

A Limitations

This work focuses exclusively on generating enzyme backbones, without modeling the specific conformations these backbones adopt when binding to substrates, which is highlighted by AtomicFlow [77]. Additionally, while our method, EnzyControl, produces diverse backbone samples, it struggles to balance diversity with designability. Enhancing designability without sacrificing diversity remains an open challenge and a direction for future work.

B More Experimental Results

Due to space limitations, Sec. 5.2 only reports overall test results without distinguishing between enzyme families. To provide a more detailed analysis, Table 8 and 9 present evaluation results broken down by EC family. Additionally, we include functional descriptions for each EC family that appears in the test set, sourced from Expasy¹, as shown in Table 7.

Table 7: EC number and corresponding enzyme functions appeared in the testset.

EC Number	Function
1.1	Acting on the CH-OH group of donors
1.6	Acting on NADH or NADPH
1.14	Acting on paired donors, with incorporation or reduction of molecular oxygen.
2.1	Transferring one-carbon groups
2.3	Acyltransferases
2.5	Transferring alkyl or aryl groups, other than methyl groups
2.7	Transferring phosphorus-containing groups
3.1	Acting on ester bonds
3.2	Glycosylases
3.4	Acting on peptide bonds (peptidases)
3.5	Acting on carbon-nitrogen bonds, other than peptide bonds
3.6	Acting on acid anhydrides
4.1	Carbon-carbon lyases
4.2	Carbon-oxygen lyases
5.6	Isomerases altering macromolecular conformation
5.99	Other isomerases
6.2	Forming carbon-sulfur bonds

C Enzyme Commission number

The Enzyme Commission number (EC number) is a numerical classification scheme for enzymes, based on the chemical reactions they catalyze. As a system of enzyme nomenclature, every EC number is associated with a recommended name for the corresponding enzyme-catalyzed reaction. EC numbers do not specify enzymes but enzyme-catalyzed reactions. If different enzymes (for instance from different organisms) catalyze the same reaction, then they receive the same EC number. We also provide an illustration of EC number in Fig. 11. These categories are applied by our EnzyControl to guide the enzyme backbone generation with specific functions.

D More Details on the Dataset

D.1 Data Licenses

EnzyBind is made available under the Creative Commons Attribution 4.0 International (CC BY 4.0). This license allows users to copy, redistribute, remix, transform, and build upon the dataset for any purpose, including commercial use, provided appropriate credit is given to the creators. A copy of the license is available at <https://creativecommons.org/licenses/by/4.0/>. This dataset is

¹<https://enzyme.expasy.org/>

Table 8: pLDDT comparison on EnzyBench. The best-performing results are marked in **bold**.

EC number	1.1.1	1.14.13	1.14.14	1.2.1	2.1.1	2.3.1	2.4.1	2.4.2	2.5.1	2.6.1	2.7.1	2.7.10	2.7.11	2.7.4	2.7.7
PROTSEED	77.10	71.19	74.24	78.67	77.40	74.54	75.18	77.11	74.79	75.55	75.90	81.05	74.05	76.50	78.00
RFDiffusion+IF	82.47	81.12	89.32	82.04	82.49	85.14	85.61	81.13	86.25	81.60	87.51	86.75	85.91	88.30	81.25
ESM2+EGNN	90.67	90.93	90.30	87.67	79.40	84.78	84.80	84.56	90.21	87.47	83.52	88.92	85.59	90.09	81.80
EnzyGen	91.86	93.02	92.70	91.99	83.47	87.71	92.81	87.02	89.69	89.20	85.19	87.55	87.64	91.81	83.75
Ours	91.64	90.33	93.85	89.63	85.46	88.75	93.92	88.94	91.22	90.51	88.76	89.03	88.39	92.45	86.62
EC number	3.1.1	3.1.3	3.1.4	3.2.2	3.4.19	3.4.21	3.5.1	3.5.2	3.6.1	3.6.4	3.6.5	4.1.1	4.2.1	4.6.1	Avg
PROTSEED	76.29	77.89	75.75	78.76	73.56	82.40	76.70	75.90	75.16	74.62	83.46	76.36	78.87	83.31	76.91
RFDiffusion+IF	80.01	81.59	81.22	92.04	89.72	77.20	84.05	85.47	71.35	82.87	84.49	81.31	79.02	76.11	83.22
ESM2+EGNN	87.27	87.05	85.50	72.23	71.31	82.62	83.48	88.69	84.96	73.34	80.77	87.72	89.70	85.48	84.86
EnzyGen	89.79	85.40	89.68	74.44	77.14	89.11	86.70	89.80	85.98	76.31	84.32	85.71	91.88	87.55	87.21
Ours	90.75	82.33	83.02	87.56	72.19	90.62	87.47	90.06	87.29	84.16	86.33	88.59	90.48	89.91	88.28

Table 9: ESP score comparison on EnzyBench. The best-performing results are marked in **bold**.

EC number	1.1.1	1.14.13	1.14.14	1.2.1	2.1.1	2.3.1	2.4.1	2.4.2	2.5.1	2.6.1	2.7.1	2.7.10	2.7.11	2.7.4	2.7.7
PROTSEED	0.54	0.24	0.39	0.57	0.83	0.52	0.29	0.75	0.58	0.45	0.77	0.88	0.81	0.78	0.69
RFDiffusion+IF	0.45	0.54	0.39	0.47	0.43	0.48	0.39	0.52	0.46	0.53	0.50	0.51	0.60	0.55	0.53
ESM2+EGNN	0.58	0.35	0.35	0.63	0.79	0.53	0.32	0.80	0.59	0.51	0.76	0.88	0.88	0.77	0.70
EnzyGen	0.64	0.38	0.42	0.72	0.80	0.61	0.38	0.86	0.66	0.53	0.76	0.92	0.93	0.80	0.79
Ours	0.67	0.56	0.51	0.65	0.84	0.59	0.47	0.79	0.72	0.65	0.68	0.53	0.55	0.82	0.61
EC number	3.1.1	3.1.3	3.1.4	3.2.2	3.4.19	3.4.21	3.5.1	3.5.2	3.6.1	3.6.4	3.6.5	4.1.1	4.2.1	4.6.1	Avg
PROTSEED	0.70	0.90	0.84	0.48	0.29	0.69	0.31	0.10	0.50	0.57	0.37	0.84	0.83	0.42	0.58
RFDiffusion+IF	0.33	0.61	0.62	0.49	0.62	0.45	0.47	0.44	0.55	0.63	0.59	0.59	0.84	0.45	0.52
ESM2+EGNN	0.71	0.78	0.82	0.43	0.22	0.56	0.35	0.11	0.61	0.73	0.37	0.81	0.89	0.54	0.60
EnzyGen	0.76	0.62	0.88	0.47	0.26	0.73	0.40	0.14	0.66	0.78	0.40	0.80	0.93	0.57	0.64
Ours	0.41	0.44	0.65	0.51	0.63	0.77	0.59	0.53	0.69	0.75	0.66	0.84	0.56	0.60	0.63

derived from the PDBbind database. PDBbind is a curated database of protein-ligand complexes derived from the Protein Data Bank (PDB). Users must also comply with the licensing terms of the original PDB and PDBbind datasets.

D.2 Complex Preprocessing

Traditional data-splitting strategies for enzyme datasets often rely on chronological order—training on complexes published before a certain date and testing on those afterward. However, since our objective is to generate enzyme backbones conditioned on desired functions, we adopt a functionally meaningful split based on sequence similarity. Specifically, we use CD-HIT [86] to cluster enzyme

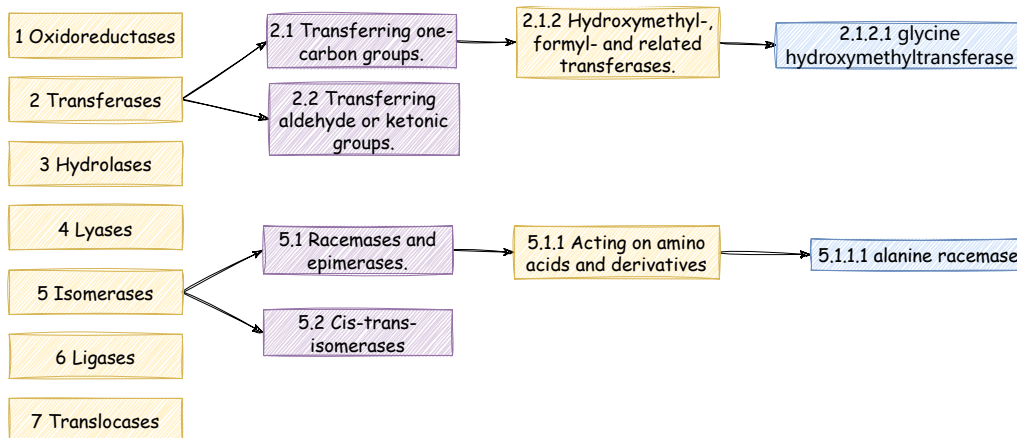


Figure 11: Enzyme Commission (EC) number in BRENDA.

939 sequences and ensure that enzymes in the training and test sets are disjoint. Clusters are then randomly
 940 assigned to either training or testing, and enzyme-substrate pairs are sampled accordingly.

941 Our dataset consists of 11,100 enzyme-substrate complexes, which are first filtered and standardized.
 942 We begin by removing complexes that cannot be parsed by the RDKit library [66]. Following the
 943 preprocessing pipeline from EquiBind [87], we standardize each molecule using Open Babel [88],
 944 correct hydrogen placements on enzymes, and add missing hydrogens with the reduce tool².

945 One remaining challenge is that our model cannot process multi-chain enzymes or symmetric
 946 complexes containing repeated enzyme units, as illustrated in Fig. 12. To address this, we retain only
 947 the substrate atoms that are within 10Å of any enzyme atom, ensuring that each sample represents a
 948 physically relevant interaction while excluding redundant or ambiguous structural data.

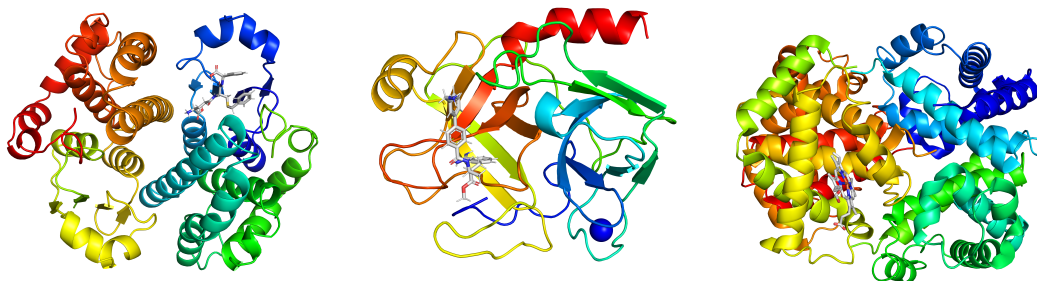


Figure 12: Examples of multi-chain structures and symmetric enzyme complexes.

949 D.3 Multiple Sequence Alignment

950 We identify functional sites in protein sequences using
 951 multiple sequence alignment (MSA). As illustrated in Fig.
 952 13, each row represents an enzyme sequence from the same
 953 enzyme family, based on the second-level classification in
 954 the BRENDA database.

955 We align these sequences using MAFFT and identify
 956 conserved residues by applying an identity threshold τ .
 957 Residues that appear consistently across all aligned se-
 958 quences—such as F, L, and E in our example—are consid-
 959 ered functional sites. Following the EnzyGen approach,
 960 we set $\tau = 0.3$ in our experiments.

961 Once functional sites are determined, we encode them as a
 962 binary vector with the same length as the original sequence.
 963 Each position in the vector is set to 1 if the corresponding
 964 amino acid is a functional site, and 0 otherwise.

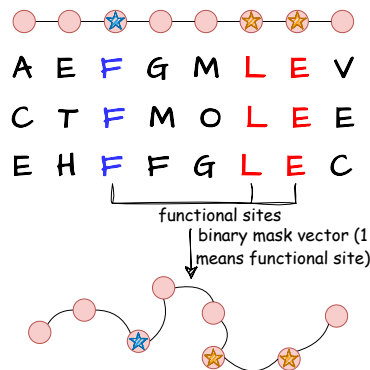


Figure 13: Multiple sequence alignment.

965 E More Methodology Details

966 E.1 Backbone Frame Representation

We represent each residue’s backbone atoms using a local reference frame Fig. 14. As noted in Sec. 4.1, we assume idealized atomic coordinates for the backbone atoms—N, C $_{\alpha}$, C, O—based on standard chemical bond lengths and angles. To construct a local frame for each residue, we follow the rigid3Point procedure used in AlphaFold2. This method defines a coordinate frame from the

²<https://github.com/rlabduke/reduce>

backbone atoms using the following steps:

$$\begin{aligned}
v_1 &= \mathbf{C} - \mathbf{C}_\alpha, & v_2 &= \mathbf{N} - \mathbf{C}_\alpha \\
e_1 &= v_1 / \|v_1\|, & u_2 &= v_2 - e_1(e_1^T v_2) \\
e_2 &= u_2 / \|u_2\| \\
e_3 &= e_1 \times e_2 \\
R &= \text{concat}(e_1, e_2, e_3) \\
x &= \mathbf{C}_\alpha \\
T &= (R, x)
\end{aligned}$$

where the first four lines follow from Gram-Schmidt. The operation of going from coordinates to frames is called atom2frame.

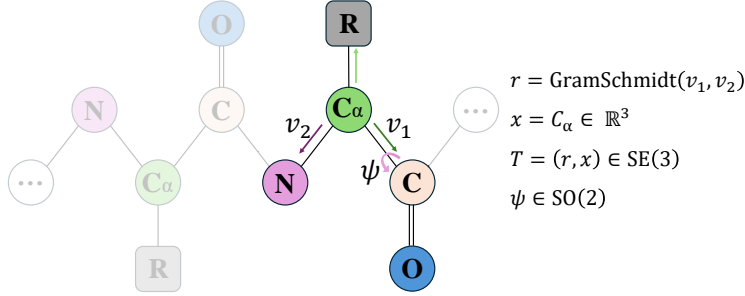


Figure 14: Backbone frame representation.

E.2 Model Architecture of Projector

The projector consists of two linear layers and a layer normalization (Fig. 15), it can decompose the substrate embedding from the pretrained Uni-Mol encoder and output well-aligned substrate features.

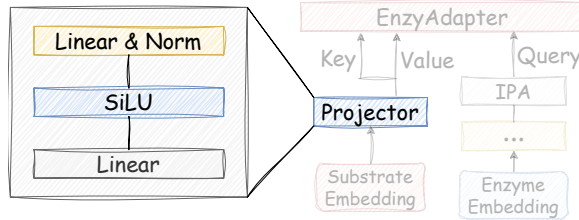


Figure 15: Model architecture of projector.

E.3 Edge and Backbone Update

Edge Update. The edge update step is a critical component of the message-passing mechanism in the Evoformer architecture used by AlphaFold 2. It updates the pairwise edge features by integrating information from the source and target node embeddings. This process is formally defined as follows:

$$\begin{aligned}
h_{\text{down}} &= \text{Linear}(h_{k+1}), & h_{\text{down}} &\in \mathbb{R}^{D_h/2} \\
z_{\text{in}}^{nm} &= \text{concat}(h_{\text{down}}^n, h_{\text{down}}^m, z_k^{nm}), & z_{\text{in}}^{nm} &\in \mathbb{R}^{D_h + D_z} \\
z_{k+1}^{nm} &= \text{LayerNorm}(\text{MLP}(z_{\text{in}}^{nm})), & z_{k+1}^{nm} &\in \mathbb{R}^{D_z}
\end{aligned} \tag{6}$$

We elaborate on each step below:

- At layer $k + 1$, each node embedding $h_{k+1} \in \mathbb{R}^{D_h}$ is projected to a lower-dimensional representation of size $D_h/2$ using a learned linear transformation. This projection reduces the computational cost of subsequent operations and serves as a bottleneck that encourages the model to extract the most salient features for edge-level reasoning. Notably, this transformation does not include a non-linear activation function.

- To construct the input to the edge update MLP for the residue pair (n, m) , the model concatenates three components: the down-projected source node embedding $\mathbf{h}_{\text{down}}^n$, the down-projected target node embedding $\mathbf{h}_{\text{down}}^m$, and the current edge feature vector \mathbf{z}_k^{nm} . This combined representation $\mathbf{z}_{\text{in}}^{nm}$ lies in $\mathbb{R}^{D_h + D_z}$ and captures both contextual and pairwise information relevant to the interaction between residues n and m .
- The combined vector $\mathbf{z}_{\text{in}}^{nm}$ is passed through a multi-layer perceptron (MLP), which typically consists of multiple fully connected layers with non-linear activation functions such as ReLU or GELU. The MLP outputs an updated edge embedding $\mathbf{z}_{k+1}^{nm} \in \mathbb{R}^{D_z}$. A Layer Normalization operation is applied afterward to stabilize the learning dynamics and ensure consistent feature scaling across the embedding dimension.

Backbone Update. The backbone update step in AlphaFold2 is responsible for refining the 3D position and orientation of each residue’s local frame using a learnable rigid-body transformation. This transformation is modeled as an element of the special Euclidean group $\text{SE}(3)$, combining both rotation and translation. The update proceeds through the following sequence of operations:

$$\begin{aligned}
b, c, d, x_{\text{update}} &= \text{Linear}(h_k) \\
(a, b, c, d) &= (1, b, c, d) / \sqrt{1 + b^2 + c^2 + d^2} \\
R_{\text{update}} &= \begin{pmatrix} a^2 + b^2 - c^2 - d^2 & 2bc - 2ad & 2bd + 2ac \\ 2bc + 2ad & a^2 - b^2 + c^2 - d^2 & 2cd - 2ab \\ 2bd - 2ac & 2cd + 2ab & a^2 - b^2 - c^2 + d^2 \end{pmatrix} \\
T_{\text{update}} &= (R_{\text{update}}, x_{\text{update}}) \\
T_{k+1} &= T_k \cdot T_{\text{update}}
\end{aligned} \tag{7}$$

We elaborate on each step below:

- A linear transformation is applied to the node embedding $h_k \in \mathbb{R}^{D_h}$, producing three scalar values $b, c, d \in \mathbb{R}$ and a translation vector $x_{\text{update}} \in \mathbb{R}^3$. These values parameterize a spatial transformation to be applied to the current residue frame.
- The scalar 1 is prepended to (b, c, d) to construct a 4-dimensional vector (a, b, c, d) , which is then normalized to unit norm. This vector forms a unit quaternion, a robust and differentiable representation of a 3D rotation.
- The unit quaternion is converted into a 3×3 rotation matrix $R_{\text{update}} \in \text{SO}(3)$ using a closed-form expression. This guarantees orthonormality and preserves the group structure of the transformation.
- The rotation R_{update} is combined with the translation vector x_{update} to form a rigid-body transformation $T_{\text{update}} \in \text{SE}(3)$, representing a learned update in 3D space.
- Finally, the transformation T_{update} is applied to the current residue frame T_k by composition, yielding the updated frame T_{k+1} . This results in a new position and orientation for the residue, enabling progressive refinement of backbone geometry across Evoformer layers.

F More Details on Experimental Settings

F.1 Additional Training Details

We adopt Low-Rank Adaptation (LoRA) with a rank of $r = 16$ and a scaling factor $\alpha = 32$, targeting key linear projection modules across attention and embedding components, as specified in Table 10. The node and edge embeddings are configured with dimensionalities of 256 and 128, respectively.

Our model supports a maximum of 2000 residues and embeds 1000 discrete timesteps using both sinusoidal and learned positional encodings. Node-level features include spatial coordinates, timestep embeddings, and optional chain-level signals. For edge features, we employ relative position encoding, discretized into 22 bins, and include diffusion-specific masks and self-conditioning mechanisms to enhance robustness.

The IPA module comprises six stacked blocks with multi-head attention (8 heads), point-based QK and V projections, and a lightweight sequence-level Transformer consisting of 2 layers with 4 heads

each. These configurations were selected based on empirical validation to balance computational efficiency with modeling capacity.

Table 10: Model configuration and hyperparameter settings.

Module	Parameter	Value	Description
LoRA Settings	<code>lora_r</code>	16	Rank of low-rank matrices in LoRA.
	<code>lora_alpha</code>	32	Scaling factor for LoRA adaptation.
	<code>lora_dropout</code>	0.0	Dropout applied to LoRA layers.
	<code>lora_bias</code>	"none"	Whether LoRA includes bias terms.
	<code>lora_target_modules</code>	List of 12 modules	Target modules for LoRA adaptation.
Embedding Dimensions	<code>node_embed_size</code>	256	Dimensionality of node embeddings.
	<code>edge_embed_size</code>	128	Dimensionality of edge embeddings.
Node Features	<code>c_s</code>	256	Size of node feature representation.
	<code>c_pos_emb</code>	128	Positional embedding dimensionality.
	<code>c_timestep_emb</code>	128	Timestep embedding dimensionality.
	<code>max_num_res</code>	2000	Maximum number of residues.
	<code>timestep_int</code>	1000	Number of discrete time intervals.
	<code>embed_chain</code>	False	Whether to embed chain-level info.
Edge Features	<code>single_bias_transition_n</code>	2	Number of single bias transitions.
	<code>c_s</code>	256	Node representation dimension.
	<code>c_p</code>	128	Edge embedding dimensionality.
	<code>relpos_k</code>	64	Relative positional embedding size.
	<code>feat_dim</code>	64	Feature vector dimensionality.
	<code>num_bins</code>	22	Number of distance bins.
	<code>self_condition</code>	True	Whether to apply self-conditioning.
IPA Module	<code>c_s</code>	256	Input node feature dimension.
	<code>c_z</code>	128	Input edge feature dimension.
	<code>c_hidden</code>	128	Hidden dimension of IPA module.
	<code>no_heads</code>	8	Number of attention heads.
	<code>no_qk_points</code>	8	Number of QK reference points.
	<code>no_v_points</code>	12	Number of V reference points.
	<code>seq_tfmr_num_heads</code>	4	Heads in sequence-level Transformer.
	<code>seq_tfmr_num_layers</code>	2	Layers in sequence-level Transformer.
	<code>num_blocks</code>	6	Number of IPA module blocks.

F.2 Evaluation Details

This section details the computation of each evaluation metric used in our study.

To evaluate self-consistency, we measure the structural similarity between the generated protein backbones and the all-atom structures predicted by ESMFold. Specifically, we compute the TM-score and RMSD between the two. TM-scores are calculated using the `tmtools`, while RMSD is computed after structural alignment following the procedure described in FrameFlow.

We assess two aspects of enzyme properties: EC number classification and catalytic efficiency (k_{cat}). Both are predicted using pretrained models. For EC number prediction, we use CLEAN, which takes only the amino acid sequence as input. For k_{cat} prediction, we input both the sequence and the substrate’s SMILES representation into a separate predictive model. We define the EC match rate as the proportion of samples whose predicted EC number matches that of the corresponding native enzyme. For k_{cat} , we report the average predicted value across all samples.

Substrate binding is evaluated using two methods. First, we perform docking simulations with GNINA to assess how well each enzyme binds to a given molecule. We treat the entire enzyme structure as the search space for docking. Second, we compute the ESP score using the pretrained ESP model, which takes both the sequence and substrate as inputs. The final ESP score is the mean value across all samples.

To evaluate diversity and novelty, we use Foldseek. Diversity is measured by clustering the generated proteins with Foldseek and calculating the ratio of the number of clusters to the total number of samples. Novelty is defined as the average of the maximum TM-scores between each generated enzyme and all native proteins. Lower scores indicate greater novelty, as they reflect less structural similarity to known proteins.

1046 **E.3 Computational Resource**

1047 All experiments were conducted on a high-performance computing node equipped with 4× NVIDIA
1048 A100 GPUs (80GB) and dual Intel(R) Xeon(R) Gold 6348 CPUs (2.60GHz, 2 sockets, 28 cores per
1049 socket, 112 threads in total).