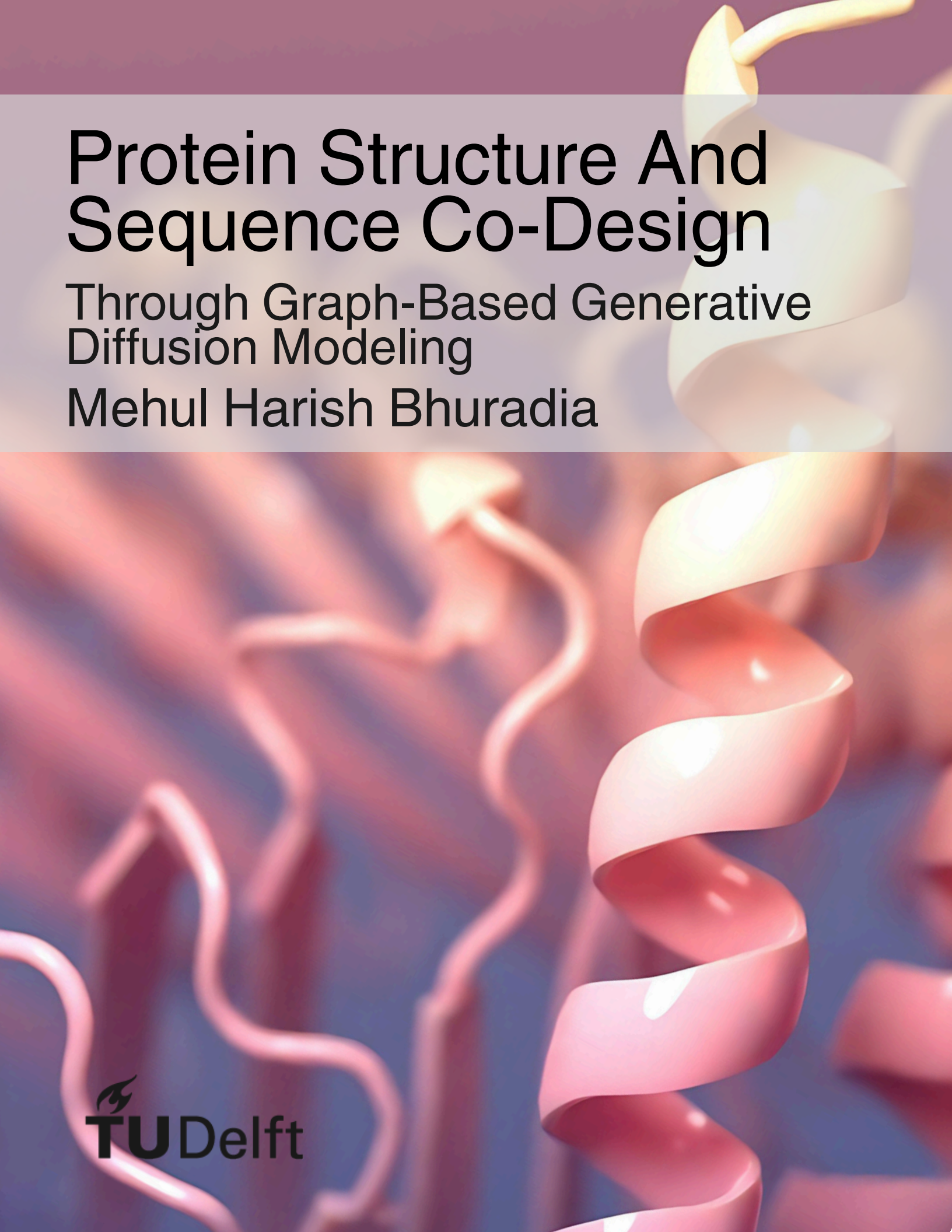
Protein Structure And Sequence Co-Design

Through Graph-Based Generative Diffusion Modeling

Mehul Harish Bhuradia



TU Delft

Protein Structure And Sequence Co-Design Through Graph-Based Generative Diffusion Modeling

by

Mehul Harish Bhuradia

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday, June 24, 2024, at 1:00 PM.

Student Number:	4848969		
Degree:	MSc. Computer Science, Artificial Intelligence Track		
Faculty:	Electrical Engineering, Mathematics and Computer Science		
Project Duration:	October 2023 – June 2024		
Thesis Committee:	Dr. Amelia Villegas-Morcillo,	TU Delft,	Daily Co-Supervisor
	Dr. Hadi Jamali-Rad,	TU Delft,	Daily Supervisor
	Dr. Jana Weber,	TU Delft,	Daily Supervisor
	Prof. dr. Marcel Reinders,	TU Delft,	Thesis Advisor
	Dr. Wendelin Böhmer,	TU Delft	

An electronic version of this thesis is available at `https://repository.tudelft.nl/`

Protein Structure And Sequence Co-Design Through Graph-Based Generative Diffusion Modeling

Mehul Harish Bhuradia

Delft University of Technology

Abstract :

Proteins are fundamental biological macromolecules essential for cellular structure, enzymatic catalysis, and immune defense, making the generation of novel proteins crucial for advancements in medicine, biotechnology, and material sciences. This study explores protein design using deep generative models, specifically Denoising Diffusion Probabilistic Models (DDPMs). While traditional methods often focus on either protein structure or sequence design independently, recent trends emphasize a co-design approach addressing both aspects simultaneously. We propose a novel methodology utilizing Equivariant Graph Neural Networks (EGNNs) within the diffusion framework to co-design protein structures and sequences. We modify the EGNN architecture to improve its effectiveness in learning intricate data patterns. Experimental results show that our approach effectively generates high-quality protein sequences, although challenges remain in producing plausible protein backbones and ensuring strong sequence-structure correlation.

Index Terms: Protein co-design, Diffusion, EGNNs, Generative AI

1 Introduction

Proteins are the building blocks of life; they perform many essential roles such as driving chemical reactions, fighting infections, and regulating hormone levels. While many proteins exist in nature, generating novel proteins with desired properties is crucial for advancing medicine, biotechnology, and material sciences. Novel proteins can be used to combat emerging diseases, create efficient industrial enzymes, and drive innovation across various fields [1]. The creation of novel proteins can be achieved through both experimental and/or computational methods.

Over the years, many methods have been proposed for designing proteins computationally. Leading approaches rely on heuristics and structure optimization through energy minimization. However, these methods face significant challenges and are limited by the vast search space of protein sequences and structures [2]. Even with substantial resource and time investment, traditional methods often yield proteins that are not plausible. Additionally, the diversity of the generated samples is typically constrained by the provided data, limiting their utility. However, with the recent rise of deep learning in various fields, its ability to efficiently handle vast search spaces and generalize beyond provided data, the field of protein design has embraced it. Deep generative models, such as Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), autoregressive models, and Denoising Diffusion Probabilistic Models (DDPMs) [3] — hereafter referred to as diffusion models — are now being explored for protein generation [2, 4–22]. This project explores computational protein design, employing generative diffusion models.

Various approaches leveraging deep generative modeling have been investigated to design novel proteins. Traditionally, methods have predominantly concentrated on designing either the protein backbone structure [2, 4–10] or the protein sequence [11–16] independently. The protein backbone structure comprises only the coordinates of a few selected atoms per amino acid, without including the amino acid type. Meanwhile, the protein sequence specifies the linear order of amino acids in the protein. However, recent advancements aim to address both aspects simultaneously through a process known as protein co-design [17–22].

Combining methods for designing either the protein structure or sequence with techniques like inverse folding [23] and protein folding [24, 25] facilitates the generation of the entire protein structure and sequence, achieving outcomes akin to a co-design model while reducing the problem complexity. However, Jin et al. [17] discovered that generating sequences independently led to subpar results. This underscores the importance of developing models for protein co-design, which offer a strategic advantage by comprehensively addressing both structure and sequence design, thereby capturing their intricate relationship.

Among all deep generative modeling techniques, diffusion models have emerged as the predominant method in the field of protein design [2, 7–10, 16, 19, 21, 22, 26]. This is mainly due to their versatility, allowing for tasks such as inpainting [27] and the conditional generation [28] of proteins with desired properties. Diffusion models consist of a forward noising process that iteratively adds noise to the data and a reverse generative process that iteratively removes noise from the data. Within this framework, a neural network - hereafter referred to as the denoising neural network - is utilized to approximate the reverse generative process. This network is trained to remove noise from corrupted data and subsequently employed to denoise randomly sampled data, thereby generating new data samples.

Various denoising neural network architectures have been employed within a diffusion framework. Among these, Graph Neural Networks (GNNs) [29] have showcased their proficiency in generating chemical [30] and protein structures [2, 10]. They represent data as graphs and iteratively exchange and aggregate information between neighboring nodes. Meanwhile, within the domain of protein structure generation and protein co-design, there is a noticeable trend towards adopting denoising neural networks that are equivariant to rotations and translations. Equivariant denoising neural networks ensure that the probability distribution from which the protein structures are sampled remains invariant to any rotations or translations, applied to the protein structures [31]. This invariance improves the robustness,

parameter efficiency, and data efficiency of the neural network, and also enhances the quality of the generated structures.

For protein structure generation, the GNNs employed are the Equivariant Graph Neural Networks (EGNNs), proposed by Satorras, Hoogeboom, and Welling [32]. They have demonstrated considerable success in prior research focused on generating novel protein structures using diffusion [2, 10]. They have also proven to be highly effective in diffusion models that integrate coordinate features with categorical features [30]. This highlights their potential for use in protein co-design, which similarly relies on both coordinate and categorical features. However, EGNNs have not yet been leveraged for the task of protein co-design. To address this gap, we propose a novel approach that utilizes modified EGNNs as denoising neural networks within a diffusion modeling framework to co-design proteins.

Contrary to previous research that employs EGNNs as the denoising neural network for protein structure generation via diffusion [2, 10], we aim to generate both the protein structure and sequence. Another notable contrast lies in our modeling approach, which incorporates the coordinates of three atoms per amino acid in the protein, whereas both Fu et al. [2] and Trippe et al. [10] only consider the coordinates of one atom per amino acid. We opt to model the coordinates of three atoms along with the protein sequence because this combination holds promise for efficiently generating the coordinates of all other atoms in the protein structure using side chain packing algorithms [33]. While these methodological differences might open up numerous potential applications, they also significantly escalate the complexity of our task.

We summarize our contributions related to protein structure and sequence co-design as follows:

- We investigate the efficacy of EGNNs for the co-design task.
- We propose to update the coordinate operation in the EGNN architecture to improve its performance while maintaining its equivariance properties.
- We establish an evaluation pipeline to accurately assess a model's capabilities in protein sequence generation, protein structure generation, and protein co-design tasks.
- We define two new metrics to evaluate the correlation between the generated protein sequences and the generated protein structures.

This report is structured as follows: Section 2 provides a brief overview of key terms and concepts. Section 3 provides an overview of recent related work in the field. Section 4 outlines the methods used, including the diffusion process, neural network architecture, and dataset. Section 5 discusses the research questions and the experiments designed to answer them. Section 6 presents the results and includes a discussion of the findings. Finally, Section 7 concludes the report by summarizing the key findings and outlining potential avenues for future research.

2 Background

2.1 Proteins

A protein is composed of numerous amino acids linked together sequentially. For protein design tasks, proteins are typically represented using the protein sequence, the protein structure, or a combination of both. The protein sequence and protein structure representations are explained in the following subsections.

2.1.1 Protein Sequence

The protein sequence serves as a fundamental representation of a protein's composition and order of amino acids. In this representation, each amino acid is denoted by a single-letter code, simplifying the complex structure of proteins into a linear string of characters. For instance, the amino acid alanine is represented by the letter "A," and the amino acid lysine by "K," and so forth. By arranging these letters in a specific order that corresponds to the sequence of amino acids in the protein, we can achieve a simple yet informative representation of the protein.

2.1.2 Protein Structure

Each amino acid within a protein consists of multiple atoms, with the C_α (central atom), the C, and the N atoms being the most important. The C_α atom serves as the central point within the amino acids, while the C and N atoms are connected to it at specific angles. These three atoms form bonds between amino acids, connecting them to create proteins. The bonds formed by these atoms ultimately determine how the protein folds. Consequently, the coordinates of the C_α, the C, and the N atoms for each amino acid are commonly used to represent protein structures. A visual representation illustrating the arrangement of the C_α, C, and N atoms in an amino acid is presented in Fig 1.

2.2 Roto-Translation Equivariance

Roto-translation equivariance refers to a property of a function where its output remains consistent when the input undergoes rotations and translations. In mathematical terms, the function f is considered to be roto-translation equivariant if for any input x and any rotation matrix $\mathcal{R}$ and translation vector $\mathcal{T}$, the following holds:

$$f(\mathcal{R}x + \mathcal{T}) = \mathcal{R}f(x) + \mathcal{T} \tag{1}$$

In simpler words, if you rotate or translate the input data, the output of the function should undergo the same transformations.

2.3 Denoising Diffusion Probabilistic Models (DDPMs)

Based on the formulation described by Ho, Jain, and Abbeel [3], constructing a diffusion model requires establishing two processes defined by Markov chains over T timesteps. The first is the forward noising process, which spans from timestep $t = 0$ to $t = T$. This process incrementally adds noise to the data until it converges to the prior distribution, effectively mimicking random noise. The forward noising process is defined below:

$$q(\mathbf{x}^t|\mathbf{x}^{t-1}) = \mathcal{N}\left(\mathbf{x}^t \,\middle|\, \sqrt{1-\beta^t}\cdot\mathbf{x}^{t-1}, \beta^t\mathbf{I}\right), \tag{2}$$

where the data is denoted as $\mathbf{x}^0$ with dimensionality $\mathbf{x}_{dim}$, whereas $\mathbf{x}^{1:T}$ represent the progressively noisier versions of the data as it undergoes the forward noising process. The amount of noise

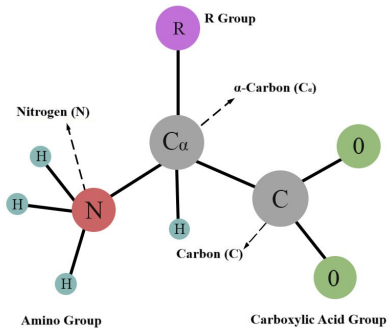


Figure 1. Visual depiction of an amino acid's chemical structure, showcasing its components: the R group, amino group, and carboxylic acid group connected to the central C_α atom. Emphasis is placed on highlighting the positions of the crucial C_α, C, and N atoms.

added is controlled by a variance schedule, β^t. $\mathbf{I}$ represents a $\mathbf{x}_{dim} \times \mathbf{x}_{dim}$ identity matrix. Since this process is Markovian, we can efficiently transition from $t = 0$ to any timestep:

$$q(\mathbf{x}^t|\mathbf{x}^0) = \mathcal{N}\left(\mathbf{x}^t \,\middle|\, \sqrt{\bar{\alpha}^t} \cdot \mathbf{x}^0, (1 - \bar{\alpha}^t)\mathbf{I}\right), \tag{3}$$

where $\alpha^t = 1 - \beta^t$ and $\bar{\alpha}^t = \prod_{\tau=1}^{t} \alpha^\tau$. We can then compute $\mathbf{x}^t$ using the reparameterization trick proposed by Ho, Jain, and Abbeel [3]:

$$\mathbf{x}^t = \sqrt{\bar{\alpha}^t}\mathbf{x}^0 + \sqrt{1 - \bar{\alpha}^t}\boldsymbol{\epsilon}, \tag{4}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$.

The second process is the reverse generative process, which operates in the opposite direction from $t = T$ to $t = 0$. Starting from the prior distribution, this process recursively removes noise from the data until it matches the desired distribution. A neural network, $\boldsymbol{\epsilon}_\theta(\mathbf{x}^t, t)$, is trained to approximate the reverse generative process using the data generated by the forward noising process. The reverse generative process is formally defined as:

$$p(\mathbf{x}^{t-1}|\mathbf{x}^t) = \mathcal{N}\left(\mathbf{x}^{t-1} \,\middle|\, \boldsymbol{\mu}_\theta(\mathbf{x}^t), \beta^t\mathbf{I}\right), \tag{5}$$

$$\boldsymbol{\mu}_\theta(\mathbf{x}^t) = \frac{1}{\sqrt{\alpha^t}}\left(\mathbf{x}^t - \frac{\beta^t}{\sqrt{1 - \bar{\alpha}^t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}^t, t)\right), \tag{6}$$

The neural network is trained to approximate the reverse generative process by minimizing the simplified loss introduced by Ho, Jain, and Abbeel [3], as defined below:

$$L(\theta) = \mathbb{E}_{t\sim\texttt{Uniform}(1\ldots T)}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0^t + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}, t\right)\right\|\right], \tag{7}$$

where t is uniformly sampled between 1 and T.

3 Related Work

In recent years, the utilization of deep learning methods for protein generation has witnessed a significant surge, spanning across both protein structures and sequences. In this section, we delve into the recent advancements in the development of deep learning methods for protein sequence generation, protein structure generation, and the co-design of protein structure and sequence.

3.1 Protein Sequence Generation

The array of deep learning methods employed for this task includes the use of Variational Autoencoders (VAEs) [11, 12], Generative Adversarial Networks (GANs) [13], and autoregressive Transformers [14, 15]. Additionally, diffusion models operating on latent embeddings of the protein sequence have also shown promising results. Alamdari et al. [26] employ a Dilated 1D convolution neural network architecture operating on learned embeddings of the protein sequence, Meshchaninov et al. [16] utilize Transformers operating on the ESM-2 [34] embeddings of the protein sequence.

3.2 Protein Structure Generation

Similar to protein sequence generation, various deep learning approaches have been applied to the task of protein structure generation. However, most of these efforts focus on generating the protein backbone rather than the all-atom protein structure. They choose to only model the protein backbone, consisting of the C_α, C and N atoms (see Section 2.1.2), because all amino acids contain these atoms and it is simpler to represent compared to the all-atom protein structure. The C_α, C and N atoms dictate how the protein folds. Their positions along with the protein sequence (which can be generated using inverse folding [23]) can be used in side-chain packing algorithms [33] to calculate the positions of all other atoms and obtain the all-atom protein structure [35].

While methods like GANs [4, 5] and VAEs [6] have been used, the field is predominantly dominated by diffusion models. This is primarily due to their numerous advantages over alternative generative models, like their capacity to seamlessly learn complex distributions, accommodate high-dimensional data, and generate highly diverse samples [36].

These diffusion models employ diverse protein backbone representations and utilize a variety of denoising neural network architectures. Wu et al. [7] employ Transformers as the denoising neural network and choose to represent the protein structure using 6 torsion angles between adjacent amino acid residues. This representation is inherently invariant to rotations and translations.

The predominant method to represent protein backbones involves utilizing the Cartesian coordinates of the C_α atom along

with an orientation matrix that describes the orientation of the $N - C_\alpha - C$ frame with respect to the C_α atom. This orientation matrix along with the C_α atom coordinate can then be used to calculate the coordinates of the C and N atoms. However, this representation requires the utilization of roto-translation equivariant neural networks since the probability distribution from which the protein structures are sampled is no longer inherently invariant to rotations and translations [31]. Watson et al. [8] utilize this representation with a fine-tuned pretrained RoseTTAFold [25] network as the denoising neural network. Yim et al. [9] also employ this representation but they incorporate the Equivariant transformer with invariant point attention (IPA) module from AlphaFold [24] as the denoising neural network instead.

A simpler approach to representing the protein structure involves using only the coordinates of the C_α atoms. While this simplification reduces the complexity of the problem, it confines the model to designing only the C_α atoms rather than the entire backbone. EGNNs have been successfully employed with this simplified representation to generate C_α-only protein backbones in both diffusion [10] and latent diffusion [2] settings.

3.3 Protein Co-Design

While protein structure and sequence co-design is a relatively new field, it follows the trends seen in protein structure generation, as protein structure generation is a component of protein co-design.

Many different techniques have been applied to co-design proteins. For instance, Jin et al. [17] utilize a graph-based model that generates a sequence in an autoregressive manner and iteratively refines the predicted global structure. Meanwhile, Anishchenko et al. [18] utilize trRosetta [37], an existing structure prediction network, to generate new proteins through Monte Carlo sampling in amino acid sequence space. Similarly, Wang et al. [20] employ both RoseTTAFold [25] and AlphaFold [24], existing structure prediction networks, to create new proteins by iteratively optimizing their networks within the sequence space.

However, the prevailing method is to use a diffusion model on a modified version of the protein structure representation used by Watson et al. [8] and Yim et al. [9], with a customised version of the IPA module from AlphaFold [24] as the denoising neural network. This modified representation includes the amino acid type in addition to the C_α atom coordinate, the orientation for each amino acid in the protein. This approach has been used by Luo et al. [19], Shi et al. [21], and Anand and Achim [22].

3.4 Insights

In our examination of related work on protein design, crucial insights have guided our distinctive approach and innovations, particularly in addressing identified challenges. Notably, diffusion emerges as the most common method for generating novel protein sequences, structures, or both together. Hence, we also opt to utilize diffusion in our approach. Another observation we made was that existing methodologies in protein structure generation and co-design predominantly rely on either inherently invariant protein representations or equivariant denoising neural networks, underscoring the significance of equivariance. We also observe that, despite the success of EGNNs in protein structure generation, their potential for the protein co-design task remains largely untapped. To bridge this gap and fully investigate their capabilities, we opt to incorporate EGNNs as the denoising neural network in our approach. We also noticed that existing works using EGNNs for protein structure generation tend to focus exclusively on modeling C_α atoms, neglecting other crucial atoms like C and N atoms. Consequently, we aim to further explore their capability to model multiple atoms. Moreover, we observed that the most commonly used representation includes the amino acid type, the C_α atom coordinate, and the orientation for each amino acid in the protein. However, the use of C_α, C, and N atom coordinates rather than orientation has not been investigated. We address this gap by exploring the use of C_α, C, and N atom coordinates. One advantage of this representation over orientation is its suitability for EGNNs, which are inherently equivariant in coordinate space.

4 Methods

In this section, we describe the protein representation, the diffusion processes, the denoising neural network architecture, our modification to the EGNN architecture, the hyperparameter tuning performed and the dataset used.

4.1 Protein Representation

The 3D structure of a protein is modeled as a fully connected graph, with each individual node representing a specific amino acid. We use fully connected graphs rather than explicit edges, as our neural network architecture predicts edge weights, which gives us a soft estimation of the presence of edges. This approach of utilizing a fully connected graph followed by edge weight prediction has proven successful in similar tasks, such as protein backbone structure generation [10].

The graph is denoted as $\mathcal{G} = \{(s_j, \mathbf{x}_j) | j = 1, ..., n\}$, where n denotes the total number of amino acids in the protein. Here, $\mathcal{G}_j$ represents information about the j^{th} amino acid in the graph. This includes the type of amino acid, denoted as $s_j \in \{ACDEFGHIKLMNPQRSTVWY\}$. It also includes the normalized x, y, and z coordinates for the C_α, C, and N atoms in the amino acid, denoted as $\mathbf{x}_j \in \mathbb{R}^9$. We apply normalization to the coordinates using the mean and standard deviation from the protein structures in the training set. This normalization transforms the coordinates into a standard normal distribution, which enables us to use Gaussian diffusion [3].

The representation explained above draws inspiration from the work of Luo et al. [19], which represents an amino acid by its type, the coordinates of the C_α atom, and the orientation to depict the protein structure. In contrast to the representation used by Luo et al. [19], we opt to utilize the coordinates of the C_α, C, and N atoms. This decision stems from the fact that the orientation of the individual amino acids is calculated using the coordinates of the C_α, C, and N atoms, effectively providing the network with the same information. Additionally, this choice aligns better with the neural network architecture used and simplifies the implementation of equivariance since the network is already roto-translation equivariant in the coordinate space.

4.2 Diffusion Processes

Given that the amino acid coordinates ($\mathbf{x}_j$) are continuous, while the amino acid types (s_j) are categorical, it is necessary to establish two separate diffusion processes that operate simultaneously. We use Gaussian diffusion [3] for continuous values and multinomial diffusion for categorical values [38] following the approach of Luo et al. [19]. Fig. 2 offers a high-level overview of how the Gaussian and multinomial diffusion processes are utilized to train the denoising neural network.

We denote the type and coordinates of a node j at timestep t as s_j^t and $\mathbf{x}_j^t$, respectively. The entire structure and sequence sampled at timestep t are represented as $\mathcal{G}^t = \{(s_j^t, \mathbf{x}_j^t) | j = 1, ..., n\}$. The following subsections detail the diffusion processes employed for the amino acid coordinates and the amino acid types.

4.2.1 Diffusion for Amino Acid Coordinates

The diffusion process for amino acid coordinates $\mathbf{x}_j$ follows the method outlined by Luo et al. [19], which is based on the Gaussian diffusion process defined by Ho, Jain, and Abbeel [3] (See Section 2.3).

The key difference from the DDPM formulation is that instead of processing the entire data sample at once, we focus on the coordinates of a single node in the graph ($\mathbf{x}_j$) at a time, conditioned on the entire graph ($\mathcal{G}$), which includes the amino acid types.

To establish the forward noising process, we update Eq. 4 to operate on the normalized amino acid coordinates of an individual node, as defined below:

$$\mathbf{x}_j^t = \sqrt{\bar{\alpha}_{\text{pos}}^t}\mathbf{x}_j^0 + \sqrt{1 - \bar{\alpha}_{\text{pos}}^t}\boldsymbol{\epsilon}_j, \tag{8}$$

where $\boldsymbol{\epsilon}_j \sim \mathcal{N}(0, \mathbf{I})$, $\alpha_{\text{pos}}^t = 1 - \beta_{\text{pos}}^t$ and $\bar{\alpha}_{\text{pos}}^t = \prod_{\tau=1}^t \alpha_{\text{pos}}^\tau$. The rate of diffusion, parameterized by β_{pos}^t, follows the cosine noise schedule used by Luo et al. [19].

To establish a reverse generative process, we similarly adapt Eq. 5 and Eq. 6 to operate on the normalized amino acid coordinates of an individual node, conditioned on the entire graph, as defined below:

$$p(\mathbf{x}_j^{t-1}|\mathcal{G}^t) = \mathcal{N}\left(\mathbf{x}_j^{t-1} \,\middle|\, \boldsymbol{\mu}_p(\mathcal{G}^t), \beta_{\text{pos}}^t\mathbf{I}\right), \tag{9}$$

$$\boldsymbol{\mu}_p(\mathcal{G}^t) = \frac{1}{\sqrt{\alpha_{\text{pos}}^t}}\left(\mathbf{x}_j^t - \frac{\beta_{\text{pos}}^t}{\sqrt{1 - \bar{\alpha}_{\text{pos}}^t}}\text{Dzyn[c]}(\mathcal{G}^t)[j]\right), \tag{10}$$

where Dzyn[c]$(\cdot)[j]$ is a part of the neural network defined in Section 4.5. It takes in the entire protein graph at timestep t and predicts the standard Gaussian noise $\boldsymbol{\epsilon}_j \sim \mathcal{N}(0, I)$ added in the forward noising process defined in Eq. 8.

We also adjust the loss term in Eq. 7, employed for training the neural network. An important modification is the removal of the expectation previously present, which will be reintroduced later when combining the loss for amino acid coordinates with the loss for amino acid types.

$$L_{\text{pos}}^t = \frac{1}{n}\sum_j \|\boldsymbol{\epsilon}_j - \text{Dzyn[c]}(\mathcal{G}^t)\|^2 \tag{11}$$

4.2.2 Diffusion for Amino Acid Types

The diffusion process for amino acid types s_j follows the method outlined by Luo et al. [19], which is based on the multinomial diffusion process defined by Hoogeboom et al. [38].

The core concept of multinomial diffusion closely resembles that of DDPMs. However, the formulation differs significantly due to its focus on modelling multinomial distributions. Similar to Eq. 2 and Eq. 3, we establish a forward process for multinomial distributions, specifically tailored to the amino acid type (s_j) of a given node in the graph, defined as follows:

$$q(s_j^t|s_j^{t-1}) = \texttt{Mt}\left((1 - \beta_{\text{type}}^t)\cdot \texttt{onehot}(s_j^{t-1}) + \beta_{\text{type}}^t \cdot \frac{1}{20}\cdot \mathbf{1}\right), \tag{12}$$

where the function $\texttt{onehot}(\cdot)$ transforms amino acid types into 20 dimensional one-hot vectors, $\texttt{Mt}(\cdot)$ represents a multinomial distribution, and $\mathbf{1}$ denotes a 20 dimensional vector with all elements set to one. Similar to the diffusion process defined for amino acid coordinates, β_{type}^t follows the cosine noise schedule used by Luo et al. [19]. Given the Markovian nature of this process, we can efficiently transition from $t = 0$ to any timestep:

$$q(s_j^t|s_j^0) = \texttt{Mt}\left(\bar{\alpha}_{\text{type}}^t \cdot \texttt{onehot}(s_j^0) + (1 - \bar{\alpha}_{\text{type}}^t)\cdot \frac{1}{20}\cdot \mathbf{1}\right), \tag{13}$$

where $\bar{\alpha}_{\text{type}}^t = \prod_{\tau=1}^t (1 - \beta_{\text{type}}^\tau)$.

Based on Eq. 12 and Eq. 13 we derive the following posterior [38]:

$$q(s_j^{t-1}|s_j^t, s_j^0) = \texttt{Mt}\Bigg(\left[\alpha_{\text{type}}^t \cdot \texttt{onehot}(s_j^t) + (1 - \alpha_{\text{type}}^t)\cdot \frac{1}{20}\cdot \mathbf{1}\right] \odot \left[\bar{\alpha}_{\text{type}}^{t-1} \cdot \texttt{onehot}(s_j^0) + (1 - \bar{\alpha}_{\text{type}}^{t-1})\cdot \frac{1}{20}\cdot \mathbf{1}\right]\Bigg). \tag{14}$$

We establish a reverse generative process for multinomial distributions, specifically tailored to the amino acid type of a single node conditioned on the entire graph, similar to Eq. 5, defined as follows:

$$p(s_j^{t-1}|\mathcal{G}^t) = \texttt{Mt}\left(\text{Dzyn[h]}(\mathcal{G}^t)[j]\right), \tag{15}$$

where Dzyn[h]$(\cdot)[j]$ is a part of the neural network defined in Section 4.5. It takes in the entire protein graph at timestep t and predicts the amino acid type for the j^{th} amino acid in the protein at timestep $t - 1$.

To effectively denoise the amino acid types, the reverse generative process, as described by $p(s_j^{t-1}|\mathcal{G}^t)$, should approach the posterior $q(s_j^{t-1}|s_j^t, s_j^0)$, defined in Eq. 14. Hence, to train the neural network we minimize the expected Kullback–Leibler (KL) divergence between these distributions, as shown below:

$$L_{\text{type}}^t = \frac{1}{n}\sum_j D_{KL}\left(q(s_j^{t-1}|s_j^t, s_j^0)\middle\| p(s_j^{t-1}|\mathcal{G}^t)\right) \tag{16}$$

4.3 Training

The loss function used to train the neural network is expressed as an expectation with respect to t over the weighted sum of the losses defined in Eq. 16 and Eq. 11.

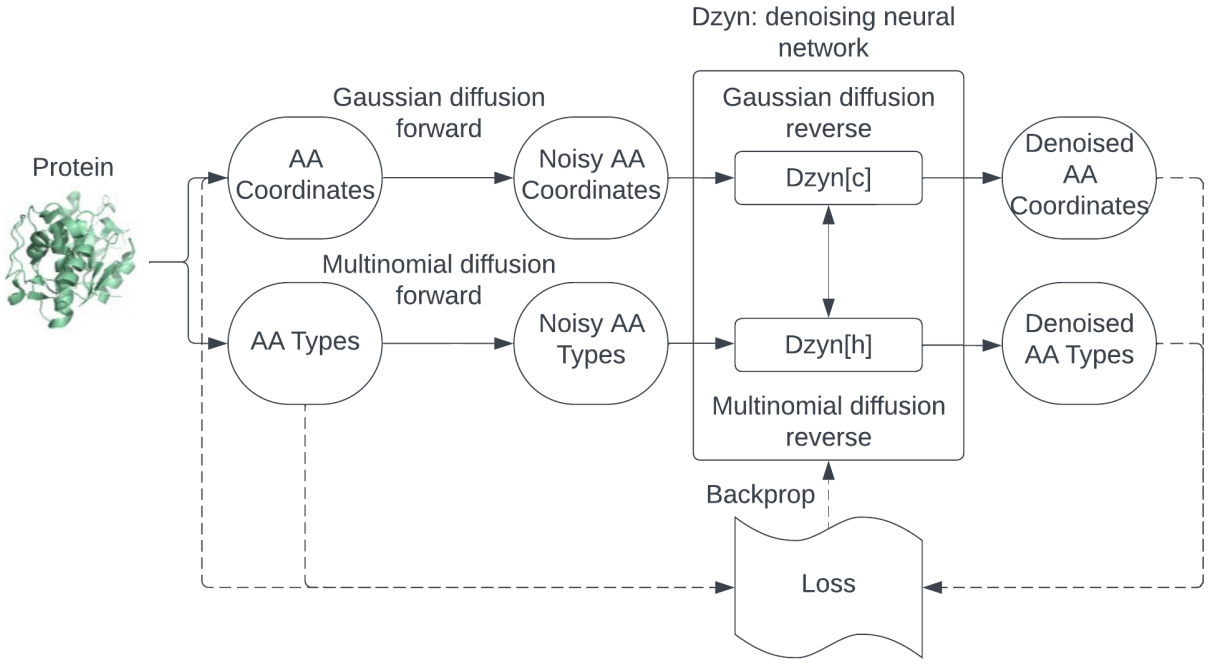


Figure 2. A high-level overview of the training process showcasing how the Gaussian and multinomial diffusion processes work simultaneously. First, the forward noising processes are used to add noise to the amino acid (AA) coordinates and the AA types. Then the neural network is trained to approximate the reverse generative processes by denoising the noisy coordinates and types. The neural network consists of two parts that work concurrently, exchanging information with each other. Dzyn[c] approximates the reverse Gaussian diffusion process and Dzyn[h] approximates the reverse multinomial diffusion process.

$$L = \mathbb{E}_{t\sim \texttt{Uniform}(1\ldots T)}\left[w_{\text{type}} \cdot L^t_{\text{type}} + w_{\text{pos}} \cdot L^t_{\text{pos}}\right] \tag{17}$$

The diffusion model is trained by randomly sampling t from 1 to T, where we set $T = 1000$. The neural network takes the noisy protein $\mathcal{G}^t$, obtained using Eq. 13 and Eq. 8, as the input. It produces $\boldsymbol{\epsilon}^{pred}$ and s^{pred} as the output, which are then used to calculate the loss defined in Eq. 17 and update the networks parameters. $\boldsymbol{\epsilon}^{pred}$ and s^{pred} can also be used to calculate $\mathcal{G}^{t-1}$ which is used in the generation process described in Section 4.4. The training process is illustrated in Fig. 3.

4.4 Sampling

To generate a novel protein sample, we begin by randomly selecting a length, denoted as L, from the distribution of protein lengths within our training set. With this length determined, we construct a fully connected graph comprising L nodes. Each node in this graph represents an amino acid in the new protein. These nodes are then initialized with an amino acid type, chosen from a uniform distribution over the 20 types. Additionally, for each node, the positions for the C_α, C, and N atoms are also initialized by sampling from the standard normal distribution. This new graph is denoted as $\mathcal{G}^T$, where $\mathcal{G}^T = \{(s_j^T \sim \text{Uniform}(20), \mathbf{x}_j^T \sim \mathcal{N}(0, \mathbf{I}) | j = 1, ..., L\}$.

Dzyn is then iteratively utilized to sample $\mathcal{G}^{t-1}$ from $\mathcal{G}^t$, beginning at time step $t = T$ with input $\mathcal{G}^T$, and continuing until $t = 0$. This process results in the generation of a novel protein $\mathcal{G}^0$. The sampling process is illustrated in Fig. 4.

4.5 Neural Network Architecture

The neural network utilized to model the reverse generative process is a modified adaptation of the Equivariant Graph Neural Network (EGNN) introduced by Satorras, Hoogeboom, and Welling [32]. They operate by partitioning node features from a standard Graph Convolution Network (GCN) into two separate categories: node features and coordinate features. Node features are dedicated to features that do not require equivariance, like the amino acid sequence in our case, while coordinate features are used for features that require rotational and translational equivariance, such as the C_α, C, and N atom coordinates. The network achieves equivariance by utilizing the distance between the coordinate features of nodes rather than directly incorporating the coordinate features within the network.

We designate the neural network as Dzyn, comprising two primary components: Dzyn[h] and Dzyn[c]. Dzyn[h] is tasked with denoising the node features (amino acid types), while Dzyn[c] focuses on denoising the coordinate features (amino acid coordinates). Although we simplify our explanation by splitting the network into these two components, it is important to emphasize that both Dzyn[h] and Dzyn[c] utilize all the information from

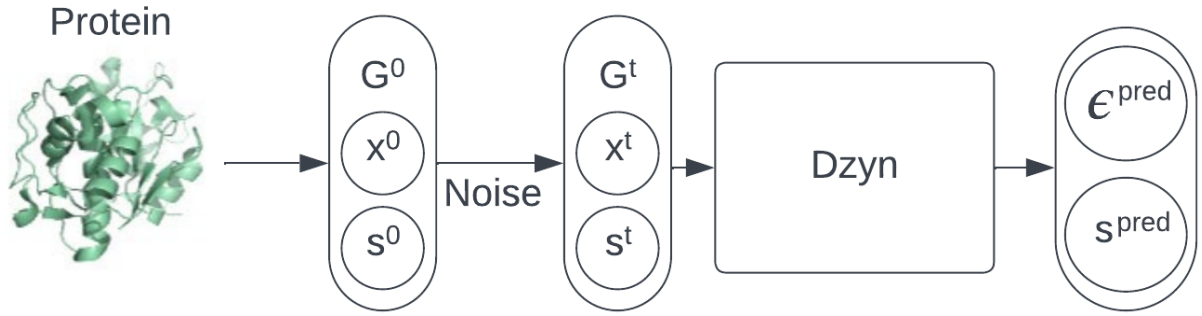


Figure 3. An overview of the neural network training procedure. First the protein is converted to a graph $\mathcal{G}^0$ which consists of the C_α, C, and N atom coordinates (x^0) and amino acid types (s^0) for each node. Noise is added to x^0 and s^0 for t timesteps to get x^t and s^t, respectively. The neural network is then trained to predict ϵ^{pred} (the noise added to x^0 to get x^t) and s^{pred} (a prediction for s^0).

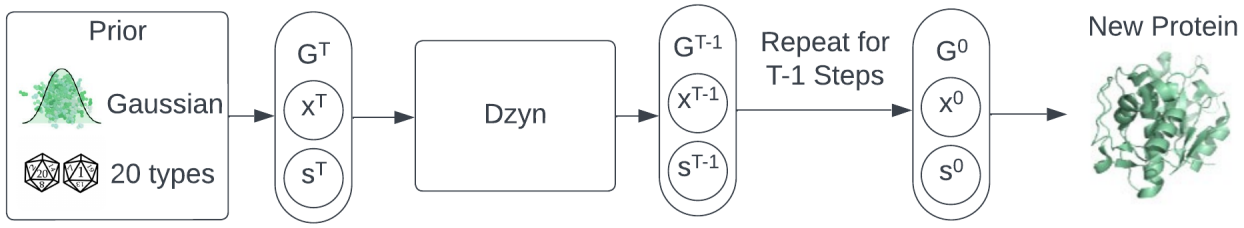


Figure 4. An overview of the generation procedure. First an arbitrary length is chosen and a graph, $\mathcal{G}^T$, of this length is created. For each node in the graph, coordinates are sampled from a Gaussian distribution and an amino acid type is sampled from a uniform distribution over the 20 types. $\mathcal{G}^T$ is then iteratively refined using the trained neural network (Dzyn) to generate $\mathcal{G}^0$, a novel protein.

node and coordinate features, as well as exchange information with each other.

In accordance with the methodology proposed by Kingma et al. [39], we concatenate the sinusoidal encoding of the diffusion timestep and the sequence position to the node features before they are fed into the neural network. We use sinusoidal encodings of the diffusion timestep because they provide a structured and interpretable way for the model to understand temporal patterns and dependencies, essential for effectively capturing the evolution of data distributions over time in diffusion models. Additionally, we use sinusoidal encodings of the sequence position to represent the sequence order allowing the model to capture positional dependencies within the data, ensuring that the model can effectively process protein sequences and understand relationships between the nodes. These encodings are also concatenated to the intermediate output node features of each Equivariant Graph Convolution layer (EGCL) before they are fed into the next EGCL. Dzyn consists of 8 EGCLs, with linear layers between each layer to encode and decode the concatenated node features to and from the hidden dimension. A visual representation of the overall neural network architecture is shown in Fig 5.

The neural network was trained for 500 epochs over a two week period on a NVIDIA A40 GPU. The hyperparameter configuration used and the hyperparameter tuning performed is described in Section 4.7.

4.6 Modification to EGNNs

In the original EGNN proposed by Satorras, Hoogeboom, and Welling [32] the equations used to compute the output of the network are as follows:

$$m_{ij} = \phi_e\left(h_i^l, h_j^l, \left\|c_i^l - c_j^l\right\|^2, a_{ij}\right), \tag{18}$$

$$c_i^{l+1} = c_i^l + C\sum_{j\neq i}\left(c_i^l - c_j^l\right)\phi_x\left(m_{ij}\right), \tag{19}$$

$$m_i = \sum_{j\neq i} m_{ij}, \tag{20}$$

$$h_i^{l+1} = \phi_h\left(h_i^l, m_i\right), \tag{21}$$

where c_i^l and h_i^l represent the coordinate and node features for i^{th} amino acid at layer l, respectively. The term m_{ij} signifies the edge embedding message from node j to node i. a_{ij} represents the edge attributes. ϕ_e, ϕ_x, ϕ_h are edge, coordinate and node operations approximated by MLPs.

The term $C\sum_{j\neq i}\left(c_i^l - c_j^l\right)\phi_x\left(m_{ij}\right)$ in Eq. 20 is a weighted sum of the relative distances $(c_i^l - c_j^l)$ for all nodes $j \neq i$. Here, $\phi_x : \mathbb{R}^f \to \mathbb{R}^1$ is an MLP that outputs a scalar weight based on the edge embedding message, m_{ij} of dimension f, between the two nodes. C is an arbitrary constant.

Using a single scalar weight as the output of ϕ_x restricts the

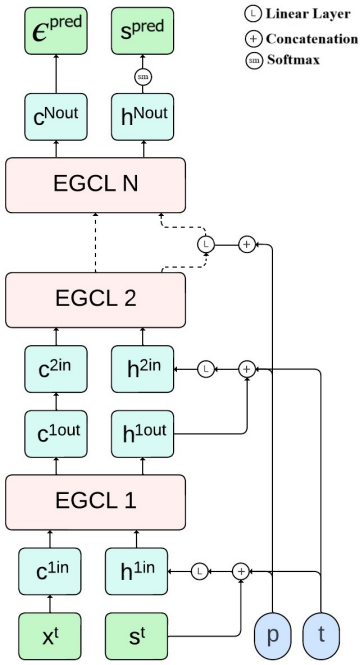


Figure 5. The architecture of the neural network used. Here, $\mathbf{x}^t$ and s^t represent the amino acid coordinates and types at timestep t respectively. p and t represent the sinusoidal positional and timestep encodings. $\boldsymbol{\epsilon}^{\text{pred}}$ represents the neural network's prediction for the noise added to the amino acid coordinates. s^{pred} represents the neural networks prediction for the amino acid types at timestep 0.

network's ability to capture complex patterns, as it forces the model to apply the same weight to all coordinates in vector c. This may work well for straightforward models where each node represents just one particle in 3D space. However, this approach becomes limiting when nodes need to represent multiple particles, like in our task where each node encapsulates three particles in the 3D space. To overcome this limitation, we propose updating $\phi_x : \mathbb{R}^f \rightarrow \mathbb{R}^1$ to $\phi_x : \mathbb{R}^f \rightarrow \mathbb{R}^{c_{dim} \times c_{dim}}$. In this context, c_{dim} refers to the number of coordinate features per node. For our task, $c_{dim} = 9$, since we have 9 coordinate features in total: 3 (x, y, and z) for each of the 3 atoms (C_α, C, and N) in a node. This modification transforms ϕ_x into a matrix that serves as the weights for the weighted sum operation.

By transitioning to a matrix output, we not only preserve the dimensionality of the input vector but also improve the networks capability to discern and learn intricate patterns within the data. With ϕ_x now outputting a matrix, each element in the vector c can influence others, fostering a richer interaction among the coordinates. Moreover, considering that each node represents three atoms, this approach enables specific atoms to directly influence the coordinates of other atoms, thereby enhancing the model's capacity to capture atomic interactions. This enhanced capability was previously unattainable when utilizing a scalar weight as the output of ϕ_x. Importantly, this adjustment ensures that the network retains its equivariance properties.

An ablation study was conducted to verify whether the modification improved the performance of the EGNN. The network used in the ablation study was trained for only one week instead of the intended two weeks due to time constraints. The results of this study are presented in Fig. 6. These results clearly indicate that the modification significantly reduces the loss on the coordinate features (C_α, C, and N positions) while having no substantial effect on the node features (amino acid sequence). Overall, this modification improves the EGNN's performance, enabling it to learn more complex patterns from the data in the coordinate space.

4.7 Hyperparameter Tuning

EGNNs have numerous hyperparameter options, along with several other hyperparameters that govern the training process. Each of these parameters can influence the network's performance, thereby directly affecting the quality of the generated proteins. Consequently, hyperparameter tuning is essential to optimize the performance of the denoising neural network and generate high quality proteins.

However, due to the considerable time and resources required for exhaustive hyperparameter tuning via grid search, with each combination of the many hyperparameters taking two weeks to fully converge, such an approach was deemed impractical. Instead, ablation studies were conducted to assess the impact of specific hyperparameters on models trained for 100 epochs to accommodate time constraints. For each run, one hyperparameter was modified, and the lowest validation loss value throughout the run was documented. Hyperparameters yielding reduced loss were selectively integrated into the final model. Table 1 details the hyperparameter configurations for both the baseline model used in the ablation study and the final model with the best hyperparameters found.

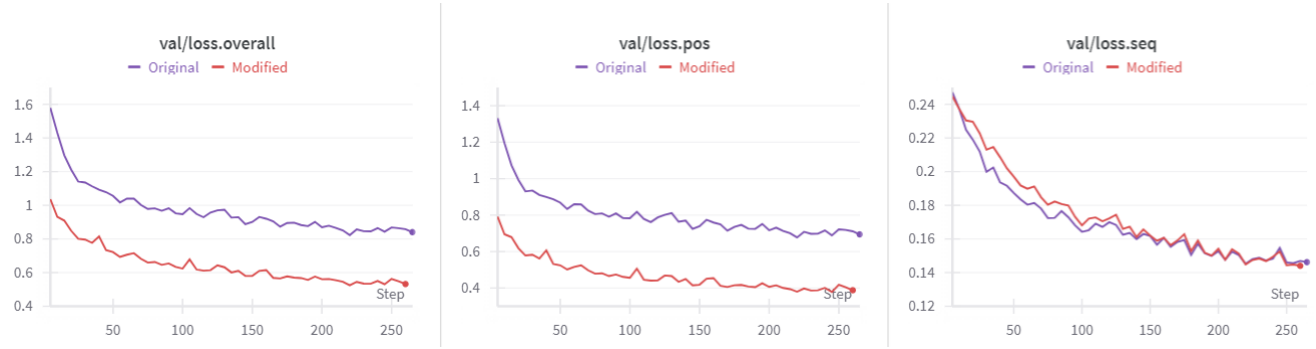


Figure 6. Comparison of validation loss values for the Original EGNN (purple) and the Modified EGNN (red). The charts illustrate the overall loss (left), the loss on the C_α, C, and N positions (center), and the loss on the amino acid sequence (right). The Modified EGNN demonstrates significantly better performance in predicting the positions, while its performance on the sequence remains largely unchanged compared to the Original EGNN.

Table 1
Hyperparameter configurations for the baseline model utilized in the ablation study and for the final model with the best hyperparameters found.

Hyperparameter	Baseline	Final
EGNN layers	4	8
MLP layers	0	0
Optimizer	Adam	Adam
Learning rate	1.0×10^{-5}	1.0×10^{-5}
Diffusion timesteps	100	1000
Batch size	2	2
Positional weight (w_{pos})	1.0	1.0
Sequence weight (w_{type})	1.0	1.0
EGNN Hidden dimension	1024	1024
EGNN Edge weight prediction	False	True
EGNN Residual connections	True	False
EGNN Coordinate aggregation method	mean	mean
EGNN tanh activation for coordinate MLP	False	False
EGNN Activation function	Sigmoid Linear Unit (SiLU)	SiLU

Numerous models were created, each altering one of these hyperparameters to assess its impact on the model's performance. These adjusted models were subsequently compared with the baseline model, and the alterations were either incorporated for enhanced performance, or rejected due to diminished performance, prolonged convergence time or instability. Table 2 presents the hyperparameters modified for the ablation study, alongside the resulting best validation loss and whether they were included or rejected (and the reason for rejection if they were rejected).

In Table 2, we observe that the final model, optimized with the best hyperparameters, performs significantly better than the baseline model. Consequently, this model is expected to produce proteins of much higher quality.

4.8 Dataset And Preprocessing

The dataset utilized comprises protein structures and sequences of short monomers (single-chained proteins). Due to the limited availability of experimentally obtained protein structures, AlphaFold predictions are used for protein sequences sourced from the Swiss-Prot dataset [40]. The protein sequences are sourced from the Swiss-Prot dataset because it provides high-quality sequences meticulously annotated by experts. An advantage of using structures predicted by AlphaFold is that it exclusively predicts monomers, simplifying the problem. This eliminates the need to manage the complexity of multiple chains within the protein structure, allowing for more straightforward training and analysis process.

We refine our dataset by filtering based on two criteria: the number of amino acid residues in each protein and the confidence level of the AlphaFold prediction, indicated by the predicted Local Distance Difference Test (pLDDT) score [24]. We focus on proteins with 50 to 100 residues, balancing computational feasibility with utility. Additionally, we exclude structures with an average pLDDT score below 70, as scores above this threshold typically indicate well-modeled backbone structures [24]. This process yields 38,595 proteins, which are then divided into training and validation sets using a 90:10 split ratio.

5 Experiments

This study aims to answer the following research questions:

How effectively can graph-based diffusion models facilitate the co-design of protein structures and sequences?

- Do the protein sequences generated using graph-based diffusion models exhibit high foldability and similarity to natural sequences?

Table 2
A list of all the hyperparameters modified for the ablation study, including the Best Validation Loss and Outcome (Included/Rejected), along with the reason for rejection.

Hyperparameter	Val Loss (↓)	Outcome
Baseline	0.7484	-
Final	0.6343	-
EGNN layers = 8	0.7159	Included
EGNN layers = 16	-	Rejected (Runtime)
MLP layers = 4	0.7514	Rejected (Runtime & Performance)
Learning rate = 5.0×10^{-5}	-	Rejected (Instability)
Diffusion timesteps = 1000	0.6939	Included
Batch size = 1	0.7203	Rejected (Instability)
Hidden dimension = 2048	0.7172	Rejected (Runtime)
Hidden dimension = 512	0.7883	Rejected (Performance)
EGNN Edge weight prediction = True	0.7111	Included
EGNN Residual connections = False	0.7038	Included
EGNN Coordinate aggregation method = sum	-	Rejected (Instability)
EGNN tanh activation for coordinate MLP = True	0.7786	Rejected (Performance)

- Are the protein structures generated using graph-based diffusion models designable and novel?
- To what extent do the protein structures and sequences co-designed using graph-based diffusion models correlate with each other?

To address these research questions, we employed a sampling strategy consistent with Trippe et al. [10], wherein we sampled 500 proteins ranging in length from 50 to 99 residues, with 10 proteins sampled for each length. Following this established sampling procedure allows for a direct comparison of structure designability with LatentDiff [2] and ProtDiff [10].

In the following subsections, we explain our evaluation objectives, discuss their importance, and describe the metrics used to assess the generated proteins.

5.1 Sequence Foldability

To evaluate the quality of the generated sequences, it is essential to thoroughly assess their foldability. Foldability, the ability of a protein sequence to adopt a stable three-dimensional structure, is a fundamental characteristic of proteins. A protein sequence that cannot fold into a stable structure cannot be regarded as a true protein sequence. Therefore, ensuring high foldability is crucial for generating protein sequences that are both biologically meaningful and functionally useful.

A common approach to assess the foldability of a generated sequence involves employing deep learning based structure prediction models [24, 25, 41, 42] to predict its structure. The predicted local distance difference test (pLDDT) score provided by these models indicates their confidence in the predicted structure. In line with previous studies by Meshchaninov et al. [16] and Alamdari et al. [26], where this metric was employed to assess protein sequence foldability, we also adopt it in our research.

Various deep learning-based structure prediction models, including OmegaFold, AlphaFold, ESMFold, and RoseTTAFold, can be used to evaluate the foldability of the generated sequence. However, we opt to utilize OmegaFold over AlphaFold because of its faster runtime, and over ESMFold and RoseTTAFold due to its superior accuracy, thus achieving a balanced compromise between speed and precision [41, 43].

5.2 Sequence Similarity to Natural Sequences

Assessing the similarity of generated sequences to natural sequences provides valuable insights into the fidelity of the model's outputs. Natural protein sequences have evolved over time to fulfill specific biological functions, and they exhibit characteristic patterns and features [44]. Therefore, if the generated sequences closely resemble natural sequences, it suggests that the model has successfully captured the underlying distribution of protein sequences in nature. Such similarity enhances the likelihood that the generated sequences are biologically plausible and functionally relevant, contributing to our confidence in their utility for further analysis and experimentation.

To assess the similarity of generated sequences to natural protein sequences, we employ the pseudo perplexity metric defined by Meshchaninov et al. [16]. The pseudo perplexity is calculated using ESM-2 [34], a language model trained on protein sequences. This involves masking each token (amino acid) in the sequence and subsequently predicting it using ESM-2 based on all the other tokens, then computing the average loss for each prediction to determine the pseudo perplexity. The ESM-2 pseudo perplexity (ESM-2 pppl) score of a generated protein sequence gauges how effectively it aligns with the learned patterns of the ESM-2 model based on its training set (UniRef50, a clustered subset of UniProt [40]).

5.3 Structure Designablity

A protein structure is considered designable if a corresponding sequence that folds into it can be identified. The significance of ensuring the designability of a generated protein structure lies in the fact that a protein structure lacking a sequence that folds into it cannot be considered a true protein structure. Therefore, it is crucial to ensure that the generated protein structures are designable.

A widely used method to evaluate the designability of generated

protein structures is to compute and analyse the Self Consistency Template Modeling (scTM) score [2, 7, 9, 10]. To calculate the scTM score for each protein structure generated by our model Dzyn, we first predict 8 sequences for the generated protein structure. These sequences are predicted using Protein MPNN [23], a deep learning-based inverse folding method that takes a protein structure and predicts sequences that could fold into that structure. Subsequently, we employ OmegaFold [41] to predict protein structures for these inverse folded sequences. These OmegaFold predicted protein structures are then compared to our Dzyn generated protein structure using the Template Modeling (TM) score [45], which is calculated by aligning the C_α atom positions in protein structures, serving as a measure of their structural similarity. This yields a list of 8 TM scores, from which we select the maximum as the scTM score. The formula to compute the scTM score is shown below:

$$\text{scTM} = \max\left(\{\text{TM}(\text{OmegaFold}(\text{PMPNN}(X)_i), X)\}_{i=1}^{8}\right), \quad (22)$$

where X represents a protein structure generated using Dzyn and $\text{PMPNN}(X)_i$ represents the i^{th} protein sequence inverse folded for it using Protein MPNN.

A TM score greater than 0.5 suggests that two structures share the same fold [45]. Thus, we consider a structure designable if scTM > 0.5 [10]. The fact that OmegaFold can generate a similar structure suggests the existence of a corresponding sequence for the Dzyn generated structure, thereby confirming its designability.

5.4 Structure Novelty

In the context of protein structure generation, novelty is a critical requirement. Specifically, the generated protein structures should differ from those present in the training dataset. This emphasis on novelty stems from the fact that a model that merely memorizes training data without grasping the underlying patterns and producing new data lacks practical utility. Consequently, ensuring the novelty of generated structures becomes essential.

We assess the similarity between each generated structure and any training structure by computing the maximum TM score across the entire training dataset. This metric, known as the Training TM score [7], enables us to evaluate the structural similarity of the generated protein structures to those in the training set. Subsequently, we analyze the distribution of these Training TM scores for all the generated structures. If this distribution closely approaches a value of 1, it indicates that the model is essentially memorizing the training set. The formula to compute the Training TM score is given below:

$$\text{TrainingTM} = \max_{X_{\text{train}} \in \text{Training Set}} \left(\text{TM}(X, X_{\text{train}})\right), \quad (23)$$

where X represents a protein structure generated using Dzyn, and X_{train} represents a protein structure in the training set.

5.5 Co-Design Correlation

The task of protein structure and sequence co-design is a relatively new frontier, lacking established metrics to evaluate the correlation between generated protein sequences and structures. While models may produce high-quality structures and sequences, ensuring that the generated sequences indeed fold into the corresponding structures remains a challenge. Hence, there's a critical need to assess the correlation between generated protein sequences and structures.

To address the current limitations in evaluating co-design correlation, we introduce two novel metrics: the Cross Consistency TM (ccTM) score and Amino Acid Consistency (AAC). The ccTM score assesses how well the designed amino acid sequence folds into the predicted protein structure, whereas AAC evaluates how well the predicted structure predicts the original sequence. By introducing these metrics, we pave the way for robust evaluation and advancement in protein co-design research.

5.5.1 Cross-Consistency TM Score

To assess whether the generated sequence folds into the generated structure, we introduce a new metric called Cross-Consistency TM (ccTM) score, drawing inspiration from the widely used scTM score metric. Unlike scTM, which generates eight sequences using Protein MPNN, we directly utilize the sequence generated by Dzyn. Subsequently, a protein structure is predicted for the sequence using OmegaFold. Following this, we calculate the TM score between the OmegaFold generated protein structure and the Dzyn generated protein structure, termed the ccTM score. The formula to compute the ccTM score is given below:

$$\text{ccTM} = \text{TM}(\text{OmegaFold}(S), X), \quad (24)$$

where where X represents the protein structure generated using Dzyn and S represents the protein sequence generated using Dzyn.

Similar to the scTM threshold, a ccTM score exceeding 0.5 suggests that both the OmegaFold generated protein structure and the Dzyn generated protein structure share the same fold. This implies that the Dzyn generated sequence indeed folds into the Dzyn generated structure, indicating correlation.

5.5.2 Amino Acid Consistency

To determine whether the generated structure inversely folds into the generated sequence, we introduce the Amino Acid Consistency (AAC) metric. To compute the AAC we first use Protein MPNN to predict a single sequence for the Dzyn generated structure. We then compute the similarity of the Dzyn generated sequence to the Protein MPNN predicted sequence. Specifically, we calculate the percentage of amino acids that match at each position. This similarity percentage is denoted as AAC. The formula for calculating AAC is given below:

$$AAC = \frac{Number\ of\ matching\ amino\ acids}{Total\ number\ of\ amino\ acids} \times 100\%. \quad (25)$$

6 Results and Discussion

This section presents the results of our experiments evaluating the co-design of protein structures and sequences using graph-based diffusion models. We used various metrics to comprehensively assess the generated sequences, structures, and the correlation between them. The following subsections detail the findings and

discuss their implications for the performance and reliability of our model.

6.1 Sequence Quality

To evaluate the quality of the generated sequences, we used the average OmegaFold pLDDT and the average ESM-2 pppl over the set of sequences generated using Dzyn. We compared our model with several other protein sequence generation models [13, 16, 26, 34, 46–48]. We also compared it with a set of 500 protein sequences generated by random sampling from a uniform distribution over the amino acids and a set of 500 protein sequences chosen from the training set. Both of these sets were curated following the same length sampling strategy described in Section 5. Additionally, to evaluate the effect of varying the number of diffusion timesteps (T), we also assessed a version of Dzyn with only 100 timesteps instead of 1000.

Table 3 presents the comparison of Dzyn to various protein sequence generation models using the average pLDDT (along with the protein structure prediction model used to compute the pLDDT score) and the average ESM-2 pppl scores over the generated sequences. The scores for the other sequence generation models used for comparison were obtained from Meshchaninov et al. [16] and Alamdari et al. [26]. Meshchaninov et al. [16] utilize ESMFold to compute the pLDDT scores, while Alamdari et al. [26] use OmegaFold (same as us). Therefore, a direct comparison to the results of Meshchaninov et al. [16] might be unfair, but it provides a rough indication of how the models compare.

We found that out of the 500 sequences generated using Dzyn, 153 (approximately 30%) had an OmegaFold pLDDT score greater than 70, suggesting they are highly foldable. In Table 3, we observe that the quality of the protein sequences generated using Dzyn is second only to DiMA [16], the current state-of-the-art model for protein sequence generation. However, it is important to note that DiMA was trained solely to design protein sequences, whereas Dzyn was trained to co-design both the protein sequence and structure, which is a significantly harder task. Since the average ESM-2 pppl of the sequences generated using Dzyn is close to that of the training set, we can infer that they are very similar to natural sequences. This suggests that Dzyn can be used to generate high-quality protein sequences that are highly foldable and similar to natural sequences. Another noteworthy observation from the results presented in Table 3 is the significant improvement in the quality of generated sequences when we use the final hyperparameter configuration with $T = 1000$ instead of the baseline hyperparameter configuration with $T = 100$ (See Section 4.7).

Fig. 7a shows the scatter plot of the OmegaFold pLDDT scores versus the sequence length. Fig. 7b displays a scatter plot of ESM-2 pppl scores against sequence length. Lastly, Fig. 7c exhibits a scatter plot of the OmegaFold pLDDT scores versus ESM-2 pppl scores. Based on Fig. 7a, we can say that there is no correlation between the length and the foldability of the generated protein sequences. Fig. 7b indicates that the ESM-2 pppl decreases as the size of the protein increases, possibly due to inherent biases in the ESM-2 model. This is because longer protein sequences are more frequently present in both the training set used to train ESM-2 and in nature. Fig. 7c suggests there is no direct correlation between the OmegaFold pLDDT and the ESM-2 pppl of the generated sequences. Therefore, the analysis reveals that while certain biases exist in the ESM-2 model, there is no clear relationship between sequence length and foldability, or between foldability and similarity to natural sequences for the generated protein sequences.

6.2 Structure Quality

To evaluate the designability and novelty of the generated protein structures, we computed the scTM and Training TM scores, respectively. The distribution of the scTM scores of the generated structures is shown in Fig. 9a, and the distribution of the training TM scores is shown in Fig. 9b. Fig. 8 shows some of the generated protein structures alongside samples from the dataset.

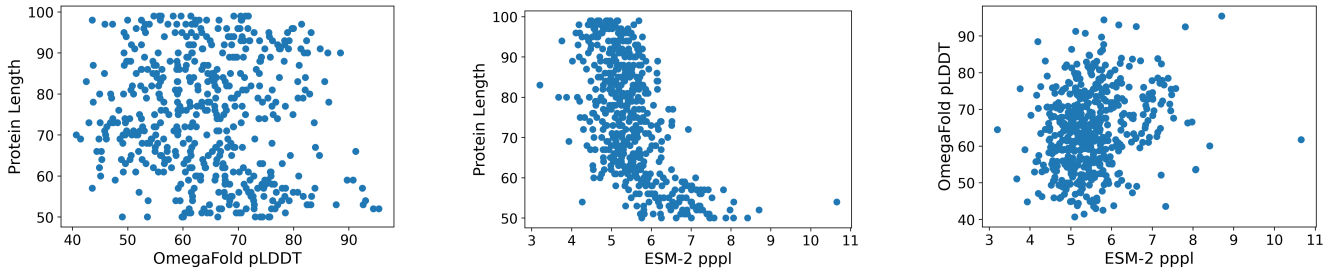
From Fig. 9a we can see that none of the protein structures generated using Dzyn have an scTM score greater than 0.5, suggesting that they are not designable. Protein structure generation models like ProtDiff [10] and LatentDiff [2] that also utilize EGNNs, report that 17.1% and 64.7% of the structures generated by them were considered designable, respectively. A possible reason for this difference in performance could be that both ProtDiff and LatentDiff only model the C_α atom. Whereas, Dzyn models the C_α, C, and N atoms and the amino acid type, which greatly increases the complexity of the task. To gain a comprehensive understanding of the strengths and limitations of EGNNs in designing protein backbones, which include the C_α, C, and N atoms, further experimentation is essential. Two primary avenues warrant exploration. Firstly, adapting the graph representation to incorporate distinct nodes for each of these atoms and assessing the potential enhancement in designability. Secondly, examining whether EGNNs are more effective in designing particular protein regions as opposed to the entire structure holds promise. This entails training EGNNs on datasets tailored to focus on designing specific functional or structural elements of proteins, offering valuable insights. The findings from these experiments can inform the refinement of protein co-design methodologies utilizing EGNNs as denoising neural networks.

The protein structures generated using Dzyn, as depicted in Fig. 8, exhibit similarities to protein structures from the dataset, especially in the presence of helix segments. However, despite these similarities, some discrepancies are evident in the Dzyn generated structures. These include deviations in overall shape and slight deformations within the helices themselves. Notably, while the generated structures contain helix segments, they may not be as well-formed or evenly spaced as those in the dataset. This observation aligns with the scTM scores of the generated structures, indicating that while certain features are replicated, the overall designability and plausibility of the generated protein structures is limited.

Based on the distribution shown in Fig. 9b, we observe that none of the training TM scores approach one, suggesting that the protein structures generated using Dzyn are novel, and the model is not memorizing the training set. However, despite the novelty of the generated protein structures, their lack of designability indicates that Dzyn cannot be successfully used to generate quality

Table 3
A comparison of Dzyn to various protein sequence generation models using the average pLDDT and the average ESM-2 pppl scores over the generated sequences. 'Source Data' refers to 500 sequences sampled from the training set, while 'Random' signifies sequences sampled from a uniform distribution over the amino acids. Our results are shaded in gray, while the best results are highlighted in bold.

Model	Average pLDDT (↑)	Structure Prediction Model	Average ESM-2 pppl (↓)
ESM-1b (Rao et al. [46], 2021)	58.0	OmegaFold	-
proteinGAN (Repecka et al. [13], 2021)	30.4	ESMFold	16.48
SeqDesign (Shin et al. [47], 2021)	43.1	ESMFold	11.89
ESM-2 (Lin et al. [34], 2023)	50.7	OmegaFold	-
EvoDiff-OADM (Alamdari et al. [26], 2023)	44.4	OmegaFold	15.77
nanoGPT (Karpathy [48], 2023)	61.0	ESMFold	8.18
DiMA(Meshchaninov et al. [16], 2024)	**80.8**	**ESMFold**	**5.20**
Dzyn (Ours, Final, T = 1000)	64.4	OmegaFold	**5.51**
Dzyn (Ours, Baseline, T = 100)	42.9	OmegaFold	16.38
Source Data	71.2	OmegaFold	5.20
Random	32.0	OmegaFold	20.14



(a) **(b)** **(c)**

Figure 7. The results from the experiments to evaluate the foldability and similarity to natural sequences for the generated sequences. (a) A scatter plot of the OmegaFold pLDDT scores vs the length of the generated sequences. (b) A scatter plot of the ESM-2 pppl scores vs the length of the generated sequences. (c) A scatter plot of the ESM-2 pppl scores vs OmegaFold pLDDT scores of the generated sequences.

protein structures.

6.3 Co-Design Correlation

The ccTM scores and AAC scores were used to evaluate the correlation between the generated protein sequences and structures. The distributions of the ccTM scores and AAC scores over the generated proteins are shown in Figures 10a and 10b, respectively.

We found that the average ccTM score over the set of generated proteins was 0.17, and none of the generated proteins had a ccTM score greater than 0.5 (See Fig. 10a). We also found that the average AAC score was 4.78% (See Fig. 10b). For protein structures and sequences that are correlated, we expect the AAC scores to be much higher and the ccTM scores to be greater than 0.5. These results suggest that the generated protein structures and sequences do not correlate with each other. However, this was expected because the generated protein structures are not designable and, therefore, do not have a sequence that folds into them. Due to the lack of designability of the protein structures, we cannot draw meaningful conclusions on the correlation between the generated structures and sequences.

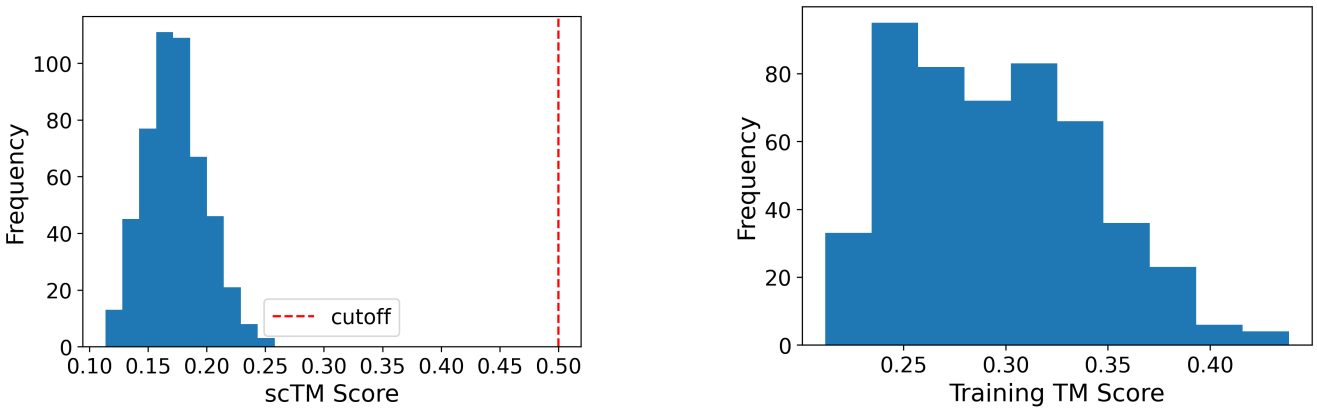
7 Conclusion

In conclusion, our study has introduced Dzyn, a novel diffusion model tailored for protein structure and sequence co-design, leveraging EGNNs as the denoising neural network. Additionally, we have proposed a modification to the EGNN architecture, enhancing its performance as demonstrated through an ablation study. Through rigorous experimentation and evaluation, we have assessed the quality of both generated sequences and structures, as well as the correlation between them.

To address the key question of how effectively graph-based diffusion models facilitate the co-design of protein structures and sequences, we designed experiments that evaluate the quality of the generated sequences and structures, and the correlation between them. Our analysis highlights Dzyn's ability to generate high-quality protein sequences that are both highly foldable and closely resemble natural sequences. We also find that the quality of the sequence generated by Dzyn is second only to the current SOTA model, DiMA, which was trained exclusively on protein sequence generation, a much simpler task. However, despite the visual similarities between the protein structures generated by Dzyn and those in the dataset, we observed that these Dzyn generated structures were not designable. Furthermore, our findings indicate that the generated structures exhibit novelty, indicating that the model does not merely memorize the training set but rather explores diverse solutions within the design space. Despite this, we have observed a lack of correlation between the generated sequences and structures, primarily due to issues with the quality of the generated structures. While our findings cannot definitively conclude whether current graph-based diffusion mod-

(a) (b)

Figure 8. Qualitative comparison of a few selected protein structures sourced from the dataset and those generated using Dzyn. Variations in scale are observed due to the differing lengths and spatial requirements of the protein structures. (a) Protein structures sourced from the dataset. (b) Protein structures generated using Dzyn.



(a) (b)

Figure 9. The results from the experiments to evaluate the designability and novelty of the generated structures. (a) A histogram showing the distribution of the scTM scores for the generated structures. (b) A histogram showing the distribution of the TrainingTM scores for the generated structures.

els are suitable for co-design, Dzyn's success in generating high-quality sequences warrants further exploration to unlock their full potential in this domain.

For future endeavors, it is crucial to address the limitations encountered by EGNNs in designing protein structures, particularly their challenge in accurately modeling multiple atoms (C_α, C, and N atoms) within a single node. To address this limitation, we recommend exploring three approaches. First, we suggest representing the C_α, C, and N atoms as individual nodes, simplifying the representation for the neural network by reducing the dimensionality of the coordinate features from 9 to 3 per node. However, this approach comes at the cost of tripling graph size and computation time, making it harder to scale for larger proteins. Utilizing GNN architectures like GraphSage [49], designed for large graphs, might be advantageous in this case. Second, we propose simplifying the task by training the network to design only a specific part of the protein while conditioning on the remaining structure, similar to the work of Luo et al. [19]. This approach could provide valuable insights into whether EGNNs can be utilized for a smaller sub-problem of the co-design task. Third, we suggest exploring alternative neural network architectures, particularly transformers. This approach would utilize the coordinates of the C_α, C, and N atoms. This stands in contrast to most current co-design methods using transformers, which focus utilize the amino acid orientation instead. By incorporating atomic coordinates, we can not only gain valuable insights into protein structure design with transformers but also discern if the C_α, C, and N coordinate representation itself is a limiting factor. The success of transformers in other tasks using different representations suggests this is a worthwhile avenue for exploration.

In summary, our study explores a novel approach for protein co-design, investigating the effectiveness of both a graph representation and EGNNs as a denoising neural network. While approach

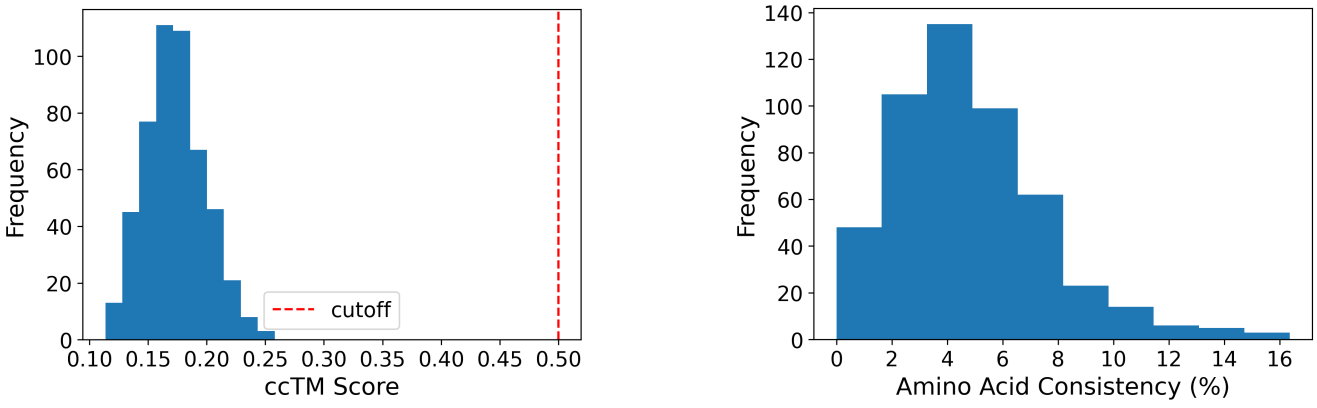


Figure 10. The results from the experiments to evaluate the correlation between the generated structures and sequences. (a) A histogram showing the distribution of the ccTM scores for the generated proteins. (b) A histogram showing the distribution of the AAC scores for the generated proteins.

successfully generates protein sequences, challenges remain in structure generation and sequence-structure correlation. We further establish a robust evaluation pipeline for co-design models and introduce two new metrics specifically designed to evaluate the critical, yet previously unexplored, aspect of correlation between generated structures and sequences. Our work paves the way for promising future advancements in protein co-design, but also highlights the need for further exploration to fully unlock the potential of this approach.

References

[1] Po-Ssu Huang, Scott E Boyken, and David Baker. "The coming of age of de novo protein design". In: *Nature* 537.7620 (2016), pp. 320–327.

[2] Cong Fu et al. "A latent diffusion model for protein structure generation". In: *The Second Learning on Graphs Conference*. 2023.

[3] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models". In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.

[4] Namrata Anand, Raphael Eguchi, and Po-Ssu Huang. "Fully differentiable full-atom protein backbone generation". In: (2019).

[5] Sari Sabban and Mikhail Markovsky. "RamaNet: Computational de novo helical protein backbone design using a long short-term memory generative adversarial neural network". In: *BioRxiv* (2019), p. 671552.

[6] Raphael R Eguchi, Christian A Choe, and Po-Ssu Huang. "Ig-VAE: Generative modeling of protein structure by direct 3D coordinate generation". In: *PLoS computational biology* 18.6 (2022), e1010271.

[7] Kevin E. Wu et al. *Protein structure generation via folding diffusion*. 2022. arXiv: 2209.15611 [q-bio.BM].

[8] Joseph L Watson et al. "De novo design of protein structure and function with RFdiffusion". In: *Nature* 620.7976 (2023), pp. 1089–1100.

[9] Jason Yim et al. "Se (3) diffusion model with application to protein backbone generation". In: *arXiv preprint arXiv:2302.02277* (2023).

[10] Brian L Trippe et al. "Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem". In: *arXiv preprint arXiv:2206.04119* (2022).

[11] Alex Hawkins-Hooker et al. "Generating functional protein variants with variational autoencoders". In: *PLoS computational biology* 17.2 (2021), e1008736.

[12] Emre Sevgen et al. "ProT-VAE: protein transformer variational autoencoder for functional protein design". In: *bioRxiv* (2023), pp. 2023–01.

[13] Donatas Repecka et al. "Expanding functional protein sequence spaces using generative adversarial networks". In: *Nature Machine Intelligence* 3.4 (2021), pp. 324–333.

[14] Ali Madani et al. "Progen: Language modeling for protein generation". In: *arXiv preprint arXiv:2004.03497* (2020).

[15] Noelia Ferruz, Steffen Schmidt, and Birte Höcker. "ProtGPT2 is a deep unsupervised language model for protein design". In: *Nature communications* 13.1 (2022), p. 4348.

[16] Viacheslav Meshchaninov et al. "Diffusion on language model embeddings for protein sequence generation". In: *arXiv preprint arXiv:2403.03726* (2024).

[17] Wengong Jin et al. "Iterative refinement graph neural network for antibody sequence-structure co-design". In: *arXiv preprint arXiv:2110.04624* (2021).

[18] Ivan Anishchenko et al. "De novo protein design by deep network hallucination". In: *Nature* 600.7889 (2021), pp. 547–552.

[19] Shitong Luo et al. "Antigen-specific antibody design and optimization with diffusion-based generative models for protein structures". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 9754–9767.

[20] Jue Wang et al. "Deep learning methods for designing proteins scaffolding functional sites". In: *BioRxiv* (2021), pp. 2021–11.

[21] Chence Shi et al. "Protein sequence and structure co-design with equivariant translation". In: *arXiv preprint arXiv:2210.08761* (2022).

[22] Namrata Anand and Tudor Achim. *Protein Structure and Sequence Generation with Equivariant Denoising Diffusion Probabilistic Models*. 2022. arXiv: 2205.15019 [q-bio.QM].

[23] Justas Dauparas et al. "Robust deep learning–based protein sequence design using ProteinMPNN". In: *Science* 378.6615 (2022), pp. 49–56.

[24] John Jumper et al. "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596.7873 (2021), pp. 583–589.

[25] Minkyung Baek et al. "Accurate prediction of protein structures and interactions using a three-track neural network". In: *Science* 373.6557 (2021), pp. 871–876.

[26] Sarah Alamdari et al. "Protein generation with evolutionary diffusion: sequence is all you need". In: *bioRxiv* (2023), pp. 2023–09.

[27] Zhangyang Gao et al. "DiffSDS: A geometric sequence diffusion model for protein backbone inpainting". In: ().

[28] Sitao Zhang et al. "PRO-LDM: Protein Sequence Generation with a Conditional Latent Diffusion Model". In: *bioRxiv* (2023), pp. 2023–08.

[29] Thomas N Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907* (2016).

[30] Emiel Hoogeboom et al. *Equivariant Diffusion for Molecule Generation in 3D*. 2022. arXiv: 2203.17003 [cs.LG].

[31] Mengchun Zhang et al. “A survey on graph diffusion models: Generative ai in science for molecule, protein and material”. In: *arXiv preprint arXiv:2304.01565* (2023).

[32] Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. “E (n) equivariant graph neural networks”. In: *International conference on machine learning*. PMLR. 2021, pp. 9323–9332.

[33] Rebecca F Alford et al. “The Rosetta all-atom energy function for macromolecular modeling and design”. In: *Journal of chemical theory and computation* 13.6 (2017), pp. 3031–3048.

[34] Zeming Lin et al. “Evolutionary-scale prediction of atomic-level protein structure with a language model”. In: *Science* 379.6637 (2023), pp. 1123–1130.

[35] RA Engh and R Huber. “Structure quality and target parameters”. In: (2012).

[36] Zhiye Guo et al. “Diffusion models in bioinformatics: A new wave of deep learning revolution in action”. In: *arXiv preprint arXiv:2302.10907* (2023).

[37] Jianyi Yang et al. “Improved protein structure prediction using predicted interresidue orientations”. In: *Proceedings of the National Academy of Sciences* 117.3 (2020), pp. 1496–1503.

[38] Emiel Hoogeboom et al. “Argmax flows and multinomial diffusion: Learning categorical distributions”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 12454–12465.

[39] Diederik Kingma et al. “Variational diffusion models”. In: *Advances in neural information processing systems* 34 (2021), pp. 21696–21707.

[40] “UniProt: the universal protein knowledgebase in 2023”. In: *Nucleic acids research* 51.D1 (2023), pp. D523–D531.

[41] Ruidong Wu et al. “High-resolution de novo structure prediction from primary sequence”. In: *BioRxiv* (2022), pp. 2022–07.

[42] Zeming Lin et al. “Language models of protein sequences at the scale of evolution enable accurate structure prediction”. In: *BioRxiv* 2022 (2022), p. 500902.

[43] *Benchmark in Machine Learning Methods for Protein Folding*. Accessed on 20-July-2023. URL: https://310.ai/2023/05/17/benchmarking-machine-learning-methodsfor-protein-folding-a-comparative-study-of-esmfold-omegafold-and-alphafold/ (visited on 07/20/2023).

[44] OG Righetti. *Protein Structure. A Practical Approach. TE Creighton*. 1989.

[45] Y Zhang and J Skolnick Tm-Align. “A protein structure alignment algorithm based on the TM-score., 2005, 33”. In: *DOI: https://doi. org/10.1093/nar/gki524. PMID: https://www. ncbi. nlm. nih. gov/pubmed/15849316* (), pp. 2302–2309.

[46] Roshan M Rao et al. “MSA transformer”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8844–8856.

[47] Jung-Eun Shin et al. “Protein design and variant prediction using autoregressive generative models”. In: *Nature communications* 12.1 (2021), p. 2403.

[48] Andrej Karpathy. *nanoGPT*. https://github.com/karpathy/nanoGPT. 2023.

[49] Will Hamilton, Zhitao Ying, and Jure Leskovec. “Inductive representation learning on large graphs”. In: *Advances in neural information processing systems* 30 (2017).