Full-Atom Peptide Design with Geometric Latent Diffusion

Xiangzhe Kong[1,2] **Yinjun Jia**[3] **Wenbing Huang**[4*] **Yang Liu**[1,2,5*]
[1]Dept. of Comp. Sci. & Tech., Tsinghua University
[2]Institute for AIR, Tsinghua University [3]School of Life Sciences, Tsinghua University
[4]Gaoling School of Artificial Intelligence, Renmin University of China
[5]Shanghai Artificial Intelligence Laboratory, Shanghai, China

Abstract

Peptide design plays a pivotal role in therapeutics, allowing brand new possibility to leverage target binding sites that are previously undruggable. Most existing methods are either inefficient or only concerned with the target-agnostic design of 1D sequences. In this paper, we propose a generative model for full-atom **Pep**tide design with **G**eometric **LA**tent **D**iffusion (PepGLAD) given the binding site. We first establish a benchmark consisting of both 1D sequences and 3D structures from Protein Data Bank (PDB) and literature for systematic evaluation. We then identify two major challenges of leveraging current diffusion-based models for peptide design: the full-atom geometry and the variable binding geometry. To tackle the first challenge, PepGLAD derives a variational autoencoder that first encodes full-atom residues of variable size into fixed-dimensional latent representations, and then decodes back to the residue space after conducting the diffusion process in the latent space. For the second issue, PepGLAD explores a receptor-specific affine transformation to convert the 3D coordinates into a shared standard space, enabling better generalization ability across different binding shapes. Experimental Results show that our method not only improves diversity and binding affinity significantly in the task of sequence-structure co-design, but also excels at recovering reference structures for binding conformation generation.

1 Introduction

Peptides are short chains of amino acids and acts as vital mediators of many protein-protein interactions in human cells. Designing functional peptides has attracted increasing attention in biological research and therapeutics, since the highly-flexible conformation space of peptides allows brand new possibility to target binding sites previously undruggable with antibodies or small molecules [17, 35]. The key of peptide design is to generate peptides that interact compactly with target proteins (see Figure 1), since they mostly exhibit flexible conformations [20] unless bound to these receptors [60].

Conventional simulation or searching algorithms rely on frequent calculations of physical energy functions [6, 9], which are inefficient and prone to poor local optimum. Recent advances illuminate the remarkable success of exploiting geometric deep generative models, particularly the equivariant diffusion models [45, 72], for molecule design [21, 39], antibody design [29, 44, 34] and protein design [64, 28], as well as latent diffusion models further enhancing the performance [71, 18]. Inspired by these successes, a natural idea is leveraging diffusion models for peptide design as well, which, yet, is challenging in two aspects. From the dataset aspect, existing databases (PepBDB [65], Propedia [46]) merely collect data from Protein Data Bank (PDB) [5], neither performing filters according to practical relevance [49] and redundancy, nor providing an adequate split for evaluation.

[*]Correspondence to Wenbing Huang <hwenbing@126.com>, Yang Liu <liuyang2011@tsinghua.edu.cn>

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

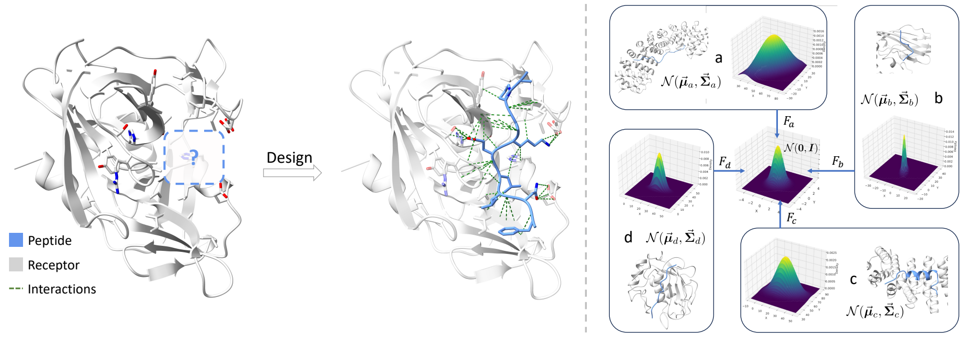

?
Design
Peptide
Receptor
Interactions
a
$\mathcal{N}(\vec{\mu}_a, \vec{\Sigma}_a)$
b
$\mathcal{N}(\vec{\mu}_b, \vec{\Sigma}_b)$
F_a
$\mathcal{N}(\mathbf{0}, \boldsymbol{I})$
F_d
F_b
F_c
d
$\mathcal{N}(\vec{\mu}_d, \vec{\Sigma}_d)$
c
$\mathcal{N}(\vec{\mu}_c, \vec{\Sigma}_c)$

Figure 1: **Left**: Peptide design requires generating peptides that form compact interactions with the binding site on the receptor. The intricacy of protein-peptide interactions demands efficient exploration in the vast space for sequence-structure co-design. **Right**: Different binding sites (a, b, c, d) adopt disparate center offsets and geometric shapes, approximating variable 3D Gaussian distributions that deviate from $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$. We propose to convert the geometry into a standard space approximating standard Gaussian, via an affine transformation derived from the binding site (§3.3).

Therefore, this paper first curates a benchmark from PDB [5] and the literature [59], and then systematically evaluates the generative models in terms of diversity, consistency, and binding affinity.

From the methodology aspect, it is nontrivial to adopt latent diffusion models to characterize the geometry of protein-peptide interactions. The first nontriviality stems from the *full-atom geometry*, which determines the comprehensive protein-peptide interactions in the atomic level, yet difficult to preserve. Throughout the generation process, the type of each amino acid always changes and thus requires us to generate different number of atoms, which is unfriendly to diffusion models that prefer fixed-size generation. Current latent diffusion models on molecules [71] or protein backbones [18] still tackle tasks with fixed number of atoms, thus leaving this challenge untouched. The second nontriviality lies in the *variable binding geometry*. Diffusion models are typically implemented directly in the data space, which might be suitable for regular data (*e.g.* images with fixed value range), yet ill-suited for our case on 3D coordinates where the value range is not fixed and even cursed with high variances due to the rich diversity in protein-peptide interactions. These variances define divergent target distributions of Gaussian with disparate expectation and covariance, which hinders the transferability of the diffusion process across different binding sites and thereby yields unsatisfactory generalization capability. Unfortunately, this point is seldom investigated previously.

To address the above problems, we propose a powerful model for full-atom **Pep**tide design with **G**eometric **LA**tent **D**iffusion (PepGLAD) with the following contributions:

- We construct a new benchmark from PDB and literature based on practical relevance and non-redundancy, then systematically evaluate available sequence-structure co-design models on the task of target-specific peptide design.
- To capture the *full-atom geometry*, we first learn a Variational AutoEncoder (VAE) to obtain a fixed-size latent representation (including a 3D coordinate and a hidden feature) for each residue of the input peptide, and then conduct the diffusion process in this latent space, both of which are conditioned on the binding site to better model protein-peptide interactions. Notably, the proposed design enables our model to accommodate full-atom input and output.
- Regarding the *variable binding geometry*, we derive a shared standard space from the binding sites by proposing a novel skill—receptor-specific affine transformation. Such affine transformation is computed by the center offset and covariance Cholesky decomposition of the binding site coordinates, serving as a mapping from the binding site distribution to standard Gaussian. With the affine transformation applied to both the binding sites and the peptides, we are able to project the shape of all complexes into approximately standard Gaussian distribution, which facilitates generalization to diverse binding sites.

Favorably, all the aforementioned models and processes meet the desired symmetry, *i.e.*, E(3)-equivariance, as proved by us. Experiments on sequence-structure co-design and complex conformation generation demonstrate the superiority of PepGLAD over the existing generative models.

2 Related Work

Peptide Design Conventional methods directly sample residues [6] or building blocks from libraries containing small fragments of proteins [26, 57, 9, 8], with guidance from delicate physical energy functions [2]. These methods are time-consuming and easy to be trapped by local optimum. Recent advances with deep generative models mainly focuses on target-agnostic 1D language models [48], antimicrobial peptides [13, 63], or a subtype of peptides with α-helix [68, 69]. While geometric deep generative models are exhibiting notable potential in other domains of target-specific binder design (*e.g.* antibodies), their capability of target-specific peptide design remains unclear, which is the first problem we answer in this paper. Other contemporary work includes peptide design algorithms with flow matching frameworks [38, 40].

Geometric Protein/Antibody Design Protein design primarily aims to generate stable secondary or tertiary structures [27], where diffusion models demonstrate inspiring performance [66, 58, 3, 72]. In particular, RFDiffusion [64] first generates backbones via diffusion, and then designs the sequences through cycles of inverse folding and structure refining with empirical force fields. Chroma [28] adopts a similar strategy, but further explores controllable generative process with custom energy functions. Antibody design, encompassing a special family of proteins in the immune system to capture antigens, mainly focuses on inpainting complementarity-determining regions (CDRs) at the interface between the antigen and the framework [33, 34, 61], where the geometric diffusion models exhibit promising potential [44, 45] in co-designing sequence and structure. Unlike antibodies which are constrained by framework regions, peptides exhibit a more irregular binding pattern and greater flexibility, adapting to binding sites upon interaction [35]. Thus the target distributions are remarkably divergent on different binding sites, posing an urgent need for more robust generative modeling.

Geometric Latent Diffusion Models Diffusion models learn a denoising trajectory to generate desired data distribution from a prior distribution, commonly standard Gaussian [54, 55, 25]. Recent literature extends diffusion to 3D small molecules satisfying the E(3)-equivariance [70], which triggers subsequent advances in geometric design of macro molecules (*e.g.* antibody, protein) as aforementioned. Further efforts are made to latent diffusion models [53, 71, 18], which implement the generative process in the compressed latent space of pretrained auto-encoders, to improve the performance. Compared to the literature that either encodes atom-wise representation in the latent space for small molecule generation [71], or compress a fixed number of atoms into one latent node for protein backbone generation [18], we explore compression of the full-atom geometry by directly generating different residues with variable number of atoms in the latent space. Moreover, we propose a novel technique, namely the data-specific affine transformations, to enhance the generalization ability of diffusion models, which is barely explored before.

3 Our Method: PepGLAD

We first define the notations in the paper and formalize peptide design in §3.1. The overall workflow of our PepGLAD is presented in Figure 2, which consists of three modules: (1) An autoencoder that defines the joint latent space for sequences and structures conditioned on the full-atom context of the binding site (§3.2); (2) An affine transformation derived from the binding site to project the 3D geometry into a standard space approximating standard Gaussian distribution (§3.3); (3) A latent diffusion model trained on the standard latent space (§3.4). Finally, we summarize the training and the sampling procedures in §3.5.

3.1 Definitions and Notations

We represent binding sites and peptides as geometric graphs $\mathcal{G} = \{(x_i, \vec{\boldsymbol{X}}_i)\}$, where each node i is a residue with its amino acid type x_i and the coordinates of all its c_i atoms $\vec{\boldsymbol{X}}_i \in \mathbb{R}^{c_i \times 3}$. In later sections, we use the simplified notations $i \in \mathcal{G}$ to denote that a node i is in the geometric graph $\mathcal{G}$, and $|\mathcal{G}|$ to denote the total number of nodes in $\mathcal{G}$. We use $\mathcal{G}_p$ and $\mathcal{G}_b$ to represent the geometric graph of the peptide and the binding site, respectively. In this work, the binding site incorporates residues on the target protein within 10Å distances to the peptide residues based on C_β atoms which alleviates leakage of the side-chain interactions. Note that the threshold (10Å) is chosen to be large to better reduce leakage of the peptide geometry.

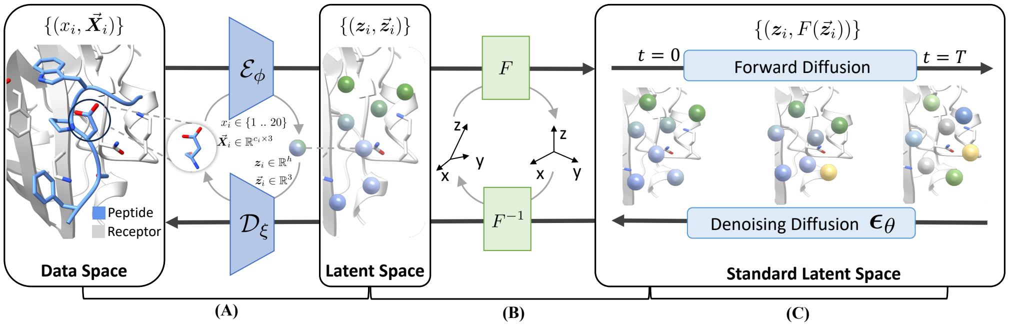

$\{(x_i, \vec{\boldsymbol{X}}_i)\}$
$\mathcal{E}_\phi$
$\{(\boldsymbol{z}_i, \vec{\boldsymbol{z}}_i)\}$
F
$\{(\boldsymbol{z}_i, F(\vec{\boldsymbol{z}}_i))\}$
$t = 0$
Forward Diffusion
$t = T$
$x_i \in \{1 .. 20\}$
$\vec{\boldsymbol{X}}_i \in \mathbb{R}^{c_i \times 3}$
$\boldsymbol{z}_i \in \mathbb{R}^h$
$\vec{\boldsymbol{z}}_i \in \mathbb{R}^3$
z
y
x
z
x
y
Peptide
Receptor
$\mathcal{D}_\xi$
F^{-1}
Denoising Diffusion $\boldsymbol{\epsilon}_\theta$
Data Space
Latent Space
Standard Latent Space
(A)
(B)
(C)

Figure 2: Overall architecture of PepGLAD. **(A)** Variational AutoEncoder (§3.2): compressing the sequence and the structure $\{(x_i, \vec{\boldsymbol{X}}_i)\}$ of the peptide into the latent space $\{(\boldsymbol{z}_i, \vec{\boldsymbol{z}}_i)\}$ with the encoder $\mathcal{E}_\phi$, and decoding the sequence and full-atom geometry from the latent states with the decoder $\mathcal{D}_\xi$. **(B)** Affine Transformation F (§3.3): projecting the geometry to approximately $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ via the receptor-specific affine transformation derived from the binding site, and recovering the data geometry with the inverse of F after the diffusion generative process. **(C)** Latent Diffusion (§3.4): jointly generating $\boldsymbol{z}_i$ and $\vec{\boldsymbol{z}}_i$ in the standard latent space.

Task Definition Given the binding site $\mathcal{G}_b$, we aim to obtain a generative model p_θ conforming to the distribution of binding peptides $q(\mathcal{G}_p|\mathcal{G}_b)$.

3.2 Variational AutoEncoder

The autoencoder [62] consists of an encoder $\mathcal{E}_\phi$ that encodes the peptide $\mathcal{G}_p$ in the presence of the binding site $\mathcal{G}_b$ into a latent state $\mathcal{G}_z$, and a decoder $\mathcal{D}_\xi$ that reconstructs the peptide from the latent state to obtain $\mathcal{G}'_p = \{(x'_i, \vec{\boldsymbol{X}}'_i)\}$. To encourage $\mathcal{E}_\phi$ to learn contextual representations of residues, we corrupt 25% of the residues in $\mathcal{G}_p$ with a [MASK] type to obtain $\tilde{\mathcal{G}}_p$ as the input:

$$\mathcal{G}_z = \mathcal{E}_\phi(\tilde{\mathcal{G}}_p, \mathcal{G}_b), \qquad \mathcal{G}'_p = \mathcal{D}_\xi(\mathcal{G}_z, \mathcal{G}_b), \tag{1}$$

where $\mathcal{G}_z = \{(\boldsymbol{z}_i, \vec{\boldsymbol{z}}_i)|i \in \mathcal{G}_p\}$ contains the latent states $\boldsymbol{z}_i \in \mathbb{R}^h$ ($h = 8$ in this paper) and $\vec{\boldsymbol{z}}_i \in \mathbb{R}^3$ sampled from the encoded distribution $\mathcal{N}(\boldsymbol{z}_i; \boldsymbol{\mu}_i, \boldsymbol{\sigma}_i)$ and $\mathcal{N}(\vec{\boldsymbol{z}}_i; \vec{\boldsymbol{\mu}}_i, \vec{\boldsymbol{\sigma}}_i)$ using the reparameterization trick [32]. We borrow the adaptive multi-channel equivariant encoder in dyMEAN [34] for both $\mathcal{E}_\phi$ and $\mathcal{D}_\xi$ to capture the full-atom geometry. In the decoder $\mathcal{D}_\xi$, we factorize the joint distribution of sequences and structures as follows:

$$p_\xi(x'_i, \vec{\boldsymbol{X}}'_i|\mathcal{G}_z, \mathcal{G}_b) = p_{\xi_1}(x'_i|\mathcal{G}_z, \mathcal{G}_b)p_{\xi_2}(\vec{\boldsymbol{X}}'_i|x'_i, \mathcal{G}_z, \mathcal{G}_b), \tag{2}$$

where the sequence is first decoded and then the all-atom geometry, initialized with replications of $\vec{\boldsymbol{z}}_i$, is reconstructed. The training objective of the autoencoder consists of the reconstruction loss $\mathcal{L}_{recon}$ and the KL divergence $\mathcal{L}_{KL}$ to constrain the latent space. The reconstruction loss includes cross entropy on the residue types, mean square error (MSE) on the full-atom structures, and an auxilary loss $\mathcal{L}_{aux}$ on bond lengths and angles [31]:

$$\mathcal{L}_{recon}(i) = H(p(x_i), p(x'_i)) + \mathrm{MSE}(\vec{\boldsymbol{X}}_i, \vec{\boldsymbol{X}}'_i) + \mathcal{L}_{aux}(i), \tag{3}$$

where H denotes cross entropy. We include details of $\mathcal{L}_{aux}$ in Appendix A. The KL divergence constrains $\boldsymbol{z}_i$ and $\vec{\boldsymbol{z}}_i$ with the prior $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and $\mathcal{N}(\vec{\boldsymbol{r}}_i, \boldsymbol{I})$, respectively, where $\vec{\boldsymbol{r}}_i$ denotes the coordinate of the alpha carbon ($\mathtt{C}_\alpha$) in node i:

$$\mathcal{L}_{KL}(i) = \lambda_1 \cdot D_{\mathrm{KL}}(\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \| \mathcal{N}(\boldsymbol{\mu}_i, \mathrm{diag}(\boldsymbol{\sigma}_i))) + \lambda_2 \cdot D_{\mathrm{KL}}(\mathcal{N}(\vec{\boldsymbol{r}}_i, \boldsymbol{I}) \| \mathcal{N}(\vec{\boldsymbol{\mu}}_i, \mathrm{diag}(\vec{\boldsymbol{\sigma}}_i))), \tag{4}$$

where D_{KL} denotes the KL divergence, λ_1 and λ_2 reweight the contraints on the sequence and the structure, respectively. $\mathcal{L}_{KL}$ prevents the scale of $\boldsymbol{z}_i$ from exploding and constrains $\vec{\boldsymbol{z}}_i$ around $\mathtt{C}_\alpha$ to retain necessary geometric information. Such regularization also helps ensure consistent scales between the peptide latent coordinates and the pocket, mitigating potential issues arising from their different levels of abstraction. Then we have the overall training objective of the variational autoencoder as follows:

$$\mathcal{L}_{AE} = \sum\nolimits_{i \in \mathcal{G}_p} (\mathcal{L}_{recon}(i) + \mathcal{L}_{KL}(i))/|\mathcal{G}_p|. \tag{5}$$

We have explored E(3)-invariant latent space, which appears to have difficulties in reconstructing the full-atom structures since it lacks information of geometric interactions with the pocket atoms (Appendix F).

3.3 Receptor-Specific Affine Transformation

With the latent space given by the autoencoder, we further exploit a standard space obtained from receptor-specific affine transformations, which enhances the transferability of diffusions on disparate binding sites (see Figure 1). Most peptides fold into complementary shape upon binding on the receptor [60, 41]. Thus, the target distribution is inherently characterized by the shape of the binding site. Given the wide disparity in binding geometries, directly implementing diffusion in the data space yields minimal transferability among different binding sites. To address this deficiency, we propose to implement the diffusion process on a shared standard space converted via an affine transformation derived from the binding site. Formally, denoting the $\mathtt{C}_\alpha$ coordinates of the residues in a given binding site $\mathcal{G}_b$ as $\vec{\boldsymbol{R}} \in \mathbb{R}^{3\times|\mathcal{G}_b|}$, we can derive their center $\vec{\boldsymbol{\mu}} = \mathbb{E}[\vec{\boldsymbol{R}}] \in \mathbb{R}^3$ and covariance $\vec{\boldsymbol{\Sigma}} = \mathrm{Cov}(\vec{\boldsymbol{R}}, \vec{\boldsymbol{R}}) \in \mathbb{R}^{3\times 3}$, so that these coordinates can be regarded as sampled from the distribution $\mathcal{N}(\vec{\boldsymbol{\mu}}, \vec{\boldsymbol{\Sigma}})$. We then calculate the Cholesky decomposition [19] of $\vec{\boldsymbol{\Sigma}}$:

$$\vec{\boldsymbol{\Sigma}} = \vec{\boldsymbol{L}}\vec{\boldsymbol{L}}^\top, \vec{\boldsymbol{L}} \in \mathbb{R}^{3\times 3}, \tag{6}$$

where $\vec{\boldsymbol{L}}$ is a lower triangular matrix. $\vec{\boldsymbol{L}}$ is unique [19] and invertible since the covariance matrix is a real-valued symmetric positive-definite matrix[2]. Then we can define the affine transformation $F : \mathbb{R}^3 \to \mathbb{R}^3$, which enables the projection of the geometry into the standard space approximating standard Gaussian $F(\vec{\boldsymbol{R}}) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. Further, we can easily obtain the inverse of F as:

$$F(\vec{\boldsymbol{x}}) = \vec{\boldsymbol{L}}^{-1}(\vec{\boldsymbol{x}} - \vec{\boldsymbol{\mu}}), \qquad F^{-1}(\vec{\boldsymbol{x}}) = \vec{\boldsymbol{L}}\vec{\boldsymbol{x}} + \vec{\boldsymbol{\mu}}. \tag{7}$$

With the above definitions, for each given binding site $\mathcal{G}_b$, we transform the geometry via the derived F to obtain the standard space, where the diffusion model is implemented, and recover the original geometry with F^{-1} (see Figure 2) after generation. Notably, we have the following proposition to ensure that the equivariance is maintained under the proposed affine transformation with scalarization-based equivariant GNNs [23, 16]:

Proposition 3.1. *Denote the invariant and equivariant outputs from a scalarization-based E(3)-equivariant GNN as* $f(\{\boldsymbol{h}_i, \vec{\boldsymbol{x}}_i\})$ *and* $\vec{f}(\{\boldsymbol{h}_i, \vec{\boldsymbol{x}}_i\})$*, respectively. With the definition of F in Eq. 7,* $\forall g \in E(3)$*, we have* $f(\{\boldsymbol{h}_i, F(\vec{\boldsymbol{x}}_i)\}) = f(\{\boldsymbol{h}_i, F_g(g \cdot \vec{\boldsymbol{x}}_i)\})$ *and* $g \cdot F^{-1}(\vec{f}(\{\boldsymbol{h}_i, F(\vec{\boldsymbol{x}}_i)\})) = F_g^{-1}(\vec{f}(\{\boldsymbol{h}_i, F_g(g \cdot \vec{\boldsymbol{x}}_i)\}))$*, where* F_g *is derived on the coordinates transformed by g. Namely, the E(3)-equivariance is preserved if we implement the GNN on the standard space and recover the original geometry from the outputs.*

The proof is in Appendix B. This is vital since it indicates the Markov kernel is E(3)-equivariant, and thus ensures the E(3)-invariance of the probability density in the diffusion process [70]. Note that our variational autoencoder (§ 3.2) and latent diffusion model (§ 3.4) are already designed to be equivariant even without the proposed affine transformation here. The purpose of defining such component is to encourage better generalization of the diffusion processes. Indeed, it is nontrivial to analyze whether such an implementation will break the equivariance of our workflow. Luckily, Proposition 3.1 manages to prove that scalarization-based equivariant networks [23], which is used in our autoencoder and diffusion model, are seamlessly compatible with such affine transformation, naturally preserving equivariance without any requirements of adaption.

3.4 Geometric Latent Diffusion Model

With the aforementioned preparations, the discrete residue types are encoded as continuous latent representations $\{\boldsymbol{z}_i\}$, and the full-atom geometry is also compressed and standardized into 3D vectors $\{\vec{\boldsymbol{z}}_i\} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. Therefore, we are ready to implement a diffusion model on the standard latent space to generate $\boldsymbol{z}_i$ and $\vec{\boldsymbol{z}}_i$. The forward diffusion process gradually adds noise to the data from $t = 0$ to $t = T$, resulting in the prior distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. The reverse diffusion process

[2] The binding site has at least 3 nonoverlapping nodes, namely $\mathrm{rank}(\vec{\boldsymbol{R}}) = 3$, thus we can ignore the corner case of semi-positive definite matrices.

generates data distribution by iteratively denosing the distribution from $t = T$ to $t = 0$. We denote $\vec{\boldsymbol{u}}_i^t = [\boldsymbol{z}_i^t, \vec{\boldsymbol{z}}_i^t]$ and $\mathcal{G}_z^t = \{(\boldsymbol{z}_i^t, \vec{\boldsymbol{z}}_i^t)\}$ as the intermediate state for node i and the entire peptide at time step t, respectively. For simplicity, we assume both $\mathcal{G}_z^t$ and the binding site $\mathcal{G}_b$ are already standardized via the transformation F_b in Eq. 7. Then we have the forward process as:

$$q(\vec{\boldsymbol{u}}_i^t|\vec{\boldsymbol{u}}_i^{t-1}) = \mathcal{N}(\vec{\boldsymbol{u}}_i^t; \sqrt{1-\beta^t} \cdot \vec{\boldsymbol{u}}_i^{t-1}, \beta^t \boldsymbol{I}), \tag{8}$$

$$q(\vec{\boldsymbol{u}}_i^t|\vec{\boldsymbol{u}}_i^0) = \mathcal{N}(\vec{\boldsymbol{u}}_i^t; \sqrt{\bar{\alpha}^t} \cdot \vec{\boldsymbol{u}}_i^0, (1-\bar{\alpha}^t) \boldsymbol{I}), \tag{9}$$

where β^t is the noise scale increasing with the timestep from 0 to 1 conforming to the cosine schedule [51], and $\bar{\alpha}^t = \prod_{s=1}^{s=t}(1-\beta^s)$. Then the state at timestep t can be sampled as:

$$\vec{\boldsymbol{u}}_i^t = \sqrt{\bar{\alpha}^t}\vec{\boldsymbol{u}}_i^0 + (1-\bar{\alpha}^t)\boldsymbol{\epsilon}_i, \tag{10}$$

where $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. Following Ho et al. [25], the reverse process can be defined with the reparameterization trick as:

$$p_\theta(\vec{\boldsymbol{u}}_i^{t-1}|\mathcal{G}_z^t, \mathcal{G}_b) = \mathcal{N}(\vec{\boldsymbol{u}}_i^{t-1}; \vec{\boldsymbol{\mu}}_\theta(\mathcal{G}_z^t, \mathcal{G}_b), \beta^t \boldsymbol{I}), \tag{11}$$

$$\vec{\boldsymbol{\mu}}_\theta(\mathcal{G}_z^t, \mathcal{G}_b) = \frac{1}{\sqrt{\alpha^t}}(\vec{\boldsymbol{u}}_i^t - \frac{\beta^t}{\sqrt{1-\bar{\alpha}^t}}\boldsymbol{\epsilon}_\theta(\mathcal{G}_z^t, \mathcal{G}_b, t)[i]), \tag{12}$$

where $\alpha^t = 1 - \beta^t$, and $\boldsymbol{\epsilon}_\theta$ is the denoising network also implemented with the equivariant adaptive multi-channel equivariant encoder in dyMEAN [34] to retain full-atom context of the binding site during generation and preserve the equivariance under affine transformations (Proposition 3.1). Finally, we have the objective at time step t as MSE between the predicted noise and the added noise in Eq. 10, as well as the overall training objective $\mathcal{L}_{LDM}$ as the expectation with respect to t:

$$\mathcal{L}_{LDM} = \mathbb{E}_{t\sim\text{Uniform}(1\ldots T)}[\sum\nolimits_i \|\boldsymbol{\epsilon}_i - \boldsymbol{\epsilon}_\theta(\mathcal{G}_z^t, \mathcal{G}_b, t)[i]\|^2/|\mathcal{G}_z^t|]. \tag{13}$$

3.5 Training and Sampling

Training The training of our PepGLAD can be divided into two phases where a variational autoencoder is first trained and then a diffusion model is trained on the standard latent space. We provide the overall training procedure in Algorithm 1 (see Appendix D). Note that a smooth and informative latent space is necessary for the consecutive training of the diffusion model, thus we resort to unsupervised data from protein fragments apart from the limited protein-peptide complexes for training the autoencoder, which we describe in Appendix E.

Sampling in Ordered Subspace The sampling procedure includes generative diffusion process on the standard latent states, recovering the original geometry with the inverse of F in Eq. 7, and decoding the sequence as well as the full-atom structure of the peptide (see Algorithm 2 in Appendix D). A problem here is that the unordered nature of graphs is not compatible with the sequential nature of peptides, thus the generated residues may have arbitrary permutation on the sequence order. Inspired by the concept of classifier-guided sampling [15], we first assign an arbitrary permutation $\mathcal{P}$ on the sequence order to the nodes. Then we steer the sampling procedure towards the desired subspace conforming to $\mathcal{P}$ with the following empirical classifier $p(1|\{\vec{\boldsymbol{z}}_i^t\})$, which estimates the probability of the current coordinates belonging to the desired subspace:

$$p(1|\{\vec{\boldsymbol{z}}_i^t\}) = \exp(-\sum\nolimits_{\mathcal{P}(i)-\mathcal{P}(j)=1} E(\|\vec{\boldsymbol{z}}_i^t - \vec{\boldsymbol{z}}_j^t\|)), \tag{14}$$

$$E(d) = \begin{cases} d - (\mu_d + 3\sigma_d), & d > \mu_d + 3\sigma_d, \\ (\mu_d - 3\sigma_d) - d, & d < \mu_d - 3\sigma_d, \\ 0, & \text{otherwise}, \end{cases} \tag{15}$$

where μ_d and σ_d are the mean and variance of the distances of adjacent residues in the latent space measured from the training set. Intuitively, this classifier gives higher confidence if the adjacent (defined by $\mathcal{P}$) residues are within reasonable distances aligning with the statistics from the training set. Nevertheless, the effect of the guidance is relatively minor, which is only a technical trick to enhance the robustness. We provide more details in Appendix G.

4 Experiments

4.1 Setup

Task We evaluate our PepGLAD and baselines on the following tasks: (1) **sequence-structure co-design** (§4.2) aims to generate both the sequence and the structure of the peptide given the specific binding site on the receptor (*i.e.* protein). (2) **Binding Conformation Generation** (§4.3) requires to generate the binding state of the peptide given its sequence and the binding site of interest.

Dataset We first extract all dimers from the Protein Data Bank (PDB) [5] and select the complexes with a receptor longer than 30 residues and a ligand between 4 to 25 residues [59]. Then we remove the duplicated complexes with the criterion that both the receptor and the peptide has a sequence identity over 90% [56], after which 6105 non-redundant complexes are obtained. To achieve the cross-target generalization test, we utilize the large non-redundant dataset (LNR) from Tsaban et al. [59] as the test set, which contains 93 protein-peptide complexes with canonical amino acids curated by domain experts. We then cluster the data by receptor with a sequence identity threshold of over 40%, and remove the complexes sharing the same clusters with those from the test set. Finally, the remaining data are randomly split based on clustering results into training and validation sets, yielding a new bechmark calling **PepBench**. Further, we exploit 70k unsupervised data from protein fragments (**ProtFrag**) to facilitate training of the variational autoencoder. We also implement a split on **PepBDB** [65] based on clustering results for evaluation. We show details and statistics of these datasets in Appendix E.

Baselines We first borrow three baselines from the antibody design domain. **HSRN** [30] autoregressively decodes the sequence while keeps refining the structure hierarchically, from the $\mathtt{C}_\alpha$ to other atoms. **dyMEAN** [34] is equipped with an full-atom geometric encoder and exploits iterative non-autoregressive generation. **DiffAb** [44] jointly diffuses on the categorical residue type, the coordinate of $\mathtt{C}_\alpha$ as well as the orientation of each residue. Next, we explore two baselines from the general protein design. **RFDiffusion** [64] exploits a pipeline that first generates the backbone via diffusion and then alternates between inverse folding [14] and structure refining based on a physical energy function [2]. **AlphaFold 2** [31] is the well-known model for protein folding, which also shows certain abilities on peptide conformation prediction [59]. We also include two traditional methods. **AnchorExtension** [26] designs peptides by first docking an existing scaffold to the binding site, and then optimizing the peptide with cycles of mutations guided by energy functions. **FlexPepDock** [42] is designed for flexible peptide docking via optimization in the landscape of a physical energy function [2]. Implementation details are provided in Appendix I.

4.2 Sequence-Structure Co-Design

Metrics A favorable generative model should produce diverse candidates while maintaining fidelity to the desired distribution. To comprehensively evaluate the models, we generate 40 candidates for each receptor and employ the following metrics: (1) **Diversity**. Inspired by [72], we measure the diversity via unique clusters of sequences and structures. Specifically, we hierarchically cluster the structures based on pair-wise root mean square deviation (RMSD) of $\mathtt{C}_\alpha$. The diversity of structures Div_{struct} is defined as the number of clusters versus the number of candidates. A similar procedure can be applied to the sequences to obtain Div_{seq}, utilizing the similarity [36] derived from alignment [24]. Then the co-design diversity is $\sqrt{\text{Div}_{seq}\text{Div}_{struct}}$. (2) **Consistency**. We measure how well the models learn the 1D&3D joint distribution by the sequence-structure consistency, quantified via Cramér's V [12] association between the clustering labels (as in Diversity) of the sequences and the structures. High consistency indicates that candidates with similar sequences also have similar structures, implying that the generative model effectively captures the dependency between 1D and 3D. (3) $\mathbf{\Delta G}$. Aligned with the literature [34, 44], we employ the binding energy (kcal/mol) provided by Rosetta [2], a widely-used suite for biomolecular modeling with physical energy functions, to evaluate the binding affinity of the generated candidates. Lower ΔG indicates stronger binding between the peptide and the target. (4) **Success**. We report the proportion of successful designs (*i.e.* $\Delta G < 0$, indicating no severe atomic clashes or twisted conformations) among all the candidates.

Table 1: Evaluation on sequence-structure co-design. On each target, 40 candidates are generated for evaluation. Div. and Con. are abbreviations for diversity and consistency, respectively.

Model	PepBench				PepBDB			
	Div.(↑)	Con.(↑)	ΔG(↓)	Success	Div.(↑)	Con.(↑)	ΔG(↓)	Success
Test Set	-	-	-35.25	95.70%	-	-	-35.96	95.79%
HSRN[3]	0.158	0.0	≥ 0	10.46%	0.111	0.0	≥ 0	10.86%
dyMEAN	0.150	0.0	-2.26	14.60%	0.150	0.0	-1.92	6.26%
DiffAb	0.427	0.670	-21.20	49.87%	0.269	0.463	-18.40	41.45%
PepGLAD (ours)	**0.506**	**0.789**	**-21.94**	**55.97%**	**0.692**	**0.923**	**-21.53**	**48.47%**

For all metrics except ΔG, we first compute values for each receptor individually and then average the results across different receptors. For ΔG, we identify the best candidate on each receptor as the outputs and report the median value across different receptors. Details about the metrics are provided in Appendix H, including discussion on AAR (Appendix H.1) and consistency (Appendix H.2).

Results Table 1 illustrates that our PepGLAD generates significantly more diversified and consistent peptides with better binding energy and success rates compared to the baselines. When benchmarking HSRN, dyMEAN, and DiffAb, which perform well on antibody CDR design, we observe a notable performance gap between non-diffusion baselines (*i.e.* HSRN, dyMEAN) and the diffusion-based baseline (*i.e.* DiffAb), suggesting the higher complexity in peptide design and the need for stronger modeling capabilities. Compared to DiffAb, which operates on categorical residue types, C_α coordinates and orientations, our PepGLAD (1) better captures the dependency between sequence and structure, as indicated by higher diversity and consistency, since diffusion is implemented on the latent space where the representation of sequence and structure are nicely correlated by the autoencoder; (2) more effectively captures the intricate protein-peptide interactions, demonstrated by better ΔG and success rates, since we leverage the full-atom context of the binding site and enhances generalization capability by converting the geometry into a standard space. We showcase two candidates designed by our PepGLAD with favorable binding energy given by Rosetta in Figure 3. Furthermore, the diversity within successful designs is 0.632, which is higher than that of all designs (0.506), indicating the high structural flexibility of peptides upon successful binding.

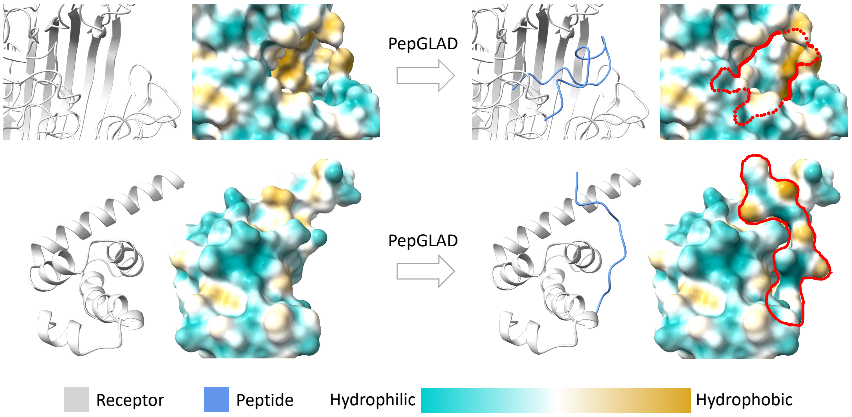


Figure 3: **Top**: A generated candidate confined within the binding site (PDB=4cu4, ΔG=-34.21). **Bottom**: A generated candidate with complementary shape to the binding site (PDB=3pkn, ΔG=-33.32). Both candidates form compact interactions at the interface.

We also evaluate our PepGLAD against two sophisticated pipeline systems in Table 2. The traditional method (*i.e.* AnchorExtension) is limited by low efficiency, thus we can only afford outputting 10 candidates for each receptor. For a relatively fair comparison with RFDiffusion, we refine the structure of the generated candidates using the empirical force field in RFDiffusion. However, the comparison may still disadvantage our PepGLAD, given that RFDiffusion is finetuned from a model pretrained on a large-scale dataset [64]. Nevertheless, as demonstrated in Table 2, our model still exhibits marvelous superiority on diversity, consistency, and success rate, while achieving competitive binding energy ΔG, with obviously higher efficiency.

Table 2: Evaluation on sequence-structure co-design with two well-established systems. Time cost is measured as the total time spent divided by the number of designed candiates.

Model	Div.(↑)	Con.(↑)	ΔG(↓)	Success	Time
AnchorExtension	0.245	0.423	-26.80	84.30%	735s
RFDiffusion	0.259	0.696	**-33.82**	79.68%	61s
PepGLAD (ours)	**0.506**	**0.789**	-29.36	**92.82%**	**3s**

[3] HSRN and dyMEAN generate homogeneous structures that are clustered together yet still sample very different sequences, leading to zero association between squence and structure.

4.3 Binding Conformation Generation

Metrics For each receptor, we generate 10 candidates and report the median value of the following metrics across different receptors to measure how well the generated distribution can recover the reference conformation: (1) **RMSD**$_{\mathtt{C}_\alpha}$: Root mean square deviation on the coordinates of $\mathtt{C}_\alpha$ between a candidate and a reference structure with the unit Å. (2) **RMSD**$_{\text{atom}}$: RMSD on all atoms to measure the quality of the full-atom geometry. (3) **DockQ** [4]. A comprehensive metric evaluating the full-atom similarity on the interface between a candidate and a reference complex. It ranges from 0 to 1, with values above 0.23 and 0.49 considered as acceptable and medium quality, respectively.

Table 3: Evaluation on binding conformation generation. On each target, 10 candidates are generated to calculate the optimal recall of the reference conformation.

Model	PepBench			PepBDB		
	RMSD$_{\mathtt{C}_\alpha}$(↓)	RMSD$_{\text{atom}}$(↓)	DockQ(↑)	RMSD$_{\mathtt{C}_\alpha}$(↓)	RMSD$_{\text{atom}}$(↓)	DockQ(↑)
FlexPepDock	6.43	7.52	0.393	-	-	-
AlphaFold 2	8.49	9.20	0.355	-	-	-
dyMEAN	7.96	8.35	0.374	17.64	17.56	0.142
HSRN	6.02	7.59	0.508	9.28	9.72	0.394
DiffAb	4.23	7.60	0.586	13.96	13.12	0.236
PepGLAD (ours)	**4.09**	**5.30**	**0.592**	**8.87**	**8.62**	**0.403**

Results As shown in Table 3, our PepGLAD surpasses all the baselines in terms of both RMSD$_{\mathtt{C}_\alpha}$ and DockQ by a large margin, highlighting the superiority of incorporating the full-atom context and the binding-site shape into the latent diffusion process. Additionally, we present the distribution of the best RMSD$_{\mathtt{C}_\alpha}$ on different test receptors using box plots and showcase a generated conformation highly resembling the reference in Figure 4. The distribution reveals that our model achieves favorable performance on RMSD$_{\mathtt{C}_\alpha}$ with lower variance on the test set compared to other baselines, exhibiting robust generalization ability across disparate binding sites.

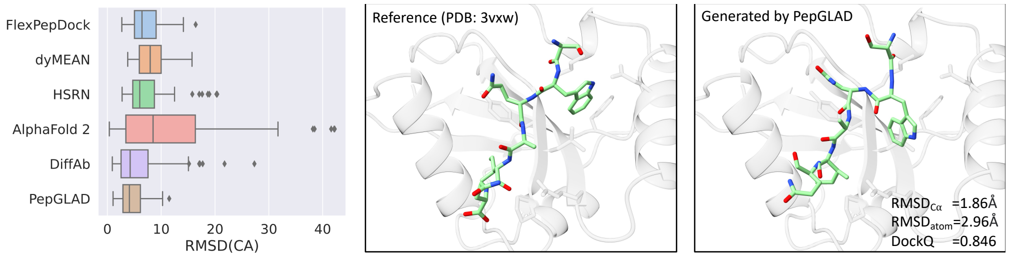


Figure 4: The distribution of RMSD$_{\mathtt{C}_\alpha}$ on the test set of PepBench and a visualized sample.

5 Analysis

We conduct the following ablations: the full-atom geometry (**Full-Atom**); the affine transformation (**Affine**); the unsupervised data from protein fragments (**ProtFrag**) and the mask policy (**Mask**) when training the autoencoder. Note that generative performance is assessed from various aspects, and improvement in one aspect at the disproportionate expense of others might be meaningless. Thus, we additionally compute the average of all the metrics to evaluate the comprehensive effect of each module, where ΔG is normalized by the statistics on the test set. Table 4 demonstrates the following observations: (1) Discarding the full-atom context results in a significant degradation on all metrics, especially the success rate, implying the necessity of the full-atom context in capturing the intricate protein-peptide interactions; (2) Implementing the diffusion directly on the data space without the proposed affine transformation incurs a notably adverse impact on all metrics, indicating the remarkable enhancement on the generalization capability made by the affine transformation; (3) Training without the unsupervised data leads to a less informative latent space, exerting a negative effect on the binding energy and success rate; (4) Removal of the mask policy reduces the correlation between sequence and structure in the latent space, thus harms the consistency.

Table 4: Ablations on different components. Avg. computes the average of all metrics, where ΔG is first normalized by the median value of the references on test set.

Ablations	Div.(↑)	Con.(↑)	ΔG(↓)	Success	Avg.
PepGLAD	0.506	0.789	-21.94	55.97%	**0.619**
w/o Full-Atom	0.441	0.751	-20.87	51.18%	0.574
w/o Affine	0.450	0.740	-19.08	52.39%	0.564
w/o ProtFrag	0.535	0.760	-20.16	52.15%	0.597
w/o Mask	0.422	0.741	-20.45	57.44%	0.579

6 Limitations

Despite the promising results, we acknowledge several limitations which might be addressed by future work. First, the binding affinity assessment relies on the Rosetta scoring function as a proxy for wetlab experiments. There may be discrepancies between the predicted and actual binding energies. The ultimate test of a peptide utility is its performance *in vivo*, which is too costly for large-scale evaluation. Nevertheless, this is a problem confronting the entire community, and we hope future research might propose more reliable *in silico* proxies to bridge the gap. Second, while this paper addresses peptide design from the aspect of proteins, it might also be reasonable to think from the aspect of small molecules if the peptides are short enough. Under such circumstances, it is also beneficial to further explore counterparts of methods for small molecule design [43, 52, 39], which we leave for future work.

7 Conclusion

In this paper, we first assemble a dataset from Protein Data Bank (PDB) and literature to benchmark generative models on target-specific peptide design in terms of diversity, consistency, and binding energy. Subsequently, we propose PepGLAD, a powerful diffusion-based model for full-atom peptide design. In particular, we explore diffusion on the latent space where the sequence and the full-atom structure are jointly encoded by a variational autoencoder. We further propose a receptor-specific affine transformation technique to project variable geometries in the data space into a standard space, which enhances the transferability of diffusion processes on disparate binding sites. Our PepGLAD outperforms the existing models on sequence-structure co-design and binding conformation generation, exhibiting high generalization across diverse binding sites. Our work represents a pioneering effort in the exploration of deep generative models for simultaneous design of 1D sequences and 3D structures of peptides, which could inspire future research in this field.

Software and Data

The curated PepBench and ProtFrag are available at `https://zenodo.org/records/13373108`. The codes for our PepGLAD are open-sourced at `https://github.com/THUNLP-MT/PepGLAD`.

Impact Statements

This paper aims to advance the field of peptide design through the construction of a benchmark and the development of a novel latent diffusion model, PepGLAD, which addresses key limitations in current methods. Our work represents a step forward in computational peptide design, with the potential to impact both scientific research and practical applications in various domains. For instance, more precise peptide design could lead to enhanced drugs in the pharmaceutical industry, and could facilitate the creation of new biomaterials, sensors, and other innovative technologies in biology and materials science. We wish our paper could inspire future research in this field.

Acknowledgments

This work is jointly supported by the National Key R&D Program of China (No.2022ZD0160502), the National Natural Science Foundation of China (No. 61925601, No. 62376276, No. 62276152), and Beijing Nova Program (20230484278).

References

[1] J. Adolf-Bryfogle, O. Kalyuzhniy, M. Kubitz, B. D. Weitzner, X. Hu, Y. Adachi, W. R. Schief, and R. L. Dunbrack Jr. Rosettaantibodydesign (rabd): A general framework for computational antibody design. *PLoS computational biology*, 14(4):e1006112, 2018.

[2] R. F. Alford, A. Leaver-Fay, J. R. Jeliazkov, M. J. O'Meara, F. P. DiMaio, H. Park, M. V. Shapovalov, P. D. Renfrew, V. K. Mulligan, K. Kappel, et al. The rosetta all-atom energy function for macromolecular modeling and design. *Journal of chemical theory and computation*, 13(6):3031–3048, 2017.

[3] N. Anand and T. Achim. Protein structure and sequence generation with equivariant denoising diffusion probabilistic models. *arXiv preprint arXiv:2205.15019*, 2022.

[4] S. Basu and B. Wallner. Dockq: a quality measure for protein-protein docking models. *PloS one*, 11(8):e0161879, 2016.

[5] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The protein data bank. *Nucleic acids research*, 28(1):235–242, 2000.

[6] G. Bhardwaj, V. K. Mulligan, C. D. Bahl, J. M. Gilmore, P. J. Harvey, O. Cheneval, G. W. Buchko, S. V. Pulavarti, Q. Kaas, A. Eletsky, et al. Accurate de novo design of hyperstable constrained peptides. *Nature*, 538(7625):329–335, 2016.

[7] J. Bose, T. Akhound-Sadegh, K. FATRAS, G. Huguet, J. Rector-Brooks, C.-H. Liu, A. C. Nica, M. Korablyov, M. M. Bronstein, and A. Tong. Se (3)-stochastic flow matching for protein backbone generation. In *The Twelfth International Conference on Learning Representations*, 2023.

[8] P. Bryant and A. Elofsson. Evobind: in silico directed evolution of peptide binders with alphafold. *bioRxiv*, pages 2022–07, 2022.

[9] L. Cao, B. Coventry, I. Goreshnik, B. Huang, W. Sheffler, J. S. Park, K. M. Jude, I. Marković, R. U. Kadam, K. H. Verschueren, et al. Design of protein-binding proteins from the target structure alone. *Nature*, 605(7910):551–560, 2022.

[10] C. Chothia and J. Janin. Principles of protein–protein recognition. *Nature*, 256(5520):705–708, 1975.

[11] P. J. Cock, T. Antao, J. T. Chang, B. A. Chapman, C. J. Cox, A. Dalke, I. Friedberg, T. Hamelryck, F. Kauff, B. Wilczynski, et al. Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422, 2009.

[12] H. Cramér. *Mathematical methods of statistics*, volume 26. Princeton university press, 1999.

[13] P. Das, T. Sercu, K. Wadhawan, I. Padhi, S. Gehrmann, F. Cipcigan, V. Chenthamarakshan, H. Strobelt, C. Dos Santos, P.-Y. Chen, et al. Accelerated antimicrobial discovery via deep generative models and molecular dynamics simulations. *Nature Biomedical Engineering*, 5(6): 613–623, 2021.

[14] J. Dauparas, I. Anishchenko, N. Bennett, H. Bai, R. J. Ragotte, L. F. Milles, B. I. Wicky, A. Courbet, R. J. de Haas, N. Bethel, et al. Robust deep learning–based protein sequence design using proteinmpnn. *Science*, 378(6615):49–56, 2022.

[15] P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[16] A. Duval, S. V. Mathis, C. K. Joshi, V. Schmidt, S. Miret, F. D. Malliaros, T. Cohen, P. Lio, Y. Bengio, and M. Bronstein. A hitchhiker's guide to geometric gnns for 3d atomic systems. *arXiv preprint arXiv:2312.07511*, 2023.

[17] K. Fosgerau and T. Hoffmann. Peptide therapeutics: current status and future directions. *Drug discovery today*, 20(1):122–128, 2015.

[18] C. Fu, K. Yan, L. Wang, W. Y. Au, M. C. McThrow, T. Komikado, K. Maruhashi, K. Uchino, X. Qian, and S. Ji. A latent diffusion model for protein structure generation. In *Learning on Graphs Conference*, pages 29–1. PMLR, 2024.

[19] G. H. Golub and C. F. Van Loan. *Matrix computations*. JHU press, 2013.

[20] C. Grathwohl and K. Wüthrich. The x-pro peptide bond as an nmr probe for conformational studies of flexible linear peptides. *Biopolymers: Original Research on Biomolecules*, 15(10): 2025–2041, 1976.

[21] J. Guan, W. W. Qian, X. Peng, Y. Su, J. Peng, and J. Ma. 3d equivariant diffusion for target-aware molecule generation and affinity prediction. In *The Eleventh International Conference on Learning Representations*, 2022.

[22] K. Guruprasad, B. B. Reddy, and M. W. Pandit. Correlation between stability of a protein and its dipeptide composition: a novel approach for predicting in vivo stability of a protein from its primary sequence. *Protein Engineering, Design and Selection*, 4(2):155–161, 1990.

[23] J. Han, Y. Rong, T. Xu, and W. Huang. Geometrically equivariant graph neural networks: A survey. *arXiv preprint arXiv:2202.07230*, 2022.

[24] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919, 1992.

[25] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[26] P. Hosseinzadeh, P. R. Watson, T. W. Craven, X. Li, S. Rettie, F. Pardo-Avila, A. K. Bera, V. K. Mulligan, P. Lu, A. S. Ford, et al. Anchor extension: a structure-guided approach to design cyclic peptides targeting enzyme active sites. *Nature Communications*, 12(1):3384, 2021.

[27] J. Ingraham, V. Garg, R. Barzilay, and T. Jaakkola. Generative models for graph-based protein design. *Advances in neural information processing systems*, 32, 2019.

[28] J. B. Ingraham, M. Baranov, Z. Costello, K. W. Barber, W. Wang, A. Ismail, V. Frappier, D. M. Lord, C. Ng-Thow-Hing, E. R. Van Vlack, et al. Illuminating protein space with a programmable generative model. *Nature*, pages 1–9, 2023.

[29] W. Jin, J. Wohlwend, R. Barzilay, and T. Jaakkola. Iterative refinement graph neural network for antibody sequence-structure co-design. *arXiv preprint arXiv:2110.04624*, 2021.

[30] W. Jin, R. Barzilay, and T. Jaakkola. Antibody-antigen docking and design via hierarchical equivariant refinement. *arXiv preprint arXiv:2207.06616*, 2022.

[31] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[32] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[33] X. Kong, W. Huang, and Y. Liu. Conditional antibody design as 3d equivariant graph translation. *arXiv preprint arXiv:2208.06073*, 2022.

[34] X. Kong, W. Huang, and Y. Liu. End-to-end full-atom antibody design. *arXiv preprint arXiv:2302.00203*, 2023.

[35] A. C.-L. Lee, J. L. Harris, K. K. Khanna, and J.-H. Hong. A comprehensive review on current advances in peptide drug development and design. *International journal of molecular sciences*, 20(10):2383, 2019.

[36] Y. Lei, S. Li, Z. Liu, F. Wan, T. Tian, S. Li, D. Zhao, and J. Zeng. A deep-learning framework for multi-level peptide–protein interaction prediction. *Nature communications*, 12(1):5465, 2021.

[37] J. K. Leman, B. D. Weitzner, S. M. Lewis, J. Adolf-Bryfogle, N. Alam, R. F. Alford, M. Aprahamian, D. Baker, K. A. Barlow, P. Barth, et al. Macromolecular modeling and design in rosetta: recent methods and frameworks. *Nature methods*, 17(7):665–680, 2020.

[38] J. Li, C. Cheng, Z. Wu, R. Guo, S. Luo, Z. Ren, J. Peng, and J. Ma. Full-atom peptide design based on multi-modal flow matching. In *Forty-first International Conference on Machine Learning*, 2024.

[39] H. Lin, Y. Huang, O. Zhang, Y. Liu, L. Wu, S. Li, Z. Chen, and S. Z. Li. Functional-group-based diffusion for pocket-specific molecule generation and elaboration. *Advances in Neural Information Processing Systems*, 36, 2024.

[40] H. Lin, O. Zhang, H. Zhao, D. Jiang, L. Wu, Z. Liu, Y. Huang, and S. Z. Li. Ppflow: Target-aware peptide design with torsional flow matching. In *Forty-first International Conference on Machine Learning*, 2024.

[41] N. London, D. Movshovitz-Attias, and O. Schueler-Furman. The structural basis of peptide-protein binding strategies. *Structure*, 18(2):188–199, 2010.

[42] N. London, B. Raveh, E. Cohen, G. Fathi, and O. Schueler-Furman. Rosetta flexpepdock web server—high resolution modeling of peptide–protein interactions. *Nucleic acids research*, 39 (suppl_2):W249–W253, 2011.

[43] S. Luo, J. Guan, J. Ma, and J. Peng. A 3d generative model for structure-based drug design. *Advances in Neural Information Processing Systems*, 34:6229–6239, 2021.

[44] S. Luo, Y. Su, X. Peng, S. Wang, J. Peng, and J. Ma. Antigen-specific antibody design and optimization with diffusion-based generative models for protein structures. *Advances in Neural Information Processing Systems*, 35:9754–9767, 2022.

[45] K. Martinkus, J. Ludwiczak, K. Cho, W.-C. Lian, J. Lafrance-Vanasse, I. Hotzel, A. Rajpal, Y. Wu, R. Bonneau, V. Gligorijevic, et al. Abdiffuser: Full-atom generation of in-vitro functioning antibodies. *arXiv preprint arXiv:2308.05027*, 2023.

[46] P. M. Martins, L. H. Santos, D. Mariano, F. C. Queiroz, L. L. Bastos, I. d. S. Gomes, P. H. Fischer, R. E. Rocha, S. A. Silveira, L. H. de Lima, et al. Propedia: a database for protein–peptide identification based on a hybrid clustering algorithm. *BMC bioinformatics*, 22:1–20, 2021.

[47] S. Mitternacht. Freesasa: An open source c library for solvent accessible surface area calculations. *F1000Research*, 5, 2016.

[48] A. T. Muller, J. A. Hiss, and G. Schneider. Recurrent neural network model for constructive peptide design. *Journal of chemical information and modeling*, 58(2):472–479, 2018.

[49] M. Muttenthaler, G. F. King, D. J. Adams, and P. F. Alewood. Trends in peptide drug discovery. *Nature reviews Drug discovery*, 20(4):309–325, 2021.

[50] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970.

[51] A. Q. Nichol and P. Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

[52] X. Peng, S. Luo, J. Guan, Q. Xie, J. Peng, and J. Ma. Pocket2mol: Efficient molecular sampling based on 3d protein pockets. In *International Conference on Machine Learning*, pages 17644–17655. PMLR, 2022.

[53] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[54] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

[55] Y. Song and S. Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

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

[57] S. Swanson, V. Sivaraman, G. Grigoryan, and A. E. Keating. Tertiary motifs as building blocks for the design of protein-binding peptides. *Protein Science*, 31(6):e4322, 2022.

[58] B. L. Trippe, J. Yim, D. Tischer, T. Broderick, D. Baker, R. Barzilay, and T. Jaakkola. Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. *arXiv preprint arXiv:2206.04119*, 2022.

[59] T. Tsaban, J. K. Varga, O. Avraham, Z. Ben-Aharon, A. Khramushin, and O. Schueler-Furman. Harnessing protein folding neural networks for peptide–protein docking. *Nature communications*, 13(1):176, 2022.

[60] P. Vanhee, A. M. van der Sloot, E. Verschueren, L. Serrano, F. Rousseau, and J. Schymkowitz. Computational design of peptide ligands. *Trends in biotechnology*, 29(5):231–239, 2011.

[61] Y. Verma, M. Heinonen, and V. Garg. Abode: Ab initio antibody design using conjoined odes. *arXiv preprint arXiv:2306.01005*, 2023.

[62] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, and L. Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.

[63] D. Wang, Z. Wen, F. Ye, H. Zhou, and L. Li. Accelerating antimicrobial peptide discovery with latent sequence-structure model. *arXiv preprint arXiv:2212.09450*, 2022.

[64] J. L. Watson, D. Juergens, N. R. Bennett, B. L. Trippe, J. Yim, H. E. Eisenach, W. Ahern, A. J. Borst, R. J. Ragotte, L. F. Milles, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, 2023.

[65] Z. Wen, J. He, H. Tao, and S.-Y. Huang. Pepbdb: a comprehensive structural database of biological peptide–protein interactions. *Bioinformatics*, 35(1):175–177, 2019.

[66] K. E. Wu, K. K. Yang, R. van den Berg, S. Alamdari, J. Y. Zou, A. X. Lu, and A. P. Amini. Protein structure generation via folding diffusion. *Nature communications*, 15(1):1059, 2024.

[67] H. Xia, J. McMichael, M. Becker-Hapak, O. C. Onyeador, R. Buchli, E. McClain, P. Pence, S. Supabphol, M. M. Richters, A. Basu, et al. Computational prediction of mhc anchor locations guides neoantigen identification and prioritization. *Science immunology*, 8(82):eabg2200, 2023.

[68] X. Xie, P. A. Valiente, and P. M. Kim. Helixgan a deep-learning methodology for conditional de novo design of α-helix structures. *Bioinformatics*, 39(1):btad036, 2023.

[69] X. Xie, P. A. Valiente, J. Kim, and P. M. Kim. Helixdiff, a score-based diffusion model for generating all-atom α-helical structures. *ACS Central Science*, 10(5):1001–1011, 2024.

[70] M. Xu, L. Yu, Y. Song, C. Shi, S. Ermon, and J. Tang. Geodiff: A geometric diffusion model for molecular conformation generation. *arXiv preprint arXiv:2203.02923*, 2022.

[71] M. Xu, A. S. Powers, R. O. Dror, S. Ermon, and J. Leskovec. Geometric latent diffusion models for 3d molecule generation. In *International Conference on Machine Learning*, pages 38592–38610. PMLR, 2023.

[72] J. Yim, B. L. Trippe, V. De Bortoli, E. Mathieu, A. Doucet, R. Barzilay, and T. Jaakkola. Se (3) diffusion model with application to protein backbone generation. *arXiv preprint arXiv:2302.02277*, 2023.

[73] Y. Zhang, Z. Zhang, B. Zhong, S. Misra, and J. Tang. Diffpack: A torsional diffusion model for autoregressive protein side-chain packing. *arXiv preprint arXiv:2306.01794*, 2023.

A Reconstruction of Full-Atom Geometry

A.1 Auxilary Loss for Training the AutoEncoder

To better recover the all-atom geometry, we employ an auxilary structural loss similar to the violation loss in Jumper et al. [31] including the supervision on $\texttt{C}_\alpha$ coordinates, bond lengths, and side-chain dihedral angles. First, since the $\texttt{C}_\alpha$ is critical in deciding the global geometry of the peptide, we exert additional loss on its coordinates to distinguish it from other atoms:

$$\mathcal{L}_{CA}(i) = \mathrm{MSE}(\vec{\boldsymbol{r}}_i, \vec{\boldsymbol{r}}'_i), \tag{16}$$

where $\vec{\boldsymbol{r}}'_i$ and $\vec{\boldsymbol{r}}_i$ are the reconstructed and the ground truth coordinates of $\texttt{C}_\alpha$ in node i. Next, we implement L1 loss on the bond lengths:

$$\mathcal{L}_{bond}(i) = \sum\nolimits_{b \in \mathcal{B}(i)} |b - b'|/|\mathcal{B}(i)|, \tag{17}$$

where $\mathcal{B}(i)$ includes all chemical bonds in node i, and b' denotes the reconstructed bond length. For simplicity, he bonds between residues are included into the bonds of the former residue. Finally, we supervise on the χ_1 to χ_4 side-chain dihedral angles [73]:

$$\mathcal{L}_{angle}(i) = \sum\nolimits_{\chi \in \mathcal{A}(i)} |\chi - \chi'|/|\mathcal{A}(i)|, \tag{18}$$

where $\mathcal{A}(i)$ includes all side-chain dihedral angles in node i, χ' and χ denotes the reconstructed and the ground truth angles, respectively. The overall auxilary loss for node i is then given by:

$$\mathcal{L}_{aux}(i) = \lambda_{CA}\mathcal{L}_{CA}(i) + \lambda_{bond}\mathcal{L}_{bond}(i) + \lambda_{angle}\mathcal{L}_{angle}(i), \tag{19}$$

where we set $\lambda_{CA} = 1.0, \lambda_{bond} = 1.0, \lambda_{angle} = 0.5$ in our experiments. We find that it is necessary to set λ_{angle} with a relatively small value to make the training process stable.

A.2 Idealization of Local Geometry

Preserving atom instances enhances the modeling of side chain interactions but introduces challenges due to potential twisting in local geometry. While in sequence-structure co-design, samples undergo fast relax by physical force field to ensure a valid local geometry, in binding conformation generation, we use an alignment technique to place the idealized side chains on the generated atom instances. Specifically, the idealized side chain can be represented as at most 4 dihedral angles (χ-angles), treating fragments like phenyl group as rigid bodies. Suppose there are n_i χ-angles and c_i atoms in node i, we can define a function to map from the χ-angles with the backbone coordinates $\vec{\boldsymbol{B}}_i \in \mathbb{R}^{4\times 3}$ to atom instances as $M_i : \mathbb{R}^{n_i} \times \mathbb{R}^{4\times 3} \to \mathbb{R}^{c_i \times 3}$, which is luckily differentiable [28]. Thus we can optimize χ-angles via gradient descent to minimize the MSE between the coordinates constructed from χ-angles and those generated by the model:

$$\boldsymbol{\chi}_i^* = \arg\min\nolimits_{\boldsymbol{\chi} \in [0, 2\pi]^{n_i}} \|M_i(\boldsymbol{\chi}, \vec{\boldsymbol{B}}_i) - \vec{\boldsymbol{X}}_i\|^2, \tag{20}$$

where $\vec{\boldsymbol{X}}_i \in \mathbb{R}^{c_i \times 3}$ denotes the coordinates of c_i atom instances in node i generated by the model. Now it should be easy to construct an idealized side chain maintaining fidelity to the generated atom instances by $M(\boldsymbol{\chi}_i^*, \vec{\boldsymbol{B}}_i)$. In the experiments of conformation generation on PepBench, such idealization gives slightly better DockQ and $\mathrm{RMSD}_{\mathrm{atom}}$ than Rosetta side-chain packing algorithm but with higher efficiency.

B Proof of Proposition 3.1

Proposition 3.1. *Denote the invariant and equivariant outputs from a scalarization-based E(3)-equivariant GNN as $f(\{\boldsymbol{h}_i, \vec{\boldsymbol{x}}_i\})$ and $\vec{f}(\{\boldsymbol{h}_i, \vec{\boldsymbol{x}}_i\})$, respectively. With the definition of F in Eq. 7, $\forall g \in E(3)$, we have $f(\{\boldsymbol{h}_i, F(\vec{\boldsymbol{x}}_i)\}) = f(\{\boldsymbol{h}_i, F_g(g \cdot \vec{\boldsymbol{x}}_i)\})$ and $g \cdot F^{-1}(\vec{f}(\{\boldsymbol{h}_i, F(\vec{\boldsymbol{x}}_i)\})) = F_g^{-1}(\vec{f}(\{\boldsymbol{h}_i, F_g(g \cdot \vec{\boldsymbol{x}}_i)\}))$, where F_g is derived on the coordinates transformed by g. Namely, the E(3)-equivariance is preserved if we implement the GNN on the standard space and recover the original geometry from the outputs.*

For simplicity, given F derived from a set of coordinates $\{\vec{\boldsymbol{x}}_i\}$ following Eq. 7, we use F_g to indicate the affine transformation derived from $\{g \cdot \vec{\boldsymbol{x}}_i\}$, where $g \in E(3)$. Additionally, we keep the terminology "standard space" for describing the space after the data-specific affine transformation F. We begin by proving a key lemma, that is, the E(3)-invariance of distances between two nodes in the standard space converted by F:

Lemma C.1. *Given two nodes i and j in the geometric graph $\mathcal{G}$, denoting their coordinates as $\vec{\boldsymbol{x}}_i$ and $\vec{\boldsymbol{x}}_j$, their distance in the standard space is E(3)-invariant. Namely, $\forall g \in E(3)$, $\|F(\vec{\boldsymbol{x}}_i) - F(\vec{\boldsymbol{x}}_j)\| = \|F_g(g \cdot \vec{\boldsymbol{x}}_i) - F_g(g \cdot \vec{\boldsymbol{x}}_j)\|$.*

Proof. $\forall g \in E(3)$, g can be instantiated as an orthogonal matrix $\boldsymbol{Q} \in O(3)$ (including rotation and reflection), and a translation vector $\vec{\boldsymbol{t}} \in \mathbb{R}^3$. Denoting all coordinates in the geometric graph as $\vec{\boldsymbol{X}} \in \mathbb{R}^{3\times|\mathcal{G}|}$, and the number of nodes in $\mathcal{G}$ as n, we can derive the E(3)-equivariance of the expectation of the coordinates:

$$\mathbb{E}[g \cdot \vec{\boldsymbol{X}}] = \frac{1}{n}\sum_i g \cdot \vec{\boldsymbol{x}}_i = \frac{1}{n}\sum_i (\boldsymbol{Q}\vec{\boldsymbol{x}}_i + \vec{\boldsymbol{t}}) \tag{21}$$

$$= \frac{1}{n}\boldsymbol{Q}(\sum_i \vec{\boldsymbol{x}}_i) + \vec{\boldsymbol{t}} = \boldsymbol{Q}\mathbb{E}[\vec{\boldsymbol{X}}] + \vec{\boldsymbol{t}} \tag{22}$$

$$= g \cdot \mathbb{E}[\vec{\boldsymbol{X}}]. \tag{23}$$

With the E(3)-equivariance of the expectation, it is easy to derive the following equation on the covariance matrix:

$$\mathrm{Cov}(g \cdot \vec{\boldsymbol{X}}, g \cdot \vec{\boldsymbol{X}}) = \frac{1}{n-1}(g \cdot \vec{\boldsymbol{X}} - \mathbb{E}[g \cdot \vec{\boldsymbol{X}}])(g \cdot \vec{\boldsymbol{X}} - \mathbb{E}[g \cdot \vec{\boldsymbol{X}}])^\top \tag{24}$$

$$= \frac{1}{n-1}(g \cdot \vec{\boldsymbol{X}} - g \cdot \mathbb{E}[\vec{\boldsymbol{X}}])(g \cdot \vec{\boldsymbol{X}} - g \cdot \mathbb{E}[\vec{\boldsymbol{X}}])^\top \tag{25}$$

$$= \frac{1}{n-1}(\boldsymbol{Q}(\vec{\boldsymbol{X}} - \mathbb{E}[\vec{\boldsymbol{X}}]))(\boldsymbol{Q}(\vec{\boldsymbol{X}} - \mathbb{E}[\vec{\boldsymbol{X}}]))^\top \tag{26}$$

$$= \frac{1}{n-1}\boldsymbol{Q}(\vec{\boldsymbol{X}} - \mathbb{E}[\vec{\boldsymbol{X}}])(\vec{\boldsymbol{X}} - \mathbb{E}[\vec{\boldsymbol{X}}])^\top\boldsymbol{Q}^\top \tag{27}$$

$$= \boldsymbol{Q}\mathrm{Cov}(\vec{\boldsymbol{X}}, \vec{\boldsymbol{X}})\boldsymbol{Q}^\top. \tag{28}$$

Based on the Cholesky decomposition used in the derivation of F, we denote $\mathrm{Cov}(\vec{\boldsymbol{X}}, \vec{\boldsymbol{X}}) = \vec{\boldsymbol{L}}\vec{\boldsymbol{L}}^\top$ and $\mathrm{Cov}(g \cdot \vec{\boldsymbol{X}}, g \cdot \vec{\boldsymbol{X}}) = \vec{\boldsymbol{L}}_g\vec{\boldsymbol{L}}_g^\top$, with which we can immediately derive:

$$\vec{\boldsymbol{L}}_g\vec{\boldsymbol{L}}_g^\top = \boldsymbol{Q}\vec{\boldsymbol{L}}\vec{\boldsymbol{L}}^\top\boldsymbol{Q}^\top. \tag{29}$$

Considering $\boldsymbol{Q}^{-1} = \boldsymbol{Q}^\top$, we further have the following equation:

$$\vec{\boldsymbol{L}}_g^{-\top}\vec{\boldsymbol{L}}_g^{-1} = \boldsymbol{Q}\vec{\boldsymbol{L}}^{-\top}\vec{\boldsymbol{L}}^{-1}\boldsymbol{Q}^\top. \tag{30}$$

Given that $F(\vec{\boldsymbol{x}}) = \vec{\boldsymbol{L}}^{-1}(\vec{\boldsymbol{x}} - \mathbb{E}[\vec{\boldsymbol{X}}])$, we are now ready to prove Lemma C.1 as follows:

$$\|F_g(g \cdot \vec{\boldsymbol{x}}_i) - F_g(g \cdot \vec{\boldsymbol{x}}_j)\| = \|\vec{\boldsymbol{L}}_g^{-1}(g \cdot \vec{\boldsymbol{x}}_i - \mathbb{E}[g \cdot \vec{\boldsymbol{X}}]) - \vec{\boldsymbol{L}}_g^{-1}(g \cdot \vec{\boldsymbol{x}}_j - \mathbb{E}[g \cdot \vec{\boldsymbol{X}}])\| \tag{31}$$

$$= \|\vec{\boldsymbol{L}}_g^{-1}(g \cdot \vec{\boldsymbol{x}}_i - g \cdot \vec{\boldsymbol{x}}_j)\| = \|\vec{\boldsymbol{L}}_g^{-1}\boldsymbol{Q}(\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j)\| \tag{32}$$

$$= \sqrt{(\vec{\boldsymbol{L}}_g^{-1}\boldsymbol{Q}(\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j))^\top(\vec{\boldsymbol{L}}_g^{-1}\boldsymbol{Q}(\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j))} \tag{33}$$

$$= \sqrt{(\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j)^\top\boldsymbol{Q}^\top\vec{\boldsymbol{L}}_g^{-\top}\vec{\boldsymbol{L}}_g^{-1}\boldsymbol{Q}(\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j)} \tag{34}$$

$$= \sqrt{(\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j)^\top\vec{\boldsymbol{L}}^{-\top}\vec{\boldsymbol{L}}^{-1}(\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j)} \tag{35}$$

$$= \sqrt{(\vec{\boldsymbol{L}}^{-1}(\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j))^\top(\vec{\boldsymbol{L}}^{-1}(\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j))} = \|\vec{\boldsymbol{L}}^{-1}(\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j)\| \tag{36}$$

$$= \|\vec{\boldsymbol{L}}^{-1}(\vec{\boldsymbol{x}}_i - \mathbb{E}[\vec{\boldsymbol{X}}]) - \vec{\boldsymbol{L}}^{-1}(\vec{\boldsymbol{x}}_j - \mathbb{E}[\vec{\boldsymbol{X}}])\| = \|F(\vec{\boldsymbol{x}}_i) - F(\vec{\boldsymbol{x}}_j)\| \tag{37}$$

□

With Lemma C.1, we are able to give the proof of Proposition 3.1 as follows.

Proof. A first observation is that to prove Proposition 3.1, we only need to prove the equivariance in the 1-layer case, since the multi-layer case can be decomposed into the 1-layer case by inserting $I = F \circ F^{-1}$ between layers, where I is the identical mapping. Generally, each layer in scalarization-based E(3)-equivariant GNN has the following paradigm:

$$\boldsymbol{m}_{ij} = \phi_m(\boldsymbol{h}_i, \boldsymbol{h}_j, \|\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j\|^2, \boldsymbol{e}_{ij}), \tag{38}$$

$$\vec{\boldsymbol{x}}_i' = \vec{\boldsymbol{x}}_i + \sum\nolimits_{j\in\mathcal{N}(i)} (\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j)\phi_x(\boldsymbol{m}_{ij}), \tag{39}$$

$$\vec{\boldsymbol{h}}_i' = \phi_h(\boldsymbol{h}_i, \sum\nolimits_{j\in\mathcal{N}(i)} \boldsymbol{m}_{ij}), \tag{40}$$

where $\mathcal{N}(i)$ denotes the neighborhood of node i, and ϕ_x outputs a scalar. Therefore, for the invariant part, we have:

$$\begin{aligned}
f_i(\{\boldsymbol{h}_i, F_g(g\cdot\vec{\boldsymbol{x}}_i)\}) &= \phi_h(\boldsymbol{h}_i, \sum\nolimits_{j\in\mathcal{N}(i)} \boldsymbol{m}_{ij,F_g}) && (41)\\
&= \phi_h(\boldsymbol{h}_i, \sum\nolimits_{j\in\mathcal{N}(i)} \phi_m(\boldsymbol{h}_i, \boldsymbol{h}_j, \|F_g(g\cdot\vec{\boldsymbol{x}}_i) - F_g(g\cdot\vec{\boldsymbol{x}}_j)\|^2, \boldsymbol{e}_{ij})) && (42)\\
&= \phi_h(\boldsymbol{h}_i, \sum\nolimits_{j\in\mathcal{N}(i)} \phi_m(\boldsymbol{h}_i, \boldsymbol{h}_j, \|F(\vec{\boldsymbol{x}}_i) - F(\vec{\boldsymbol{x}}_j)\|, \boldsymbol{e}_{ij})) && (43)\\
&= \phi_h(\boldsymbol{h}_i, \sum\nolimits_{j\in\mathcal{N}(i)} \boldsymbol{m}_{ij,F}) && (44)\\
&= f_i(\{\boldsymbol{h}_i, F(\vec{\boldsymbol{x}}_i)\}) && (45)
\end{aligned}$$

For the equivariant features, recalling $F^{-1}(\vec{\boldsymbol{x}}) = \vec{\boldsymbol{L}}\vec{\boldsymbol{x}} + \mathbb{E}[\vec{\boldsymbol{X}}]$, we have:

$$\begin{aligned}
&F_g^{-1}\vec{f}_i(\{\boldsymbol{h}_i, F_g(g\cdot\vec{\boldsymbol{x}}_i)\}) && (46)\\
&= F_g^{-1}(F_g(g\cdot\vec{\boldsymbol{x}}_i) + \sum\nolimits_{j\in\mathcal{N}(i)} (F_g(g\cdot\vec{\boldsymbol{x}}_i) - F_g(g\cdot\vec{\boldsymbol{x}}_j))\phi_x(\boldsymbol{m}_{ij,F_g})) && (47)\\
&= F_g^{-1}(F_g(g\cdot\vec{\boldsymbol{x}}_i) + \sum\nolimits_{j\in\mathcal{N}(i)} \vec{\boldsymbol{L}}_g^{-1}\boldsymbol{Q}(\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j)\phi_x(\boldsymbol{m}_{ij,F})) && (48)\\
&= F_g^{-1}(\vec{\boldsymbol{L}}_g^{-1}(\boldsymbol{Q}\vec{\boldsymbol{x}}_i + \vec{\boldsymbol{t}} - \mathbb{E}[g\cdot\vec{\boldsymbol{X}}]) + \sum\nolimits_{j\in\mathcal{N}(i)} \vec{\boldsymbol{L}}_g^{-1}\boldsymbol{Q}(\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j)\phi_x(\boldsymbol{m}_{ij,F})) && (49)\\
&= F_g^{-1}(\vec{\boldsymbol{L}}_g^{-1}(\boldsymbol{Q}\vec{\boldsymbol{x}}_i + \vec{\boldsymbol{t}} - g\cdot\mathbb{E}[\vec{\boldsymbol{X}}]) + \sum\nolimits_{j\in\mathcal{N}(i)} \vec{\boldsymbol{L}}_g^{-1}\boldsymbol{Q}(\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j)\phi_x(\boldsymbol{m}_{ij,F})) && (50)\\
&= F_g^{-1}(\vec{\boldsymbol{L}}_g^{-1}(\boldsymbol{Q}\vec{\boldsymbol{x}}_i - \boldsymbol{Q}\mathbb{E}[\vec{\boldsymbol{X}}]) + \sum\nolimits_{j\in\mathcal{N}(i)} \vec{\boldsymbol{L}}_g^{-1}\boldsymbol{Q}(\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j)\phi_x(\boldsymbol{m}_{ij,F})) && (51)\\
&= F_g^{-1}(\vec{\boldsymbol{L}}_g^{-1}\boldsymbol{Q}(\vec{\boldsymbol{x}}_i - \mathbb{E}[\vec{\boldsymbol{X}}]) + \sum\nolimits_{j\in\mathcal{N}(i)} \vec{\boldsymbol{L}}_g^{-1}\boldsymbol{Q}(\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j)\phi_x(\boldsymbol{m}_{ij,F})) && (52)\\
&= F_g^{-1}(\vec{\boldsymbol{L}}_g^{-1}\boldsymbol{Q}(\vec{\boldsymbol{x}}_i - \mathbb{E}[\vec{\boldsymbol{X}}] + \sum\nolimits_{j\in\mathcal{N}(i)} (\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j)\phi_x(\boldsymbol{m}_{ij,F}))) && (53)\\
&= \boldsymbol{Q}(\vec{\boldsymbol{x}}_i - \mathbb{E}[\vec{\boldsymbol{X}}] + \sum\nolimits_{j\in\mathcal{N}(i)} (\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j)\phi_x(\boldsymbol{m}_{ij,F})) + \mathbb{E}[g\cdot\vec{\boldsymbol{X}}] && (54)\\
&= \boldsymbol{Q}(\vec{\boldsymbol{x}}_i + \sum\nolimits_{j\in\mathcal{N}(i)} (\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j)\phi_x(\boldsymbol{m}_{ij,F})) - \boldsymbol{Q}\mathbb{E}[\vec{\boldsymbol{X}}] + g\cdot\mathbb{E}[\vec{\boldsymbol{X}}] && (55)\\
&= \boldsymbol{Q}(\vec{\boldsymbol{x}}_i + \sum\nolimits_{j\in\mathcal{N}(i)} (\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j)\phi_x(\boldsymbol{m}_{ij,F})) + \vec{\boldsymbol{t}} && (56)\\
&= g\cdot(\vec{\boldsymbol{x}}_i + \sum\nolimits_{j\in\mathcal{N}(i)} (\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j)\phi_x(\boldsymbol{m}_{ij,F})) && (57)
\end{aligned}$$

By replacing g with identical element I in $E(3)$, since $F = F_I$, we can immediately derive:

$$F^{-1}\vec{f}_i((\{\boldsymbol{h}_i, F(\vec{\boldsymbol{x}}_i)\}) = (\vec{\boldsymbol{x}}_i + \sum\nolimits_{j\in\mathcal{N}(i)} (\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j)\phi_x(\boldsymbol{m}_{ij,F})), \tag{58}$$

$$g\cdot F^{-1}\vec{f}_i((\{\boldsymbol{h}_i, F(\vec{\boldsymbol{x}}_i)\}) = F_g^{-1}\vec{f}_i(\{\boldsymbol{h}_i, F_g(g\cdot\vec{\boldsymbol{x}}_i)\}), \tag{59}$$

which concludes Proposition 3.1. □

D Algorithm for Training and Sampling

We present the pseudo codes for training in Algorithm 1 amd sampling in Algorithm 2.

Algorithm 1 Training Algorithm of PepGLAD

```
input geometric data of protein-peptide complexes $\mathcal{S}$
output encoder $\mathcal{E}_\phi$, decoder $\mathcal{D}_\xi$, denoising network $\boldsymbol{\epsilon}_\theta$
function TrainAutoEncoder($\mathcal{S}$)
    Initialize $\mathcal{E}_\phi$, $\mathcal{D}_\xi$
    while $\phi, \xi$ have not converged do
        Sample $(\mathcal{G}_p, \mathcal{G}_b) \sim \mathcal{S}$
        $\tilde{\mathcal{G}}_p \leftarrow \mathrm{mask}(\mathcal{G}_p)$    {Mask 25% Residues}
        $\{(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i, \vec{\boldsymbol{\mu}}_i, \vec{\boldsymbol{\sigma}}_i)\} \leftarrow \mathcal{E}_\phi(\tilde{\mathcal{G}}_p, \mathcal{G}_b)$    {Encoding}
        $\{(\boldsymbol{\epsilon}_i, \vec{\boldsymbol{\epsilon}}_i)\} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$    {Reparameterization}
        $\mathcal{G}_z \leftarrow \{(\boldsymbol{\mu}_i + \boldsymbol{\epsilon}_i \odot \boldsymbol{\sigma}_i, \vec{\boldsymbol{\mu}}_i + \vec{\boldsymbol{\epsilon}}_i \odot \vec{\boldsymbol{\sigma}}_i)\}$
        $\mathcal{G}'_p \leftarrow \mathcal{D}_\xi(\mathcal{G}_z, \mathcal{G}_b)$    {Decoding}
        $\mathcal{L}_{AE} = \sum_{i \in \mathcal{G}_p} (\mathcal{L}_{recon}(i) + \mathcal{L}_{KL}(i)) / |\mathcal{G}_p|$
        $\phi, \xi \leftarrow \mathrm{optimizer}(\mathcal{L}_{AE}; \phi, \xi)$
    end while
    return $\mathcal{E}_\phi$, $\mathcal{D}_\xi$
end function

function TrainLatentDiffusion($\mathcal{E}_\phi, \mathcal{D}_\xi, \mathcal{S}$)
    Initialize $\boldsymbol{\epsilon}_\theta$
    while $\theta$ have not converged do
        Sample $(\mathcal{G}_p, \mathcal{G}_b) \sim \mathcal{S}$
        $\{(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i, \vec{\boldsymbol{\mu}}_i, \vec{\boldsymbol{\sigma}}_i)\} \leftarrow \mathcal{E}_\phi(\mathcal{G}_p, \mathcal{G}_b)$    {Encoding}
        $\mathcal{G}^0_z \leftarrow \{(\boldsymbol{\mu}_i, \vec{\boldsymbol{\mu}}_i)\}$
        $F \leftarrow \mathrm{affine}(\mathcal{G}_b)$    {Affine Transformation}
        $\mathcal{G}^0_z, \mathcal{G}_b \leftarrow F(\mathcal{G}^0_z), F(\mathcal{G}_b)$    {Standard Geometry}
        $\{(\boldsymbol{z}^0_i, \vec{\boldsymbol{z}}^0_i)\} \leftarrow \mathcal{G}^0_z$
        $t \sim \mathbf{U}(1, T)$, $\{(\boldsymbol{\epsilon}_i, \vec{\boldsymbol{\epsilon}}_i)\} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
        $\mathcal{G}^t_z \leftarrow \{\sqrt{\bar{\alpha}^t}[\boldsymbol{z}^0_i, \vec{\boldsymbol{z}}^0_i] + (1 - \bar{\alpha}^t)[\boldsymbol{\epsilon}_i, \vec{\boldsymbol{\epsilon}}_i]\}$
        $\mathcal{L}^t_{LDM} = \sum_i \|[\boldsymbol{\epsilon}_i, \vec{\boldsymbol{\epsilon}}_i] - \boldsymbol{\epsilon}_\theta(\mathcal{G}^t_z, \mathcal{G}_1, t)[i]\|^2 / |\mathcal{G}^t_z|$
        $\theta \leftarrow \mathrm{optimizer}(\mathcal{L}^t_{LDM}; \theta)$
    end while
    return $\boldsymbol{\epsilon}_\theta$
end function

$\mathcal{E}_\phi, \mathcal{D}_\xi \leftarrow$ TrainAutoEncoder($\mathcal{S}$)
Fix parameters $\phi$ and $\xi$
$\boldsymbol{\epsilon}_\theta \leftarrow$ TrainLatentDiffusion($\mathcal{E}_\phi, \mathcal{D}_\xi, \mathcal{S}$)
return $\mathcal{E}_\phi$, $\mathcal{D}_\xi$, $\boldsymbol{\epsilon}_\theta$
```

E Data Preparation

We show details for constructing the datasets used in our paper here. Further statistics are presented in Table 5 and distribution of peptide lengths in Figure 5.

E.1 Unsupervised Data from Protein Fragments (ProtFrag)

We exploit unsupervised data from monomer proteins to enrich the training of the autoencoder. Specifically, we first extract all single chains from the Protein Data Bank (PDB) before December 8th, 2023, and remove the duplicated chains on a sequence identity threshold of over 90%. Then, for each chain, we extract fragments satisfying the following criteria:

Algorithm 2 Sampling Algorithm of PepGLAD

```
input  decoder $\mathcal{D}_\xi$, denoising network $\boldsymbol{\epsilon}_\theta$, binding site $\mathcal{G}_b$
output  peptide $\mathcal{G}_p$
$F \leftarrow \mathrm{affine}(\mathcal{G}_b)$                                   {Affine Transformation}
$\mathcal{G}_b \leftarrow F(\mathcal{G}_b)$                                      {Standard Geometry}
$\{(\boldsymbol{z}_i^T, \vec{\boldsymbol{z}}_i^T)\} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
for $t$ in $T, T-1, \cdots, 1$ do
    $\mathcal{G}_z^t \leftarrow \{(\boldsymbol{z}_i^t, \vec{\boldsymbol{z}}_i^t)\}$                         {Latent Denoising Loop}
    $\vec{\boldsymbol{u}}_i^t \leftarrow \boldsymbol{z}_i^t, \vec{\boldsymbol{z}}_i^t$
    $\boldsymbol{\varepsilon}_i = [\boldsymbol{\epsilon}_i, \vec{\boldsymbol{\epsilon}}_i] \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
    $\vec{\boldsymbol{u}}_i^{t-1} \leftarrow \frac{1}{\sqrt{\alpha^t}}(\vec{\boldsymbol{u}}_i^t - \frac{\beta^t}{\sqrt{1-\bar{\alpha}^t}}\boldsymbol{\epsilon}_\theta(\mathcal{G}_z^t, \mathcal{G}_b, t)[i]) + \beta_t \boldsymbol{\varepsilon}_i$
    $[\boldsymbol{z}_i^{t-1}, \vec{\boldsymbol{z}}_i^{t-1}] \leftarrow \vec{\boldsymbol{u}}_i^{t-1}$
end for
$\mathcal{G}_z \leftarrow \{(\boldsymbol{z}_i^0, \vec{\boldsymbol{z}}_i^0)\}$
$\mathcal{G}_z, \mathcal{G}_b \leftarrow F^{-1}(\mathcal{G}_z), F^{-1}(\mathcal{G}_b)$                    {Data Geometry}
$\mathcal{G}_p \leftarrow \mathcal{D}_\xi(\mathcal{G}_z, \mathcal{G}_b)$                                  {Decoding}
return $\mathcal{G}_p$
```

1. **Length**: The fragment should consist of 4 to 25 residues.
2. **Balanced constitution**: No single amino acid should constitute more than 25% of the fragment; Hydrophobic amino acids should comprise less than 45% of the fragment, with charged amino acids accounting for 25% to 45%.
3. **Isolated Stability**: Instability [22] should be below 40; Considering the surrounding amino acids as interaction partners, the fragment should have a buried surface area (BSA) above 400Å^2 [10], with a relative BSA above 20%.

We use FreeSASA [47] to calculate the surface area of fragments. Let SA$_{bound}$ represent the surface area of the isolated fragment when considering surrounding amino acids, and SA$_{unbound}$ represent the surface area when not considering them. The buried surface area is then calculated as BSA $=$ SA$_{unbound}$ $-$ SA$_{bound}$, and the relative BSA is calculated as BSA$_{rel}$ $=$ BSA$/$SA$_{unbound}$. In total, we obtain 70,645 fragments meeting these criteria.

E.2 Construction of Our PepBench

Here we illustrate the details for constructing the supervised dataset (PepBench) used in our paper. Similar to literature [65], we also exploits available data from the Protein Data Bank (PDB) [5]. We first extract all dimers in PDB deposited before December 8th, 2023, and filter out the complexes with a receptor longer than 30 residues and a ligand between 4 to 25 residues, which aligns with Tsaban et al. [59]. Peptides with lengths in this range are more relevant to practical applications such as drug discovery, as they exhibit favorable biochemical properties [49]. Then we remove the duplicated complexes with the criterion that both the receptor and the peptide has a sequence identity over 90% [56], after which 6105 non-redundant complexes are obtained. We use MMseqs2 for clustering based on sequence identity:

```
# create database from the sequences
mmseqs create seqs.fasta database
# clustering with sequence identity above 90%
mmseqs cluster database database_clusters results --min-seq-id 0.9 -c 0.95 --cov-mode 1
```

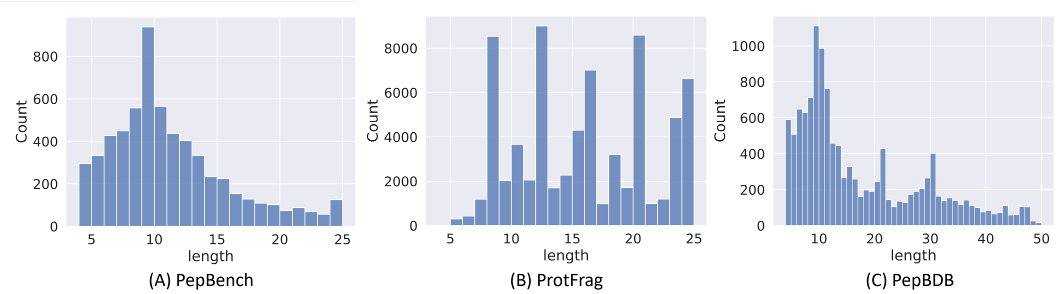
To achieve the cross-target generalization test, we utilize the large non-redundant dataset (LNR) introduced by Tsaban et al. [59] as the test set, which is curated by domain experts. LNR originally includes 96 protein-peptide complexes. We obtain 93 complexes after excluding the ones with non-canonical amino acids. We then cluster the LNR along with the PDB data by receptor with a sequence identity threshold of over 40%. Subsequently, we remove the complexes sharing the same clusters with those from the test set and those including non-canonical amino acids in the peptides. Finally, the remaining data are randomly split based on clustering results into training and validation sets. The characteristics of different splits are presented in Table 5. In addition, the binding site contains residues on the receptor within 10Å distances to the peptide, where the distance between two residues is measured by the distance between their $\mathtt{C}_\beta$ coordinates.

E.3 Split of PepBDB

Similar to the split of PepBench, we use MMseqs2 for clustering and randomly split the data into training, validation, and test sets based on the clustering results. For the test set, we randomly select one protein-peptide complex in each cluster to avoid redundancy.

Table 5: Statistics of the constructed datasets.

Split	#entry	#cluster	source
PepBench (Training)	4,157	952	PDB [5]
PepBench (Validation)	114	50	PDB [5]
PepBench (Test)	93	93	LNR [59]
PepBDB (Training)	8,434	1,617	PepBDB [65]
PepBDB (Validation)	370	95	PepBDB [65]
PepBDB (Test)	190	190	PepBDB [65]
ProtFrag	70,645	-	monomers in PDB



(A) PepBench

(B) ProtFrag

(C) PepBDB

Figure 5: Peptide (or protein fragment) length distribution of three datasets.

F E(3)-Invariant Latent Space

We found that the E(3)-equivariant latent vectors were significant, which can convey sufficient geometric information. If we use E(3)-invariant latent space without these geometric latent vectors, the reconstruction ability (RMSD, DockQ) of the VAE deteriorates significantly (Table 6). It is reasonable since E(3)-invariant latent space lacks geometric interactions with the pocket atoms, leading to difficulties in reconstructing the full-atom structures on the binding site.

Table 6: Comparison of reconstruction ability of variational autoencoders with E(3)-equivariant latent space and E(3)-invariant latent space.

Latent Space	AAR↑	RMSD ↓	DockQ↑
E(3)-equivariant	95.1%	0.79	0.898
E(3)-invariant	93.4%	1.75	0.823

G Guidance on Sequence Orders for Sampling

Peptides consist of linearly connected amino acids, which exert constraints on the 3D geometry. Specifically, residues adjacent in the sequence should also be close in the structure, since they are connected by a peptide bond. However, 3D graphs are unordered and do not incorporate such induct bias, which means the generated nodes might have arbitrary permutation on sequence orders. To tackle this problem, we take inspiration from classifier-guided diffusion [15], which adds the gradient of a classifier to the denoising outputs to guide the generative diffusion process towards the subspace where the classifier gives high confidence. We utilize an empirical classifier $p(1|\{\vec{z}_i^t\})$ defined as follows:

$$p(1|\{\vec{z}_i^t\}) = \exp(-\sum\nolimits_{\mathcal{P}(i)-\mathcal{P}(j)=1} E(\|\vec{z}_i^t - \vec{z}_j^t\|)), \tag{60}$$

$$E(d) = \begin{cases} d - (\mu_d + 3\sigma_d), & d > \mu_d + 3\sigma_d, \\ (\mu_d - 3\sigma_d) - d, & d < \mu_d - 3\sigma_d, \\ 0, & \text{otherwise}, \end{cases} \tag{61}$$

where μ_d and σ_d are the mean and variance of the distances of adjacent residues in the latent space measured from the training set. With $p(1|\{\vec{z}_i^t\})$, we are able to assign an arbitrary permutation on sequence orders to the nodes, and steer the sampling procedure towards the desired subspace conforming to $\mathcal{P}$, since this classifier gives higher confidence if the adjacent (defined by $\mathcal{P}$) residues are within reasonable distances aligning with the statistics from the training set. In particular, the coordinate denoising outputs are refined as follows:

$$\vec{\epsilon}_i^t = \vec{\epsilon}_\theta(\mathcal{G}_z^t, \mathcal{G}_b, t)[i] - \lambda\sqrt{1-\bar{\alpha}^t}\nabla_{\vec{z}_i^t} \log p(1|\{\vec{z}_i^t\}), \tag{62}$$

where λ adjusts the weight of the guidance. Besides the constraints on the distance between adjacent residues, we can also include guidance on avoiding clashes between non-adjacent residues by defining the following energy term:

$$C(d) = \begin{cases} \mu_d - d, & d < \mu_d, \\ 0, & \text{otherwise}, \end{cases} \tag{63}$$

Subsequently, we just need to revise the empirical classifier as:

$$p(1|\{\vec{z}_i^t\}) = \exp(-\sum_{\mathcal{P}(i)-\mathcal{P}(j)=1} E(\|\vec{z}_i^t - \vec{z}_j^t\|) - \sum_{\mathcal{P}(i)-\mathcal{P}(j)\neq 1} C(\|\vec{z}_i^t - \vec{z}_j^t\|) - \sum_{i\in\mathcal{G}_z, j\in\mathcal{G}_b} C(\|\vec{z}_i^t - \vec{r}_j\|), \tag{64}$$

where $\vec{r}_j$ is the $\mathtt{C}_\alpha$ coordinate of node j in the binding site. We observe a slight improvement upon including the clash energy term. Nevertheless, the guidance is only a small technical trick with minor enhancement on the performance as shown in Table 7.

Table 7: Results on binding conformation generation with and without guidance on sequence orders.

Guidance	RMSD$_{C_\alpha}$ ↓	RMSD$_{\text{atom}}$ ↓	DockQ↑
w/	4.09	5.30	0.592
w/o	4.10	5.34	0.582

H Metrics for Evaluating Sequence-Structure Co-Design

H.1 Why not Amino Acid Recovery (AAR)

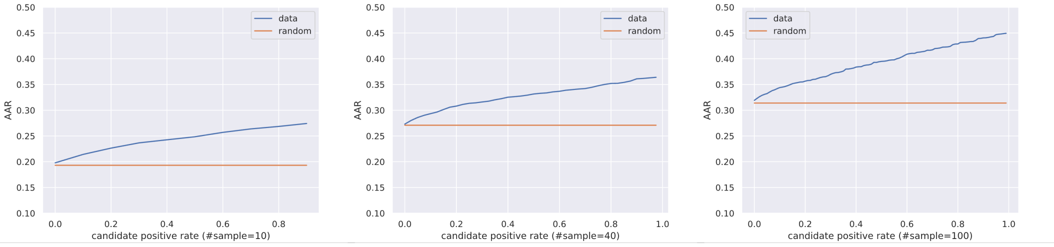


Figure 6: Best amino acid recovery (AAR) on samples constructed according to specified positive ratios. Each point contains the averaged result across 102 receptors.

While the amino acid recovery (AAR) is widely used in antibody design [33, 34, 44], we find it not informative enough for evaluating generative models on learning the distribution of binding peptides, due to the vast and highly diverse solution space. To elucidate this, we conducted an analysis of AAR using a sequence dataset from Xia et al. [67] including 328 receptors and 600k peptides with binary binding labels, from which we filter out 102 receptors with well-explored solution space (*i.e.* with at least 500 binders). For each receptor, we randomly select one binder as the reference and sample N candidates according to a specified positive ratio r, resembling the scenario of evaluating generative models for peptide design. For example, setting $r = 0.1$ and $N = 100$ involves sampling 10 binders and 90 non-binders to construct the candidates. Then we compute the best AAR of these candidates with respect to the reference as the result. This process is repeated for 10 times per receptor, and the results are averaged across different receptors, which can be interpreted as the evaluation score of AAR on a generative model that can generates candidates with a positive ratio of r. We select $N = 10, 40, 100$ and enumerate the choices of r to derive the relation plot in Figure 6, where the results of a random sequence generator is also included for comparison. It can be derived that the gap of AAR between the worst model ($r = 0.0$) and the best model ($r = 1.0$) is insignificant. Furthermore, all models, including the best model ($r = 1.0$) which always produces positive samples, exhibit performance akin to the random sequence generator, with consistent trends regardless of the choice of N. We attribute this to the vast solution space of peptide design, where evaluating with dozens of candidates relative to a single reference is unreliable. In other words, **achieving a high AAR is improbable since the model would need to fortuitously explore the subspace around the reference, which is arbitrary, on every test receptor; Conversely, a low AAR does not necessarily denote a poor generative model, as the model may be exploring distinct solutions from the single reference.** Moreover, we calculate the Spearman correlation between the receptor-level best AAR and the positive ratio r of the candidates, yielding 0.23, 0.22 and 0.24 for $N = 10$, 40, and 100, respectively, indicating very weak correlation. Based on this analysis, we assert that AAR is unsuitable for evaluating target-specific peptide design. The comparison of AAR and success rates on PepBench (Table 8) also indicates that models with similar AAR can have distinct success rates.

Table 8: Amino acid recovery and success rates of different models on PepBench.

Model	AAR	Success ($\Delta G < 0$)
HSRN	35.8%	10.46%
DiffAb	37.1%	49.87%
PepGLAD	36.7%	55.97%

H.2 "Unsupervised" Consistency

While it is common to evaluate consistency by comparing generated structures with AF2-predicted structures [7], such a "supervision-based" method suffers from severe limitations in the situation where AF2 [31] fails to achieve an acceptable performance. As highlighted not only in Tsaban et al. [59] but also demonstrated by our experiments, even state-of-the-art models like AF2 struggles to consistently produce high-quality structures of protein-peptide complexes. Our findings reveal that only 36% of test samples could be accurately predicted within a 5Å RMSD (considered as near-native conformation) by AF2, indicating even the ground truth can only achieve 36% success rates if we use such supervison-based consistency for evaluation, making such evaluation not reliable in the domain of peptide design.

In light of this limitation, we propose an "unspervised" evaluation framework to assess consistency, that is, the statistical association between the clustering results of sequences and structures. Theoretically, this serves the necessary condition for true consistency. Namely, if a model truly captures the consistency between sequence and structure, it will necessarily achieve a high score on the proposed metric. Conversely, if a model fails to attain a high score on the proposed metric, it is not possible to capture true consistency.

In our experiments, we found that our proposed consistency metric effectively distinguishes non-consistent modeling methods, such as HSRN, which tend to produce disparate sequences while sharing identical structures.

H.3 Implememtation and Elaboration on Metrics

Indeed, evaluating generative models comprehensively is crucial, requiring assessment from multiple perspectives. Generally, these evaluations can be categorized into two main aspects: diversity and fidelity to the desired distribution. Regarding diversity, we take inspiration from Yim et al. [72] and quantify it with the number of unique clusters relative to the number of candidates. This metric provides insight into the variety and richness of the generated samples. For fidelity, the primary focus should be the binding affinity, for which we adopt the physical energy from Rosetta [2] since it is widely used in various domains and exhibit robust generalization capability [1, 37, 64]. Further, considering the dependency between sequence and structure, we propose the consistency metric, which is critical for distinguishing whether the generative model is capturing the 1D&3D joint distribution and thereby truly facilitating "co-design". We have also discussed the reason for implementing the consistency metric with the "unsupervised" fashion in the above section. Below, we outline the implementation of these metrics.

Diversity We hierarchically cluster the sequences and the structures with aligning score [36] and RMSD of $\mathtt{C}_\alpha$, respectively. In particular, we implement the aligning score using Biopython [11] with BLOSUM62 matrix [24] and Needleman-Wunsch algorithm [50]. The thresholds for clustering is similarity above 0.6 and RMSD below 4.0 for sequence and structure, respectively. We provide the python codes below for clearer presentation:

```python
from math import sqrt
from typing import List
from Bio.Align import substitution_matrices, PairwiseAligner

import numpy as np


def align_score(sequence_A, sequence_B):
    # load matrix
    sub_matrice = substitution_matrices.load('BLOSUM62')
    aligner = PairwiseAligner()
    aligner.substitution_matrix = sub_matrice
    # align
    alns = aligner.align(sequence_A, sequence_B)
    best_aln = alns[0]
    aligned_A, aligned_B = best_aln
    # normalize
    base_A = aligner.score(sequence_A, sequence_A)
    base_B = aligner.score(sequence_B, sequence_B)
    base = sqrt(base_A * base_B)
    similarity = aligner.score(sequence_A, sequence_B) / base
    return similarity


def seq_diversity(seqs: List[str], th: float=0.4) -> float:
    '''
        th: sequence distance (1 - similarity)
    '''
    dists = []
    for i, sequence_A in enumerate(seqs):
        dists.append([])
        for j, sequence_B in enumerate(seqs):
            dists[i].append(1 - align_score(sequence_A, sequence_B))
    dists = np.array(dists)
    Z = linkage(squareform(dists), 'single')
    cluster = fcluster(Z, t=th, criterion='distance')
    return len(np.unique(cluster)) / len(seqs)


def struct_diversity(structs: np.ndarray, th: float=4.0) -> float:
    '''
        structs: N*L*3, alpha carbon coordinates
        th: threshold for clustering (distance < th)
    '''
    ca_dists = np.sum((structs[:, None] - structs[None, :]) ** 2, axis=-1)
    rmsd = np.sqrt(np.mean(ca_dists, axis=-1))
    Z = linkage(squareform(rmsd), 'single')
    cluster = fcluster(Z, t=th, criterion='distance')
    return len(np.unique(cluster)) / structs.shape[0]
```

Denoting the diversity of the sequences and the structures as Div_{seq} and Div_{struct}, respectively, we calculate the co-design diversity as $\sqrt{\text{Div}_{seq}\text{Div}_{struct}}$.

Consistency The clustering process in the calculation of diversity will assign each sequence and each structure with a clustering label. The labels on the sequences and those on the structures can be regarded as two nominal variables. Since similar sequences should produce similar structures, these two variables should be highly correlated if the model really learns the joint distribution. Naturally, we quantify the consistency via Cramér’s V [12] association between these two variables, with 1.0 indicating perfect association, and 0.0 means no association.

ΔG We use Rosetta [2] to calculate the binding energy with the "ref2015" score function. Both the candidates and the references first endure the fast relax protocol in Rosetta to correct atomic clashes at the interface before the computation of binding energy. We use the implementation in pyRosetta (*i.e.* the python version of Rosetta), which is borrowed from Luo et al. [44]:

```
import pyrosetta
from pyrosetta.rosetta import protocols
from pyrosetta.rosetta.protocols.relax import FastRelax
from pyrosetta.rosetta.core.pack.task import TaskFactory
from pyrosetta.rosetta.core.pack.task import operation
from pyrosetta.rosetta.core.select import residue_selector as selections
from pyrosetta.rosetta.core.select.movemap import MoveMapFactory, move_map_action
from pyrosetta.rosetta.core.scoring import ScoreType
from pyrosetta.rosetta.protocols.analysis import InterfaceAnalyzerMover

pyrosetta.init(’ ’.join([
    ’-mute’, ’all’,
    ’-use_input_sc’,
    ’-ignore_unrecognized_res’,
    ’-ignore_zero_occupancy’, ’false’,
    ’-load_PDB_components’, ’false’,
    ’-relax:default_repeats’, ’2’,
    ’-no_fconfig’,
    ’-use_terminal_residues’, ’true’,
    ’-in:file:silent_struct_type’, ’binary’
]))


class RelaxRegion(object):  # Fast Relax

    def __init__(self, max_iter=1000):
        super().__init__()
        self.scorefxn = pyrosetta.create_score_function(’ref2015’)
        self.fast_relax = FastRelax()
        self.fast_relax.set_scorefxn(self.scorefxn)
        self.fast_relax.max_iter(max_iter)

    def __call__(self, pdb_path, peptide_chain):
        pose = pyrosetta.pose_from_pdb(pdb_path)

        tf = TaskFactory()
        tf.push_back(operation.InitializeFromCommandline())
        tf.push_back(operation.RestrictToRepacking())   # Only allow residues to repack.
    No design at any position.

        gen_selector = selections.ChainSelector(chain)
        nbr_selector = selections.NeighborhoodResidueSelector()
        nbr_selector.set_focus_selector(gen_selector)
        nbr_selector.set_include_focus_in_subset(True)
        subset_selector = nbr_selector

        prevent_repacking_rlt = operation.PreventRepackingRLT()
        prevent_subset_repacking = operation.OperateOnResidueSubset(
            prevent_repacking_rlt,
            subset_selector,
            flip_subset=True,
        )
        tf.push_back(prevent_subset_repacking)

        fr = self.fast_relax
        pose = original_pose.clone()

        mmf = MoveMapFactory()
        mmf.add_bb_action(move_map_action.mm_enable, gen_selector)
        mmf.add_chi_action(move_map_action.mm_enable, subset_selector)
        mm  = mmf.create_movemap_from_pose(pose)

        fr.set_movemap(mm)
        fr.set_task_factory(tf)
        fr.apply(pose)
```

```python
        return pose


def pyrosetta_interface_energy(pdb_path, receptor_chains, ligand_chains): # binding energy
    pose = pyrosetta.pose_from_pdb(pdb_path)
    interface = ''.join(ligand_chains) + '_' + ''.join(receptor_chains)
    mover = InterfaceAnalyzerMover(interface)
    mover.set_pack_separated(True)
    mover.apply(pose)
    return pose.scores['dG_separated']
```

Success Since a negative ΔG value typically indicates a potential for binding, we report the ratio of all generated candidates that satisfy this threshold. Moreover, candidates with $\Delta G < 0$ usually are at least physically valid (*i.e.* without obvious atomic clash), thus it is meaningful to use this success rate to evaluation the generative ability of the models.

I Experiment Details

I.1 Implementation of PepGLAD

We train PepGLAD on a GPU with 24G memory with AdamW optimizer. For the autoencoder, we train for 60 epochs with dynamic batches, ensuring that the total number of edges (proportional to the square of the number of nodes) remains below 60,000. The initial learning rate is 10^{-4} and decays by 0.8 if the loss on the validation set has not decreased for 3 consecutive epochs. Regarding the diffusion model, we train for 500 epochs with the same batching strategy as for the autoencoder. The learning rate is 10^{-4} and decay by 0.6 if the loss has not decreased for 3 consecutive validations, where the validation is conducted every 10 epochs. In the experiment of binding conformation generation, we only use the supervised dataset, thus we extend the training epochs for the autoencoder and the diffusion model to 500 and 1000, respectively. Consequently, the patience of learning rate decay is extended to 15 epochs for training the autoencoder. We keep other settings unchanged. The hyperparameters of PepGLAD used in our experiments are provided in Table 9.

Table 9: Hyperparameters of PepGLAD in sequence-structure codesign and binding conformation generation.

Name	Value		Description
	codesign	conformation	
		Variational AutoEncoder	
embed size	128	128	Size of embeddings for residue types.
hidden size	128	128	Size of hidden layers.
h	8	-	Size of the latent variable for residue types in the sequences.
layers	3	3	Number of layers.
λ_1	0.1	0.0	The weight of KL divergence on the sequence.
λ_2	0.5	0.5	The weight of KL divergence on the structure.
λ_{CA}	1.0	1.0	The weight of $\mathtt{C}_\alpha$ loss in $\mathcal{L}_{aux}$.
λ_{bond}	1.0	1.0	The weight of bond loss in $\mathcal{L}_{aux}$.
λ_{angle}	0.5	0.5	The weight of side-chain dihedral angle loss in $\mathcal{L}_{aux}$.
		Latent Diffusion Model	
hidden size	128	128	Size of hidden layers in the denoising network.
layers	3	3	Number of layers.
steps	100	100	Number of diffusion steps.

I.2 Implementation of the Baselines

For **HSRN** [30], **dyMEAN** [34], and **DiffAb** [44], we directly integrate their official implementation into the same training framework as our PepGLAD, and adjust the batch size, learning rate, and training epochs to obtain the optimal performance, which we present in Table 10.

We outline the implementation of other baselines below:

Table 10: Hyperparamenters for training the baselines.

Model	Batch Size	Learning Rate	Epoch
HSRN	8	1.0e-4	50
dyMEAN	32	1.0e-4	100
DiffAb	32	1.0e-4	50

RFDiffusion [64] We follow the official instruction to randomly select 20% of residues on the binding site as "hotspots", and generate the backbone via diffusion followed by cycles of inverse folding with ProteinMPNN [14] and full-atom structural refining with the officially provided Rosetta protocol.

AnchorExtension [26] It is implemented with the Rosetta suite, thus we resort to the official release of the pipeline protocols for docking and optimizing. We use the default parameters and generate 10 candidates for each receptor due to its limitation of efficiency. We randomly pick one peptide in the training set with the same length as the reference peptide as the initial motif for docking.

FlexPepDock [42] We follow its official tutorial to implement this baseline in the C++ version of Rosetta.

AlphaFold2 [31] We borrow the results from [59], which explores two strategy to use AlphaFold2 on peptide conformation generation, including modeling the receptor and the peptide as separate chains or link them together with long loops. The results contain a total of 10 candidates for each receptor, with 5 from the separate strategy and 5 from the linked strategy.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: § 3 and § 4 illustrate the claims well from the methodology and the empirical aspects.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have an independent section (§ 6) to discuss limitations.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: See Appendix B.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: In addition to the methodology section (§ 3), we provide implementation details in Appendix I.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide codes and data for our model and the experiments.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: See § 4.1, Appendix E and Appendix I.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: The size of the dataset is large, and we report results on two different datasets for more solid evaluation.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Appendix I.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We ensure the research to conform to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See § 7.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper does not pose such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: They are properly credited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The README.md in the provided codes illustrate how to construct the benchmark from publicly available resources.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve such a topic.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.