyme kinetic parameters. *Nature communications*, 14(1):8211, 2023a.

Tianhao Yu, Haiyang Cui, Jianan Canal Li, Yunan Luo, Guangde Jiang, and Huimin Zhao. Enzyme function prediction using contrastive learning. *Science*, 379(6639):1358–1363, 2023b.

Yang Zhang and Jeffrey Skolnick. Scoring function for automated assessment of protein structure template quality. *Proteins: Structure, Function, and Bioinformatics*, 57(4):702–710, 2004.

Zaixi Zhang, Wan Xiang Shen, Qi Liu, and Marinka Zitnik. Efficient generation of protein pockets with pocketgen. *Nature machine intelligence*, 6(11):1382–1395, 2024.

Qinqing Zheng, Matt Le, Neta Shaul, Yaron Lipman, Aditya Grover, and Ricky TQ Chen. Guided flows for generative modeling and decision making. *arXiv preprint arXiv:2311.13443*, 2023.

A Additional Method Details

A.1 Shell Perturbation

The shell-conditioned model is trained using functional shells constructed from native backbones, whereas sampling uses shells constructed from predictions of the motif-conditioned model. To improve robustness to imperfect shell conditions, we perturb the shell coordinates during training.

Let the native functional shell be $\mathcal{S} = (\mathcal{R}^s, \mathbf{x}^s)$, where $N_s = |\mathcal{R}^s|$. We compute its centroid as

$$c_s = \frac{1}{N_s} \sum_{i=1}^{N_s} x_i^s. \tag{23}$$

Following the centering procedure used for shell conditioning, the shell and substrate coordinates are translated by $-c_s$.

Independently for each training example, shell perturbation is applied with probability $p_{\text{noise}} = 0.8$. When perturbation is applied, we sample

$$\sigma_s \sim \mathcal{U}(0, \sigma_{\max}), \tag{24}$$

and perturb each centered shell coordinate as

$$\tilde{x}_i^s = x_i^s - c_s + \epsilon_i, \qquad \epsilon_i \sim \mathcal{N}(0, \sigma_s^2 I), \qquad i = 1, \ldots, N_s. \tag{25}$$

The resulting perturbed shell condition is

$$\tilde{\mathcal{S}} = (\mathcal{R}^s, \tilde{\mathbf{x}}^s). \tag{26}$$

We use $\sigma_{\max} = 0.5$ Å in the main configuration.

Noise is applied to all shell coordinates, including those corresponding to the catalytic motif. The perturbed shell is not recentered. The substrate conditions are not perturbed. Shell perturbation is disabled during validation and sampling.

A.2 Training Objective Details

We provide additional details of the training objectives introduced in Sec. 3. We use the same conditioning notation as in the main text, with $C = (\mathcal{M}, \mathcal{G})$ for the motif-conditioned model and $C = (\mathcal{S}, \mathcal{G})$ for the shell-conditioned model. Unless otherwise specified, geometric auxiliary losses are evaluated using the predicted clean backbone, and distances are expressed in nanometers.

Clean backbone prediction. For the linear conditional path

$$\mathbf{x}_t = (1-t)\mathbf{x}_0 + t\mathbf{x}_1, \tag{27}$$

the target velocity is $\mathbf{x}_1 - \mathbf{x}_0$. Given the predicted vector field $\mathbf{v}_\theta(\mathbf{x}_t, t \mid C)$, the corresponding clean-backbone prediction is

$$\hat{\mathbf{x}}_1 = \mathbf{x}_t + (1-t)\mathbf{v}_\theta(\mathbf{x}_t, t \mid C). \tag{28}$$

It follows that

$$\hat{\mathbf{x}}_1 - \mathbf{x}_1 = (1-t)\left[\mathbf{v}_\theta(\mathbf{x}_t, t \mid C) - (\mathbf{x}_1 - \mathbf{x}_0)\right]. \tag{29}$$

Thus, the time-scaled endpoint regression is equivalent to velocity regression along the linear conditional path. The weighted flow-matching objective in Eq. 9 applies full-backbone supervision together with an additional shell term, normalized separately by N and N_s, respectively. We use $w_s = 5$ as specified in the main text.

Protein–substrate distance loss. Let $x_{1,i}$ and $\hat{x}_{1,i}$ denote the native and predicted clean C_α coordinates of residue i, respectively, and let x_a^g denote the coordinate of substrate atom a. We define

$$d_{ia} = \|x_{1,i} - x_a^g\|_2, \qquad \hat{d}_{ia} = \|\hat{x}_{1,i} - x_a^g\|_2. \tag{30}$$

Let $\mathcal{P}_{\text{sub}}$ contain pairs between substrate atoms and shell residues near the substrate. The substrate distance loss is

$$\mathcal{L}_{\text{sub}} = \frac{1}{|\mathcal{P}_{\text{sub}}|} \sum_{(i,a)\in\mathcal{P}_{\text{sub}}} \left(\hat{d}_{ia} - d_{ia}\right)^2. \tag{31}$$

This loss supervises distances between backbone C_α atoms and substrate atoms.

Table 4: **Model architecture configuration.** Both conditional flow models use the same architecture with separate parameters.

Configuration	Value
Transformer layers	10
Attention heads	8
Residue representation dim.	512
Pair representation dim.	256
Register tokens	10
Pair update frequency	Every 2 layers
Atom-type encoding dim.	128
Atom-name encoding dim.	256
Graph positional encoding dim.	32

Protein–substrate contact loss. Using the same pair set $\mathcal{P}_{\mathrm{sub}}$, we define the predicted contact probability and native contact label as

$$\hat{q}_{ia} = \mathrm{sigmoid}\left(\frac{r_c - \hat{d}_{ia}}{\tau}\right), \qquad q_{ia} = \mathbf{1}[d_{ia} \le r_c], \tag{32}$$

where $r_c = 0.45$ nm and $\tau = 0.05$ nm. The contact loss is defined using binary cross-entropy (BCE):

$$\mathcal{L}_{\mathrm{contact}} = \frac{1}{|\mathcal{P}_{\mathrm{sub}}|} \sum_{(i,a)\in\mathcal{P}_{\mathrm{sub}}} \mathrm{BCE}(\hat{q}_{ia}, q_{ia}). \tag{33}$$

The contact labels indicate whether a residue C_α atom lies within the contact cutoff of a substrate atom and do not explicitly encode side-chain contacts, hydrogen bonds, or catalytic interactions.

Shell–backbone interface loss. For a pair of protein residues (i, j), we define the native and predicted clean pairwise distances as

$$D_{ij} = \|x_{1,i} - x_{1,j}\|_2, \qquad \hat{D}_{ij} = \|\hat{x}_{1,i} - \hat{x}_{1,j}\|_2. \tag{34}$$

The interface pair set is

$$\mathcal{P}_{\mathrm{int}} = \left\{(i,j) : i \in \mathcal{R}^s,\ j \notin \mathcal{R}^s,\ D_{ij} \le 10\ \text{Å}\right\}. \tag{35}$$

The interface loss is

$$\mathcal{L}_{\mathrm{interface}} = \frac{1}{|\mathcal{P}_{\mathrm{int}}|} \sum_{(i,j)\in\mathcal{P}_{\mathrm{int}}} \left(\hat{D}_{ij} - D_{ij}\right)^2. \tag{36}$$

This loss is applied only to the shell-conditioned model and supervises pairwise distances between neighboring shell and non-shell residues.

Distogram supervision. An auxiliary head predicts a categorical distribution p_{ij} over 64 distance bins for each valid protein residue pair (i, j). Native C_α distances are discretized using 63 equally spaced boundaries between 0 and 1.0 nm, with an additional overflow bin above the final boundary. Let $b(D_{ij})$ denote the bin corresponding to the native distance D_{ij}. The distogram loss is

$$\mathcal{L}_{\mathrm{distogram}} = -\frac{1}{|\mathcal{P}_{\mathrm{all}}|} \sum_{(i,j)\in\mathcal{P}_{\mathrm{all}}} \log p_{ij,b(D_{ij})}, \tag{37}$$

where $\mathcal{P}_{\mathrm{all}}$ contains all valid protein residue pairs, including diagonal pairs. As defined in Eq. 11, this loss is applied only for $t > 0.3$.

A.3 Implementation Details

Model architecture. Both models use the Proteina C_α transformer (Geffner et al., 2025), extended with substrate conditioning, and have separate parameters. Architecture hyperparameters are summarized in Table 4.

Residue features encode sequence position, chain breaks, structural conditioning masks and coordinates, and self-conditioning coordinates. Protein–protein pair features encode relative sequence separation, distances in the noisy backbone and self-conditioning prediction, and masked distances from the structural condition.

Substrate atoms are represented as additional tokens with atom type, charge, atom name, graph positional encodings, coordinates, and validity indicators. Protein–substrate pair features encode pairwise distances, while substrate–substrate pair features additionally encode bond connectivity and bond order.

Training configuration. We initialize the Proteina C_α transformer parameters in both flow models from the pretrained motif-scaffolding checkpoint `proteina_v1.7_DFS_60M_notri_motif_scaffolding.ckpt`. Both models are optimized using Adam with a learning rate of 5×10^{-5}. Following Proteina, the flow time t for each training example is sampled from $p(t) = 0.98\,\mathrm{Beta}(1.9, 1.0) + 0.02\,\mathcal{U}(0, 1)$. Self-conditioning is applied during training with probability 0.5 using a detached preliminary clean-backbone prediction.

We maintain an exponential moving average (EMA) of the model parameters with a decay of 0.999 and use the EMA parameters for inference. The motif-conditioned and shell-conditioned models are trained for 200 and 50 epochs, respectively, on NVIDIA A800 GPUs.

Inference setup. For each target, we generate a backbone with the annotated sequence length. Catalytic-motif positions are obtained from EnzyBind and restricted to residues with available reference C_α coordinates. Their original chain indices are retained.

The shell is initialized with the prescribed motif, $\mathcal{S}_0 = \mathcal{M}$. At subsequent sampling steps, its coordinates are dynamically constructed from the preceding clean-backbone prediction of the motif-conditioned model, as described in Sec. 3. Motif coordinates in the updated shell are taken from this prediction rather than replaced by the prescribed coordinates.

Sampling configuration. The main configuration uses $K = 400$ shared trajectory updates and a vector-field combination weight of $\alpha = 0.65$. We use the time grid

$$t_k = \frac{1 - 10^{-2k/K}}{1 - 10^{-2}}, \qquad k = 0, \ldots, K, \tag{38}$$

with $\Delta t_k = t_{k+1} - t_k$. For stochastic sampling, we use the time-dependent coefficient

$$g_k = (t_k + \varepsilon_g)^{-1}, \tag{39}$$

and noise scale $\gamma = 0.4$. The standard deviation of the random noise added at step k is $\sqrt{2 g_k \gamma \Delta t_k}$. A single noise realization is applied to the shared backbone update, rather than independently to the two model predictions.

For $t_k < t_{\mathrm{switch}}$, the stochastic update includes the vector-field term, score correction, and random noise. We use $t_{\mathrm{switch}} = 0.99$; after this point, sampling switches to deterministic updates that retain the combined vector field but omit the score correction and random noise.

Coordinate transformations. The shared trajectory is maintained in a common coordinate frame. At each sampling step, the shell-conditioned model receives the centered shell and substrate conditions defined in Eq. 18, together with the transformed noisy backbone $\bar{\mathbf{x}}_k = \mathbf{x}_k - t_k c_k$.

In implementation, the model returns a clean-backbone prediction $\hat{\bar{\mathbf{x}}}^{(\mathcal{S})}_{1|k}$ in the centered representation. We restore this prediction to the shared frame and compute the velocity used for fusion:

$$\begin{aligned} \hat{\mathbf{x}}^{(\mathcal{S})}_{1|k} &= \hat{\bar{\mathbf{x}}}^{(\mathcal{S})}_{1|k} + c_k, \\ \mathbf{v}^{(\mathcal{S})}_k &= \frac{\hat{\mathbf{x}}^{(\mathcal{S})}_{1|k} - \mathbf{x}_k}{1 - t_k}. \end{aligned} \tag{40}$$

Let $\bar{\mathbf{v}}^{(\mathcal{S})}_k$ denote the shell-conditioned model's vector field evaluated at $(\bar{\mathbf{x}}_k, t_k)$ with conditions $(\bar{\mathcal{S}}_k, \bar{\mathcal{G}}_k)$. Since $\hat{\bar{\mathbf{x}}}^{(\mathcal{S})}_{1|k} = \bar{\mathbf{x}}_k + (1 - t_k)\bar{\mathbf{v}}^{(\mathcal{S})}_k$, this procedure is equivalent to $\mathbf{v}^{(\mathcal{S})}_k = \bar{\mathbf{v}}^{(\mathcal{S})}_k + c_k$. The transformed input is used only for the shell-conditioned model; the shared trajectory is updated using the combined vector field in the common frame.

Self-conditioning. The two models maintain separate clean-backbone prediction histories in the shared coordinate frame. The motif-conditioned model uses its previous prediction directly. For the shell-conditioned model, the previous prediction is translated by the current shell centroid before being used as a self-conditioning input:

$$\bar{\mathbf{x}}^{(\mathcal{S})}_{\mathrm{sc},k} = \hat{\mathbf{x}}^{(\mathcal{S})}_{1|k-1} - c_k, \qquad k \geq 1. \tag{41}$$

The combined prediction does not replace either model's individual self-conditioning history.

B Additional Experimental Details

B.1 Evaluation Details

Evaluation protocol. We use the same evaluation pipeline for all methods. For each target, we generate 20 backbones per method. For each generated backbone, we design five candidate sequences using ProteinMPNN (Dauparas et al., 2022) and refold each sequence using ESMFold (Lin et al., 2023). Structural metrics are computed from the resulting ESMFold-predicted structures, while functional metrics are evaluated on the designed sequences and corresponding substrates as described below.

Structural metrics. **Designability** measures sequence–structure consistency. A generated backbone is considered designable if at least one of its five designed sequences refolds to within 2 Å scRMSD of the generated backbone.

For each generated backbone, we select the designed sequence with the lowest scRMSD for subsequent structural evaluation. **Motif RMSD** measures the deviation of the catalytic motif in the selected refolded structure from the prescribed motif geometry. A generated backbone is considered successful if the selected structure has scRMSD < 2 Å and motif RMSD < 1.5 Å.

To evaluate structural diversity, we cluster all successful designs using Foldseek (Van Kempen et al., 2024) with a TM-score (Zhang & Skolnick, 2004) threshold of 0.5 and report the number of resulting clusters as **# Succ. Clusters**.

Functional metrics. Following EnzyControl (Song et al., 2025), we use CLEAN, UniKP, and Gnina to evaluate complementary aspects of predicted enzyme function. **EC Match Rate** is computed using CLEAN (Yu et al., 2023b) and measures the fraction of designed sequences whose predicted EC annotation matches the target annotation. This metric evaluates predicted functional identity rather than experimentally established catalytic activity.

We use UniKP (Yu et al., 2023a) to predict k_{cat} from each designed sequence and its corresponding substrate. These predictions provide a computational proxy for catalytic rate and should not be interpreted as experimentally measured kinetic constants.

Finally, we use Gnina (McNutt et al., 2021) to evaluate predicted enzyme–substrate binding and report the **docking score**, where lower values indicate more favorable predicted binding. The docking score provides a complementary measure of substrate compatibility but does not establish catalytic activity.

B.2 Baseline Configurations

We evaluate all baselines on the same 190 EnzyBind test targets, generating 20 backbones per target. The generated backbones are processed using the shared ProteinMPNN–ESMFold evaluation pipeline described in Section 4.1. Each method receives the conditioning inputs supported by its implementation, as detailed below.

Proteina (Geffner et al., 2025). We use the official implementation (`https://github.com/NVIDIA-BioNeMo/proteina`) and the motif-scaffolding checkpoint `proteina_v1.7_DFS_60M_notri_motif_scaffolding.ckpt`. The model is conditioned on the catalytic motif, with contigs specifying its residue positions and the target backbone length. No substrate conditioning is used. Sampling follows the default motif-scaffolding settings.

EnzyControl (Song et al., 2025). EnzyControl is run using the authors' implementation (`https://github.com/Vecteur-libre/EnzyControl`) and released checkpoint `weights/checkpoint.ckpt`. The model receives the catalytic motif's backbone coordinates and residue mask, together with the substrate representation. We adopt the default enzyme-inpainting configuration in `inference_enzyme.yaml`.

RFdiffusion2 (Ahern et al., 2026). For RFdiffusion2, we adopt the released checkpoint `RFD_173.pt` and default inference settings from the official repository (`https://github.com/RosettaCommons/RFdiffusion2`). The input complex provides both the catalytic motif and substrate, with contigs defining the motif residue positions and target backbone length.

RFdiffusion3 (Butcher et al., 2025). RFdiffusion3 uses the `rfd3_latest.ckpt` checkpoint with default inference settings in Foundry (`https://github.com/RosettaCommons/foundry`). The catalytic motif and substrate are supplied through the input complex structure, with contigs specifying the motif residue positions. Selected motif atoms and all substrate atoms serve as fixed-atom conditions.