Antibody Complementarity Determining Regions (CDRs) design using Constrained Energy Model

Tianfan Fu
Georgia Institute of Technology
Atlanta, GA, USA

Jimeng Sun
University of Illinois Urbana-Champaign
Urbana, IL, USA

ABSTRACT

In recent years, therapeutic antibodies have become one of the fastest-growing classes of drugs and have been approved for the treatment of a wide range of indications, from cancer to autoimmune diseases. Complementarity-determining regions (CDRs) are part of the variable chains in antibodies and determine specific antibody-antigen binding. Some explorations use *in silico* methods to design antibody CDR loops. However, the existing methods faced the challenges of maintaining the specific geometry shape of the CDR loops. This paper proposes a *Constrained Energy Model* (`CEM`) to address this issue. Specifically, we design a constrained manifold to characterize the geometry constraints of the CDR loops. Then we design the energy model in the constrained manifold and only depict the energy landscape of the manifold instead of the whole space in the vanilla energy model. The geometry shape of the generated CDR loops is automatically preserved. Theoretical analysis shows that learning on the constrained manifold requires less sample complexity than the unconstrained method. `CEM`'s superiority is validated via thorough empirical studies, achieving consistent and significant improvement with up to 33.4% relative reduction in terms of 3D geometry error (Root Mean Square Deviation, RMSD) and 8.4% relative reduction in terms of amino acid sequence metric (perplexity) compared to the best baseline method. The code is publicly available at https://github.com/futianfan/energy_model4antibody_design.

CCS CONCEPTS

• **Computing methodologies** → *Continuous space search.*

KEYWORDS

antibody design, protein design, energy model, deep generative model, drug discovery

ACM Reference Format:
Tianfan Fu and Jimeng Sun. 2022. Antibody Complementarity Determining Regions (CDRs) design using Constrained Energy Model. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, Washington DC, USA, 11 pages. https://doi.org/10.1145/3534678.3539285

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539285

1 INTRODUCTION

Antibody engineering has dramatically evolved since muromonab-CD3 (treating acute transplant rejection), the first therapeutic antibody, was approved for medical use by the United States Food and Drug Administration (US FDA) in 1986 [29]. As of December 2019, 79 therapeutic antibodies have been approved by the US FDA with significant growth potential. Compared with small-molecule drugs, the primary advantage of antibody drugs is their fewer adverse effects due to their high specificity to the antigen. Therapeutic antibodies have become the predominant class of new drugs developed in recent years for treating various human diseases, including many cancers, autoimmune, metabolic, and infectious diseases. Over the past five years, therapeutic antibodies have become the best-selling drugs[1] in the pharmaceutical market [33]. The global therapeutic antibody market was valued at approximately $144 billion in 2020 and is expected to reach $300 billion by the end of 2025 [15].

In terms of design methodology, most of the affinity and specificity of antibodies is modulated by a set of binding loops called the Complementarity Determining Regions (CDRs) found on the variable domain of antibody. There is a high demand to develop *in silico* methods for antibody design, especially CDR loop design [33]. However, current antibody discovery and development procedures heavily rely on a mixture of high throughput screening and experimental heuristics, which can be costly and laborious.

Recently machine learning methods have been proposed in designing novel antibodies [1, 21, 26, 42]. Most of these methods formulate the antibody design problem as either amino acid sequence design or 3D graph design tasks. However, several challenges remain: (1) Antibody CDR loops have specific geometry shape [35, 48, 55]. However, most of the existing antibody design methods do not consider it, which may lead to the generation of invalid CDR loops. (2) Most of the existing deep generative models do not leverage the external knowledge and are purely learning from data, impeding their ability to incorporate constraints. To address these issues, in this paper, we proposed Constrained Energy Model that takes geometry constraints into account during the generation of 3D CDR loops. We summarize the main contributions of the paper as follows. (1) We formulate antibody CDR design as a constrained 3D generation task and define a *constrained manifold* to represent all the geometric valid CDR loops (Sec 3.1). (2) We design a *Constrained Energy Model* that learns the 3D structure on the defined manifold (Sec 3.3). (3) Theoretical analysis shows the sample sizes required for constrained learning on the manifold is about two-thirds of the sample sizes required for unconstrained learning (Sec 3.4). (4) Experimental results confirm the effectiveness

[1]The top-selling drug worldwide is Humira (antibody), which is used to treat arthritis, plaque psoriasis, ankylosing spondylitis, Crohn's disease, and ulcerative colitis.

of the proposed method, which obtains up to 33.4% relative reduction in 3D geometry error (Root Mean Square Deviation, RMSD) and 8.4% relative improvement in terms of amino acid sequence metric (perplexity) (Sec 4).

2 RELATED WORK

2.1 Deep Generative Model

The basic idea of deep generative models (DGMs) is to learn the data likelihood with deep learning models. We briefly review several commonly used DGMs, including Variational Auto-Encoder (VAE), Generative Adversarial Network (GAN), flow model (a.k.a. normalizing flow), auto-regressive model (AR), energy model (a.k.a. energy based model, EBM) [9]. Also, we analyze the benefit of the energy model and why we choose the energy model for 3D antibody CDR loop design in this paper.

Specifically, Variational Auto-Encoder (VAE) [24] estimates a lower bound of data distribution (i.e., evidence lower bound (ELBO) in variational inference) instead of directly optimizing data distribution. Generative Adversarial Network (GAN) [16] circumvents the optimization of the likelihood directly; instead, it separates the real data from generated data and recast the generation problem into a binary classification problem. Flow model (or normalizing flow model) can reconstruct input data exactly by designing an invertible and deterministic mapping. Thus, the exact likelihood is optimized directly (instead of optimizing a surrogate, e.g., VAE) [41, 46, 54]. Auto-regressive (AR) model generates variables sequentially, i.e., generating the current variable based on all the previous variables. For example, [30] designed an autoregressive 3D graph neural network to design small-molecule conformation conditioned on the pocket conformation of the target protein. It generates a single variable (an atom) within a single step based on the conditional distribution. However, most of these deep generative models fail to preserve the intrinsic property of the underlying graph, e.g., translation- and rotation-invariance for 3D graph or permutation invariance for 2D graph. It would yield different likelihoods for different formats of the same underlying graph and impede the learning efficiency [27, 43].

On the other hand, the energy model (a.k.a. energy based model, EBM) defines an unnormalized probability distribution over the data and assigns lower energies to the data points close to the real data than others data points [6, 13, 17, 27]. The energy models bypass the need to reconstruct 3D graph structure explicitly. Because the energy measure is scalar and differentiable with respect to the data points, energy models can back-propagate the gradient to update the data points directly. The basic idea is similar to the differentiable learning in the context of molecular optimization [13]. In the context of 3D graphs, we select a translation- and rotation-invariant graph neural network as the parameterized energy function to represent the data. As a result, the translation- and rotation-invariances are naturally preserved (will be discussed in Sec 3.3).

In addition, most of the existing DGMs are learning purely from data and are hard to incorporate constraints. To address this issue, we propose a Constrained Energy Model (`CEM`). Different from the vanilla (unconstrained) energy model, `CEM` enables learning energy model while maintaining the constraints at the same time.

2.2 Protein/Antibody Design

There are two fundamental tasks in protein/antibody design. (a) One is graph conditioned protein design, whose objective is to design an amino acid sequence that can fold into the input 3D structure. (b) Another is *de novo* protein/antibody design, whose objective is to design protein (its amino acid sequence or/and the corresponding 3D structure) from scratch. Specifically, for (a) graph conditioned protein design, conditioned on the input 3D graph structure (the backbone), the goal is to recover the amino acid sequence. Most of the existing methods are based on deep generative models (DGMs) by learning a mapping from a 3D graph structure to an amino acid and generating a single amino acid at a time. Many kinds of neural network models were leveraged/designed to learn the mapping, e.g., structured transformer [20], three-dimensional convolutional neural network (3DCNN) [37, 56], graph convolutional network (GCN) [49], joint sequence-folding embedding model [5]. For (b) *de novo* protein/antibody design are usually cast into a sequence generation problem or 3D graph generation problem. Existing methods include variational autoencoder (VAE) based methods [7, 47] and generative adversarial network (GAN) based methods [23, 40]. For antibody design, [36] combines CDR canonical structures and iteratively redesign their positions; [26] designed a neural network ensemble to backpropagate the gradient to update the amino acid sequence in continuous space; [1, 42] leverage Long Short Term Memory (LSTM) to generate amino acid sequences; [21] proposed an *iterative refinement graph neural network* to jointly design 3D graph structures and amino acid sequences. However, almost all of these methods learn directly from data and omit the geometry constraints in the generation process.

3 METHOD

Overview. Firstly, Sec 3.1 formulates the problem of antibody CDR loop design. In particular, we define the constrained manifold $\mathcal{M}$ to characterize the geometry constraints. Next, Sec 3.2 describes the vanilla (unconstrained) energy model as background. Then we elaborate the Constrained Energy Model (`CEM`) in Sec 3.3, i.e., learning energy model on the constrained manifold $\mathcal{M}$, including the formulation, neural architecture of energy function, learning, and inference procedure. Finally, Sec 3.4 analyzes the theoretical properties of the proposed method and finds that learning on the constrained manifold takes less sample complexity than an unconstrained model. We list the mathematical notations in Table 1.

3.1 Problem Formulation

This section introduces the *de novo* antibody CDR loop generation problem. We start with definitions of some basic data structures.

Definition 1 (Amino acid)**.** Amino acids are the basic building blocks of proteins. Amino acids are small organic molecules that consist of an α (central) carbon atom linked to an amino group, a carboxyl group, a hydrogen atom, and a variable component called a side chain. In proteins such as enzymes antibodies, the long chain of amino acids is linked together by peptide bonds and is folded into a three-dimensional functional shape or tertiary structure. Following [20], we take the coordinate of α atom as the coordinate of the amino acid. The set of amino acids is denoted $\mathcal{V}$, which contains 20 natural amino acids. Thus, we have $|\mathcal{V}| = 20$. Frequent

Table 1: Mathematical notations and explanations.

Notations	Explanations
$\mathcal{V}$	vocabulary set of all the amino acids.
$\mathbf{a} \in R^{\lvert\mathcal{V}\rvert}$	categorical distribution of an amino acid, Definition 1
$\mathbf{a}_i \in R^{\lvert\mathcal{V}\rvert}$	$\mathbf{a} = \text{softmax}(\widehat{\mathbf{a}})$ (Eq. 1) or $\mathbf{a}$ is one-hot vector categorical distribution of i-th amino acid
$(\mathbf{a})_j$	j-th element in $\mathbf{a}$
$\mathcal{A}$	set of amino acids' vectors, $\mathcal{A} = [\mathbf{a}_1, \cdots, \mathbf{a}_N]$
N	number of amino acids in CDR loop.
$\mathbf{x}_i \in R^3$	coordinate of i-th amino acid
$\mathcal{X}$	set of coordinates, $\mathcal{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N]$
$\mathcal{Y}$	CDR loop, $\mathcal{Y} = (\mathcal{A}, \mathcal{X})$
$\kappa, \epsilon_1, \epsilon_2$	hyperparameter of constraints, Eq. (3), (4)
$\mathcal{M}$	the constrained manifold, Definition 4
E_θ	energy function, Eq. 9
P_θ	probability distribution of energy model, Eq. 9
P_{data}	real data distribution, Eq. 7
$\mathcal{Y}^{(+)}$	positive samples (real CDR loop).
$\mathcal{Y}^{(-)}$	hallucinated samples (a.k.a. negative samples).
$\lVert z \rVert$	l_2 norm of the vector z
$\mathbf{B} \in R^{\lvert\mathcal{V}\rvert \times d}$	embedding matrix of all the amino acids
d	hidden dimension of EGNN.
L	Number of layers in EGNN.
$\mathbf{v}_i^{(l)}$	message vector of node i at l-th layer
$\mathbf{w}_{ij}^{(l)}$	message vector of edge from i to j at l-th layer
$\mathbf{h}_i^{(l)}$	i-th node's embedding at l-th layer
$\mathbf{x}_i^{(l)}$	i-th node's position embedding at l-th layer
$\phi_e / \phi_h / \phi_x$	MLPs in EGNN
$\mathcal{R}_{\mathcal{M}}(\mathcal{X})$	retraction operation, Definition 5

amino acids contain histidine, isoleucine, leucine, lysine, etc. To represent the amino acid's categorical distribution (all elements are non-negative, sum of all elements are 1), we use a $|\mathcal{V}|$-dimensional vector $\mathbf{a} \in R^{|\mathcal{V}|}$. Positive and hallucinated samples have different amino acid representation. Positive and hallucinated samples would be elaborated in Sec 3.2.

(A) amino acid in positive samples (fixed): we directly let $\mathbf{a}$ to be a one-hot vector to represent the category of amino acid.

(B) amino acid in hallucinated samples (will be updated): to guarantee it is a valid categorical distribution, i.e., the sum of all the elements equals to 1, we use another vector $\widehat{\mathbf{a}} \in R^{|\mathcal{V}|}$, normalize $\widehat{\mathbf{a}}$ using softmax function,

$$\mathbf{a} = \text{softmax}(\widehat{\mathbf{a}}), \qquad \mathbf{a} \in R_+^{|\mathcal{V}|}, \tag{1}$$

where the j-th element of $\mathbf{a}$ is $(\mathbf{a})_j = \exp(\widehat{\mathbf{a}})_j / (\sum_{j'=1}^{|\mathcal{V}|} \exp(\widehat{\mathbf{a}})_{j'})$. When updating hallucinated samples, we back-propagate gradient from neural network to update $\widehat{\mathbf{a}}$ (since $\mathbf{a}$ is differentiable w.r.t. $\widehat{\mathbf{a}}$).

Definition 2 (Protein/Antibody)**.** Proteins consist of one or more chains of amino acids called *polypeptides* [14, 33]. The sequences of the amino acid chain cause the polypeptide and are folded into a three-dimensional (3D) functional shape, a.k.a. tertiary structure. An antibody is a special kind of protein that is symmetric and Y-shaped. As illustrated in Fig 1, each half of the symmetric unit has two chains: a heavy chain (H) and a light (L) chain.

The majority of the affinity and specificity of antibodies is modulated by a set of binding loops called the Complementarity Determining Regions (CDRs) found on the variable domain of each of the two chains. CDR is complicated to model computationally, causing a significant hurdle for *in silico* development of antibody biotherapeutics.

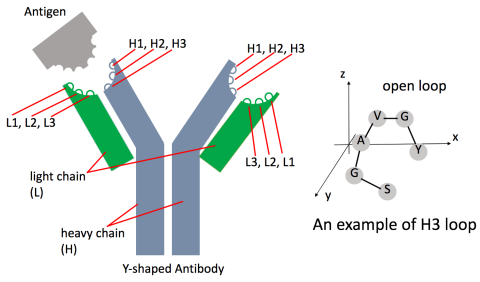

Figure 1: Data representation. An antibody is a special kind of protein with a symmetric Y shape, each half of the symmetric unit has two chains: a heavy chain (H) and a light chain (L). In total, there are four chains, two identical H/L chains. The majority of the binding affinity (to specific antigen) is modulated by a set of binding loops called the Complementarity Determining Regions (CDRs) found on the variable domain of each of the H and L chains. There are 6 CDR loops on each half of the antibody, L1, L2, L3 on the light chain, and H1, H2, H3 on the heavy chain. We show an example of the H3 loop of the antibody with protein data bank (PDB) ID 5iwl. The CDR loop in the 3D geometry graph contains six amino acids (SGAVGY) and their 3D coordinates.

Definition 3 (CDR loops)**.** In the Y-shaped antibody, there are six CDR loops, L1, L2, L3 loops on the light chain and H1, H2, H3 loops on the heavy chain [31, 45]. A 3D CDR loop (H1, H2, or H3) can be characterized by a sequence of amino acids and their 3D coordinates. A CDR loop is denoted $\mathcal{Y}$, suppose it has N amino acids, it is represented as

$$\mathcal{Y} = (\mathcal{A}, \mathcal{X}), \quad \mathcal{A} = [\mathbf{a}_1, \cdots, \mathbf{a}_N], \ \mathcal{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N]. \tag{2}$$

As described in Definition 1, $\mathbf{a}_i$ represents the i-th amino acid' categorical distribution, $\mathbf{x}_1, \cdots, \mathbf{x}_N \in R^3$ represent the amino acids' 3D coordinates. Following [20], we take the coordinate of α atom as the coordinate of the amino acid. An illustration of data representation is given in Fig 1.

Compared with L1, L2, L3 loops, the H1, H2, H3 loop in the CDR of an antibody plays a critical role in its binding ability to potential antigens [34, 39]. Therefore, this paper restricts our attention to designing H1, H2, and H3 loops separately.

Validity constraints. We define the validity of the generated loop based on empirical domain knowledge about CDR loops [35, 48, 55]. Specifically, we define the validity of a generated 3D CDR loop when it satisfies the following two constraints:

(1) Peptide bond length. Multiple amino acids are linked together by peptide bonds and form a single chain. The length of peptide bonds is relatively fixed, i.e., the distance between connected amino acids is a constant [55], ($||\cdot||_2$ represents l_2 norm of vector)

$$||\mathbf{x}_i - \mathbf{x}_{i+1}||_2 = \kappa, \ \text{for } i = 1, \cdots, N-1. \tag{3}$$

(2) Open loop. The shape of CDR is an open loop, as shown in Fig 1, where the distance between the first and the last amino acids (coordinates are $\mathbf{x}_1$ and $\mathbf{x}_N$ respectively) is within a specific range [35, 48],

$$\epsilon_1 \le ||\mathbf{x}_1 - \mathbf{x}_N||_2 \le \epsilon_2. \tag{4}$$

The setup of $\kappa, \epsilon_1, \epsilon_2$ is based on domain knowledge [35, 48] and empirical validation, e.g., for H1 loop, $\kappa = 3.80$, $\epsilon_1 = 11.4$, $\epsilon_2 = 13.1$; for H2 loop, $\kappa = 3.81$, $\epsilon_1 = 5.0$, $\epsilon_2 = 5.9$; for H3, $\kappa = 3.81$, $\epsilon_1 = 6.50$, $\epsilon_2 = 8.50$. The units of $\kappa, \epsilon_1, \epsilon_2$ are Angstrom $\mathring{A}$ (10^{-10} m).

Definition 4 (Constrained Manifold $\mathcal{M}$)**.** We define the constrained manifold $\mathcal{M}$ to represent the set of CDR loops that satisfy the constraints defined in Eq.3 and 4. Formally, $\mathcal{M}$ is defined as

$$\mathcal{M} = \big\{(\mathcal{A}, \mathcal{X}) \mid \epsilon_1 \le ||\mathbf{x}_1 - \mathbf{x}_N||_2 \le \epsilon_2, \& \\ ||\mathbf{x}_1 - \mathbf{x}_2||_2 = ||\mathbf{x}_2 - \mathbf{x}_3||_2 = \cdots = ||\mathbf{x}_{N-1} - \mathbf{x}_N||_2 = \kappa\big\}. \tag{5}$$

Problem 1 (constrained *de novo* antibody CDR loop design)**.** Constrained *de novo* antibody CDR loop design aims to generate novel 3D CDR loops $\mathcal{Y}$s within the constrained manifold $\mathcal{M}$ from scratch, i.e., $\mathcal{Y} \in \mathcal{M}$.

3.2 Background: (unconstrained) energy model

This section briefly introduces the vanilla (unconstrained) energy model as background [6, 13, 17, 27]. Energy model (a.k.a. energy based model, EBM) defines a parameterized probability distribution P_θ over all the data points (CDR loops $\mathcal{Y}$ here),

$$P_\theta(\mathcal{Y}) = \frac{e^{-E_\theta(\mathcal{Y})}}{Z(\theta)}, \quad Z(\theta) = \int e^{-E_\theta(\mathcal{Y})} d\mathcal{Y}, \tag{6}$$

where θ are the learnable parameters, energy function $E_\theta(\mathcal{Y})$ is usually a neural network whose output is a scalar. $Z(\theta)$ is normalization constant, a.k.a. partition function. Due to the high dimension, cardinality and complexity of E_θ, $Z(\theta)$ is usually computationally intractable. Intuitively, an ideal energy function assigns lower energy E_θ (corresponds to higher probability/likelihood P_θ) to data points that are close to real CDR loops and higher energies to other data points.

Compared with other deep generative models in Sec 2, energy models can bypass the need to reconstruct 3D graph structure explicitly. Because the energy measure is a scalar, and is differentiable with respect to the data points, energy models can back-propagate the gradient to update the data points directly. In the context of 3D graphs, we select a translation-, rotation- invariant graph neural network as the parameterized energy function to represent the data (3D graph). As a result, the translation-, rotation- invariances are naturally preserved (will be discussed in Sec 3.3). When generating graphs, we start from scratch and back-propagate the gradient of the energy function to update the 3D graph structure iteratively.

To learn the energy model with maximum likelihood learning, we will maximize log likelihood (Eq. 12) on real data distribution,

$$\arg\max_\theta \mathcal{L}(\theta), \ \mathcal{L}(\theta) = \frac{1}{|\mathcal{D}|} \sum\nolimits_{\mathcal{Y}^{(+)} \in \mathcal{D}} \log P_\theta(\mathcal{Y}^{(+)}) \\ = \mathcal{E}_{\mathcal{Y}^{(+)} \sim P_{\text{data}}}[-E_\theta(\mathcal{Y}^{(+)})] - \log Z(\theta), \tag{7}$$

where $\mathcal{D}$ denotes the training set; $\mathcal{E}$ denotes expectation; P_{data} is the distribution of real data points, the training data $\mathcal{Y}^{(+)}$ can be seen as i.i.d. samples drawn from P_{data}. As mentioned above, an ideal energy function assigns lower energy to real data and higher energy to the other data points. However, maximum likelihood learning only encourages the lower energies to real data and is insufficient. To address this issue, contrastive divergence is incorporated to create the *hallucinated samples* (a.k.a. negative samples, denoted $\mathcal{Y}^{(-)}$)

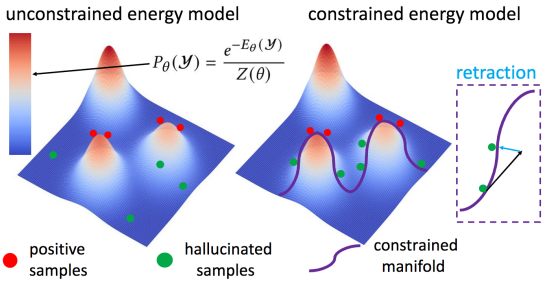

Figure 2: Unconstrained energy model versus constrained energy model. $P_\theta(\mathcal{Y})$ (Eq. 6 or 9) represents the probability/likelihood of energy model. Positive samples (red) have higher probabilities while hallucinated samples (green) have lower probabilities. Different from the unconstrained (vanilla) energy model (Sec 3.2), the constrained energy model defines a constrained manifold (purple) that represents the CDR loops that satisfy constraints on geometry shape. Hallucinated samples are restricted in the manifold so that we have high-quality hallucinated samples and do not have to explore the invalid region (out of the constrained manifold). As shown in the purple dashed box in the right figure, when updating hallucinated samples, we need to project the new samples back to the manifold (purple) using *retraction* (Definition 5), as shown in Eq. (14).

and provide contrastive information [17, 27]. More specifically, the hallucinated samples are drawn from energy-based probability distribution P_θ (Eq.6). The learning objective becomes

$$\mathcal{L}(\theta) = \mathcal{E}_{\mathcal{Y}^{(+)} \sim P_{\text{data}}}\big[-E_\theta(\mathcal{Y}^{(+)})\big] + \mathcal{E}_{\mathcal{Y}^{(-)} \sim P_\theta}\big[E_\theta(\mathcal{Y}^{(-)})\big], \tag{8}$$

where the logarithm of normalization constants $\log Z(\theta)$ are eliminated in the whole objective. In order to sample from $P_\theta(\mathcal{Y})$ defined in Eq.(6), we resort to gradient based Markov Chain Monte Carlo (MCMC) [52]. Traditional MCMC methods leverage random-walk based proposal (e.g., Gaussian). And due to random-walk behavior, it usually suffers from poor efficiency in exploring the state space. To address this issue, we propose to use gradient MCMC by leveraging the geometric information of the target probability to enhance the sampling efficiency.

3.3 Constrained Energy Model (CEM)

This section describes the proposed Constrained Energy Model (CEM), including its formulation, neural architecture of energy function, learning and inference procedures. Fig 2 illustrates the main difference between the unconstrained energy model and Constrained Energy Model, Fig 3 demonstrate CEM's pipeline.

3.3.1 Formulation. Different from an unconstrained energy model, Constrained Energy Model defines a parameterized probability distribution P_θ over all the CDR loops $\mathcal{Y}$ in the constrained manifold $\mathcal{M}$ (Definition 4), $\mathcal{M}$ is the constrained manifold that contains all the geometric valid CDR loops,

$$P_\theta(\mathcal{Y}) = \frac{e^{-E_\theta(\mathcal{Y})}}{Z(\theta)}, \quad \mathcal{Y} \in \mathcal{M}, \quad Z(\theta) = \int_{\mathcal{Y} \in \mathcal{M}} e^{-E_\theta(\mathcal{Y})} d\mathcal{Y}, \tag{9}$$

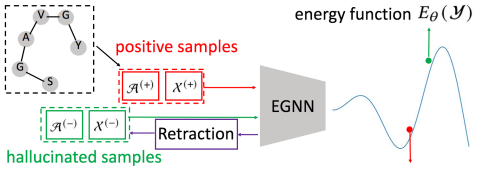

Figure 3: Pipeline of CEM. Both positive and hallucinated samples are restricted to the constrained manifold $\mathcal{M}$ (Definition 4). During the learning procedure, CEM alternatively takes the following two steps: (i) update Constrained Energy Model by maximizing the learning objective $\mathcal{L}(\theta)$ in Eq.(13), where the energy functions of positive samples (red) are pushed down and energy functions of hallucinated samples (green) are pushed up; (ii) update hallucinated samples by sampling from P_θ with *retraction* (purple box) following Eq.(14). Retraction is to project the CDR loop onto the constrained manifold (Definition 5). Then during the inference procedure, we fix the Constrained Energy Model and draw samples from $P_\theta(\mathcal{Y})$ (Eq. 9).

the data ($\mathcal{Y}$) with lower energy E_θ corresponds to higher probability/likelihood P_θ. Normalization constant $Z(\theta)$ is the integral over constrained manifold $\mathcal{M}$, and is still computationally intractable.

3.3.2 Neural architecture of energy function. Then we present the parameterized energy function E_θ. To represent the 3D CDR loop, we leverage the state-of-the-art equivariant graph neural network (EGNN) proposed in [43] so that the translation and rotation on the 3D graph (coordinates) would not change EGNN's output.

Specifically, as mentioned in Definition 3, the input CDR loop features are $\mathcal{Y} = (\mathcal{A}, \mathcal{X})$, $\mathcal{A}$ represent amino acid feature and $\mathcal{X}$ represent 3D coordinates of the amino acids. Suppose $\mathbf{B} \in R^{|\mathcal{V}|\times d}$ is the embedding matrix of all the categories of amino acids in a vocabulary set $\mathcal{V}$, is randomly initialized and learnable, d is the hidden dimension in EGNN. Each kind of amino acid corresponds to a row in $\mathbf{B}$. We suppose there are N amino acids in the CDR loop, each amino acid corresponds to a node in a 3D structure. Node embeddings at the l-th layer are denoted as $\mathbf{H}^{(l)} = \{\mathbf{h}_i^{(l)}\}_{i=1}^N$, where $l = 0, 1, \cdots, L$, L is number of layers in EGNN. The initial node embedding $\mathbf{h}_i^{(0)} = \mathbf{B}^\top \mathbf{a}_i \in R^d$ embeds the i-th node, where $\mathbf{a}_i$ is defined in Definition 1. Coordinate embeddings at the l-th layer are denoted $\mathbf{X}^{(l)} = \{\mathbf{x}_i^{(l)}\}_{i=1}^N$. The initial coordinate embeddings $\mathbf{X}^{(0)} = \{\mathbf{x}_i\}_{i=1}^N$ are the real 3D coordinates of all the nodes (Eq.2).

The following equation defines the feedforward rules of EGNN, for $i, j = 1, \cdots, N$, $i \neq j$, $l = 0, 1, \cdots, L-1$, we have

$$\begin{aligned}
\mathbf{w}_{ij}^{(l+1)} &= \phi_e\Big(\mathbf{h}_i^{(l)} \oplus \mathbf{h}_j^{(l)} \oplus ||\mathbf{x}_i^{(l)} - \mathbf{x}_j^{(l)}||_2^2\Big) \quad \in R^d, \\
\mathbf{v}_i^{(l+1)} &= \sum\nolimits_{j=1,j\neq i}^N \mathbf{w}_{ij}^{(l+1)} \in R^d, \ \mathbf{h}_i^{(l+1)} = \phi_h\Big(\mathbf{h}_i^{(l)} \oplus \mathbf{v}_i^{(l+1)}\Big) \in R^d, \\
\mathbf{x}_i^{(l+1)} &= \mathbf{x}_i^{(l)} + \sum\nolimits_{j=1,j\neq i}^N \Big(\mathbf{x}_i^{(l)} - \mathbf{x}_j^{(l)}\Big)\phi_x\Big(\mathbf{w}_{ij}^{(l)}\Big) \quad \in R^3,
\end{aligned} \tag{10}$$

where $\oplus$ denotes the concatenation of vectors; $\phi_e(\cdot) : R^{2d+1} \to R^d; \phi_x(\cdot) : R^d \to R; \phi_h(\cdot) : R^{2d} \to R^d$ are all two-layer multiple layer perceptrons (MLPs) with Swish activation in the hidden layer [38]. At the l-th layer, $\mathbf{w}_{ij}^{(l)}$ represents the message vector for the edge from node i to node j; $\mathbf{v}_i^{(l)}$ represents the message vector for node i, $\mathbf{x}_i^{(l)}$ is the position embedding for node i; $\mathbf{h}_i^{(l)}$ is the node embedding for node i. $\mathbf{H}^{(L)} = [\mathbf{h}_1^{(L)}, \cdots, \mathbf{h}_N^{(L)}]$ are the node embeddings of the L-th (last) layer. We aggregate them using sum function as readout function to obtain a representation of the whole CDR loop,

$$\mathbf{h}_\mathcal{Y} = \sum\nolimits_{i=1}^N \mathbf{h}_i^{(L)} \quad \in R^d. \tag{11}$$

The scalar energy of the 3D CDR loops is

$$E_\theta(\mathcal{Y}) = E_\theta(\mathcal{A}, \mathcal{X}) = \mathbf{h}_\mathcal{Y}^\top \mathbf{z} \quad \in R, \quad \mathbf{z} \in R^d \tag{12}$$

where $\mathbf{z}$ is learnable parameter. Now we show the energy function is invariant w.r.t. rotation and translation on 3D graph structure.

Lemma 1. Suppose $\widehat{\mathcal{X}} = [\widehat{\mathbf{x}}_1, \cdots, \widehat{\mathbf{x}}_N]$ are the 3D coordinates of the graph by rotating and translating original graph whose coordinates are $\mathcal{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N]$, $E_\theta(\mathcal{Y}) = E_\theta(\mathcal{A}, \mathcal{X})$ and $E_\theta(\widehat{\mathcal{Y}}) = E_\theta(\mathcal{A}, \widehat{\mathcal{X}})$ are the corresponding scalar energy, respectively, then the output would not change, i.e., $E_\theta(\mathcal{Y}) = E_\theta(\widehat{\mathcal{Y}})$. The proof is in Appendix.

Thus, the translation-, rotation- invariances are preserved. Also, the scalar energy is differentiable w.r.t. CDR loop feature $\mathcal{A}$ and $\mathcal{X}$, thus enabling gradient based continuous optimization, as discussed below.

3.3.3 Learning. As mentioned in Sec 3.2, the core idea to train the energy model is to push down the positive samples and push up the hallucinated samples at the same time. Different from the unconstrained case in Eq.(8), in Constrained Energy Model we restrict both the positive and hallucinated samples in the manifold $\mathcal{M}$, as illustrated in Fig 2. The learning objective becomes

$$L(\theta) = \mathcal{E}_{\mathcal{Y}^{(+)}\sim P_{\text{data}}}\big[-E_\theta(\mathcal{Y}^{(+)})\big] + \mathcal{E}_{\mathcal{Y}^{(-)}\sim P_\theta}\big[E_\theta(\mathcal{Y}^{(-)})\big], \tag{13}$$

where $\mathcal{Y}^{(+)}, \mathcal{Y}^{(-)} \in \mathcal{M}$. Specifically, gradient methods do not need to evaluate $P_\theta(\mathcal{Y})$ directly, instead, it needs to evaluate the gradient of the log-probability, i.e., $-\nabla E_\theta$. At the t-th step, a sample is updated via

$$\begin{aligned}
\big(\mathcal{A}^{(-)}\big)^{(t)} &= \big(\mathcal{A}^{(-)}\big)^{(t-1)} - \lambda_t \nabla_\mathcal{A} E_\theta((\mathcal{A}^{(-)})^{(t-1)}) + \Xi, \\
\big(\mathcal{X}^{(-)}\big)^{(t)} &= \mathcal{R}_\mathcal{M}\big((\mathcal{X}^{(-)})^{(t-1)} - \lambda_t \nabla_\mathcal{X} E_\theta((\mathcal{X}^{(-)})^{(t-1)}) + \Gamma\big),
\end{aligned} \tag{14}$$

where λ_t is the step size at the t-th iteration, Ξ and Γ have the same shape with $\mathcal{A}$ and $\mathcal{X}$, respectively. Each scalar element in Ξ/Γ is i.i.d. drawn from zero-mean Gaussian distribution whose variance is $\sqrt{\lambda_t}$, i.e., $\Gamma_\mathbf{i} \sim \mathcal{N}(\mathbf{0}, \lambda_\mathbf{t})$ for any scalar element $\Gamma_i \in \Gamma$, $\Xi_\mathbf{i} \sim \mathcal{N}(\mathbf{0}, \lambda_\mathbf{t})$ for any scalar element $\Xi_\mathbf{i} \in \Xi$. Unlike the unconstrained sampling setting, our sampling space is restricted to the manifold $\mathcal{M}$, so when update $\mathcal{X}$, we need to project the updated samples to the manifold using *retraction* operation, following [4, 25]:

Definition 5 (Retraction)**.** Given a data point out of the manifold $\mathcal{M}$, retraction is the operation that projects it onto the closest data point in the manifold. Formally, we have

$$\mathcal{R}_\mathcal{M}(\mathcal{X}) = \arg\min_{\mathcal{X}'\in\mathcal{M}} ||\mathcal{X} - \mathcal{X}'||, \quad \mathcal{X} \notin \mathcal{M} \tag{15}$$

we slightly abuse the notations, omit the $\mathcal{A}$ in the expression $\mathcal{X} \notin \mathcal{M}, \mathcal{X}' \in \mathcal{M}$. $||\cdot||$ is a distance function, which is the sum of square

Euclidean distance of coordinates. The constraints of the manifold $\mathcal{M}$ are hard (Eq.3, 4). To relax it, we augment it into the learning objective and solve the following optimization problem,

$$\arg\min_{\mathbf{x}_1^{\text{new}},\cdots,\mathbf{x}_N^{\text{new}}} \sum_{i=1}^{N} ||\mathbf{x}_i^{\text{new}} - \mathbf{x}_i^{\text{old}}||_2^2 + \gamma_1 \sum_{i=1}^{N-1} \left| ||\mathbf{x}_i^{\text{new}} - \mathbf{x}_{i+1}^{\text{new}}||_2^2 - \kappa^2 \right| + \gamma_2 \left| ||\mathbf{x}_1^{\text{new}} - \mathbf{x}_N^{\text{new}}||_2^2 - \left((\xi_1 + \xi_2)/2\right)^2 \right|, \tag{16}$$

where $\{\mathbf{x}_1^{\text{old}}, \cdots, \mathbf{x}_N^{\text{old}}\}$ and $\{\mathbf{x}_1^{\text{new}}, \cdots, \mathbf{x}_N^{\text{new}}\}$ are coordinates in $\mathcal{X}$ and $\mathcal{X}'$ in Eq.15 respectively, the first term (i.e., $\sum_{i=1}^{N} ||\mathbf{x}_i^{\text{new}} - \mathbf{x}_i^{\text{old}}||_2^2$) is a decomposition of $||\mathcal{X} - \mathcal{X}'||$ in Eq.(15); the second and third terms relax the constraints defined in Eq.(3) and (4) respectively. γ_1 and γ_2 are hyperparameters that control the weight of various terms. To make sure that the constraints are met, we set $\gamma_1, \gamma_2 \gg 1$.

Algorithm 1 Constrained Energy Model (`CEM`)

1: **Input**: training CDR loops $\{\mathcal{Y}^{(+)} = (\mathcal{A}^{(+)}, \mathcal{X}^{(+)})\}$.
2: **Output**: top-K samples with lowest energy values.
3: # Learning: alternatively update energy model and hallucinated samples. Hallucinated samples $\mathcal{Y}^{(-)} = (\mathcal{A}^{(-)}, \mathcal{X}^{(-)})$ are randomly initialized.
4: **for** $t = 1, 2, \cdots, T$ **do**
5: update Constrained Energy Model E_θ via optimizing Eq. (13) (using both $\mathcal{Y}^{(+)}$ and $\mathcal{Y}^{(-)}$).
6: update hallucinated sample $\mathcal{Y}^{(-)} = (\mathcal{A}^{(-)}, \mathcal{X}^{(-)})$ (Eq. 14).
7: **end for**
8: # Inference: sampling from the distribution $P_\theta(\mathcal{Y})$ (Eq. 9).
9: **for** $t = 1, 2, \cdots,$ **do**
10: Sampling from $P_\theta(\mathcal{Y})$ based on Eq. (14)
11: **end for**
12: select top-K samples with lowest energy values (Sec 3.3.4).

3.3.4 Inference. During the inference, we fixed Constrained Energy Model and sampling from the probability distribution $P_\theta(\mathcal{Y}) = P_\theta(\mathcal{A}, \mathcal{X})$. It leverages the same sampling strategy as Eq.14. Specifically, we start from a random initialization (i.e., $\mathcal{A}^{(0)}, \mathcal{X}^{(0)}$). Then to select most promising CDR loops, given all the drawn samples, we discretize the continuous amino acid features $\mathcal{A} = [\mathbf{a}_1, \cdots, \mathbf{a}_N]$ using argmax operation on $\mathbf{a}$ to select the amino acid category. Then we feedforward the discrete amino acid feature$\mathcal{A}$, and 3D coordinates $\mathcal{X}$ to EGNN obtain their energy value. We select the samples with lowest energy value (E_θ in Eq. 9). Based on Eq. (9) ($P_\theta(\mathcal{Y}) = e^{-E_\theta(\mathcal{Y})}/Z(\theta)$), lower energy corresponds to higher likelihood P_θ. Algorithm 1 summarizes essential steps of the pipeline.

3.4 Theoretical Analysis

In this section, we analyze the theoretical properties of the proposed method. Specifically, we show that compared with vanilla energy model (unconstrained), the proposed Constrained Energy Model requires less sample complexity to reach the same empirical risk. We start with the definition of some basic concepts in statistical learning theory. The detailed explanations and examples of these definitions are available at [50].

Definition 6 (Condition number)**.** Let $\mathcal{W}$ be a smooth d-dimensional submanifold of R^m. The condition number $c(\mathcal{W})$ is defined to be $\frac{1}{\tau}$, where τ is the largest number to have the property: for any $r < \tau$, there are no two normals of length r that are incident on $\mathcal{W}$ at different points intersect. Formally, given two linear subspaces V, W, let $< V, W >$ be the angle between V and W, defined as

$$< V, W >= \arccos\left(\sup_{\mathbf{v}\in V} \inf_{\mathbf{w}\in W} \left(\mathbf{v}\cdot\mathbf{w}\right)/\left(||\mathbf{v}||_2||\mathbf{w}||_2\right)\right) \tag{17}$$

where the arccos function is the inverse of the cosine function. For any manifold, the condition number is defined as $c(\mathcal{W}) = \inf_{x,y\in\mathcal{W}} 2\sin(< T_x, T_y >)/||x - y||_2$, where the infimum is taken over distinct points $x, y \in \mathcal{W}$, sin is sine function, T_x and T_y are the tangent spaces at x and y. Tangent space of a manifold generalizes to higher dimensions the notion of tangent planes to surfaces in three dimensions and tangent lines to curves in two dimensions.

We assume that we are learning an indicator function f whose domain is $\mathcal{W}$ with binary range. Regression case requires real-valued functions on the manifold and need to consider functional analysis on the manifold [50]. Thus, binary classification (indicator function instead of real-valued function) is usually discussed to analyze the sample complexity [3, 8, 18, 50]. Then we define the collections of indicator functions (f).

Definition 7 (Collections of indicator functions)**.** Let $\mathcal{S}_\tau(\mathcal{W})$ denote all the closed sets in manifold $\mathcal{W}$, i.e., $\mathcal{S}_\tau(\mathcal{W}) = \left\{S \mid S = \overline{S} \subset \mathcal{W} \text{ and } c(S \cap \overline{\mathcal{W}\backslash S}) \leq \frac{1}{\tau}\right\}$, where $\overline{S}$ is the closure of S. Let $\mathcal{C}_\tau = \left\{f : \mathcal{W} \to \{0,1\} \mid f^{-1}(1) \in \mathcal{S}_\tau\right\}$. The class $\mathcal{C}_\tau$ is the collection of indicators of all the closed sets in $\mathcal{W}$ whose boundaries are $(1/\tau)$-conditioned (d-1)-dimensional submanifolds of R^m. Then we define annealed entropy to measure the capacity of the set of indicator functions.

Definition 8 (Annealed entropy)**.** Let $\mathcal{P}$ be a probability measure supported on a manifold $\mathcal{W}$. Given a collection of indicator functions $\mathcal{C}$ and a set of l points $Z = \{z_1, \cdots, z_l\} \subset \mathcal{W}$, $z_1, \cdots, z_l$ are l i.i.d. (independent and identically distributed) samples from $\mathcal{P}$, let $\mathcal{N}(\mathcal{C}, l)$ be the number of ways of partitioning $z_1, \cdots, z_l$ into two sets using indicators belonging to $\mathcal{C}$. Annealed entropy (a.k.a. annealed VC entropy) is defined as $H(\mathcal{C}, \mathcal{P}, l) = \log\left[\mathbb{E}_{z_1,\cdots,z_l \overset{\text{i.i.d.}}{\sim} \mathcal{P}} \mathcal{N}(\mathcal{C}, Z)\right]$.

Remarks. When the annealed entropy is large, it means that there are a lot of different ways to classify the data points and the capacity of the set of indicator functions is large. It also means the probability measure is hard to learn. In contrast, lower annealed entropy is more desirable.

Definition 9 (Packing number)**.** Let $N_p(\epsilon_r)$ be the largest number N such that $\mathcal{W}$ contains N disjoint balls $\mathcal{B}_{\mathcal{W}}(x, \epsilon_r)$, where $\mathcal{B}_{\mathcal{W}}(x, \epsilon_r)$ is a geodesic ball in $\mathcal{W}$ around x of radius ϵ_r.

Definition 10 (Risk)**.** The risk $R(\alpha)$ of a machine learning classifier α is defined as the probability that α misclassifies a random data point (x, y) (x is the input feature, y is the label, $\alpha(x)$ is the prediction) drawn from $\mathcal{P}$. Formally, $R(\alpha) = \mathbb{E}_{x\sim\mathcal{P}}[\alpha(x) \neq y]$. The empirical risk is defined as $R_{\text{emp}}(\alpha) = \sum_{i=1}^{l} \mathbf{1}(\alpha(x_i) \neq y_i)/l$, where $\mathbf{1}(\cdot)$ denotes the indicator function, x_i and y_i are input feature and label of the i-th sampled data point, respectively.

All the definitions above are well established in [50]. Then before presenting main theoretical results in Theorem 3, we show some existing theoretical results on manifold learning in Theorem 1 & 2.

Theorem 1. (Theorem 11 in [32]). Let $\mathcal{W}$ be a d-dimensional sub-manifold of R^m, whose condition number is no greater than $1/\kappa_0$. Let $\mathcal{P}$ be a probability measure supported on $\mathcal{W}$, whose density relative to the uniform probability measure on $\mathcal{W}$ is bounded by ρ. We let $\epsilon_r = \min\{\frac{\tau}{4}, \frac{\kappa_0}{4}, 1\}\xi/2\rho$. When the number n of random samples from $\mathcal{P}$ is large, i.e., $n \geq N_p(\epsilon_r/2)d\ln(2\sqrt{d}\rho^2/\xi)/\xi^2$, the annealed entropy of C_τ can be bounded as $H(C_\tau, \mathcal{P}, \lceil n - \sqrt{n\ln(2\pi n)}\rceil) \leq 4\xi n + 1$.

Theorem 2. (Vapnik [50], Theorem 4.2) For any l and ϵ, we have $\mathbb{P}\big[\sup_{\alpha\in C}(R(\alpha) - R_{\text{emp}}(\alpha))/\sqrt{R(\alpha)} > \epsilon\big] < 4e^{(\frac{H(C,\mathcal{P},2l)}{l} - \frac{\epsilon^2}{4})l}$, where random samples are sampled from the probability distribution $\mathcal{P}$.

Assumption 1. The constrained and unconstrained case has the same number of packing number (defined in Definition 9).

During the theoretical analysis, we omit the unconstrained part amino acid feature $\mathcal{A}$ and restrict our attention to the learning of coordinate feature $\mathcal{X}$. Thus, the constrained manifold becomes $\mathcal{M} = \big\{\mathcal{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N] \;\big|\; ||\mathbf{x}_1 - \mathbf{x}_2||_2 = ||\mathbf{x}_2 - \mathbf{x}_3||_2 = \cdots = ||\mathbf{x}_{N-1} - \mathbf{x}_N||_2 = \kappa, \epsilon_1 \leq ||\mathbf{x}_1 - \mathbf{x}_N||_2 \leq \epsilon_2\big\}$.

Lemma 2. The constrained manifold $\mathcal{M}$ above is a (2N+1)-dimensional, smooth sub-manifold of R^{3N}, N is number of amino acids in CDR.

Theorem 3 (Sample complexity)**.** We suppose the sample complexity (i.e., number of random samples) of `CEM` and vanilla energy model are denoted $n^{(c)}$ and $n^{(u)}$, respectively. Under Assumption 1, given ϵ and δ, to make sure the risks $R(\alpha)$ satisfies $\mathbb{P}\big[\sup_{\alpha\in C}(R(\alpha) - R_{\text{emp}}(\alpha))/\sqrt{R(\alpha)} < \epsilon\big] < \delta$, we have $n^{(c)} < (2N+1)/(3N)n^{(u)}$.

Remarks. To guarantee the same empirical learning risk, learning on the constrained manifold $\mathcal{M}$ requires less sample complexity (around two thirds) than the unconstrained method.

4 EXPERIMENT

4.1 Data and Preprocessing

This section describes the datasets and the preprocessing procedure. Structural antibody database **SabDab**[2] is a database containing 5,494 antibodies available in the Protein Data Bank (PDB), annotated and presented in a consistent fashion [10, 19]. One antibody may contain multiple CDR loops (both amino acid sequence and 3D structure). After filtering the antibodies whose CDR structures are not available, we obtain 8,381 CDR loops. The average lengths and standard deviations for H1, H2, and H3 are 7.2 ± 0.6, 5.9 ± 0.7, and 12.3 ± 4.4, respectively. The data points are randomly split into training, validation, and test set with a ratio of 8:1:1. The data is in the PDB file format with all atom coordinates available. We use the coordinate of α-carbon atom to denote the coordinate of amino acid, following [20, 21].

[2]It is publicly available at http://opig.stats.ox.ac.uk/webapps/newsabdab/sabdab/.

4.2 Evaluation Metrics

We describe metrics for amino acid sequence and 3D structures. **(1) Amino acid sequence level metrics** include: **(1.1) Protein Perplexity (PPL)**. Protein perplexity is the exponential of the average log-likelihood and measures how well a probability model predicts amino acid sequence [2, 19, 20, 57]. It is defined as $\text{Perplexity}(\mathbf{C}) = \exp\big(-(1/N)\sum_{i=1}^{N}\log Q(\mathbf{c}_i|\mathbf{C}_{<i})\big)$, where $\mathbf{C}$ is an amino acid sequence, $\mathbf{c}_i$ is the i-th amino acid in $\mathbf{C}$, $\mathbf{C}_{<i}$ denotes the first $i-1$ amino acids in the sequence $\mathbf{C}$. $Q(\mathbf{c}_i|\mathbf{C}_{<i})$ is a well-trained LSTM model. Lower perplexity values are more desirable for an amino acid sequence. **(1.2) Similarity** is measured on amino acid sequence level. We define the similarity between two sequences c_1 and c_2 as

$$\text{sim}(c_1, c_2) = |\text{LCS}(c_1, c_2)|/\max\{|c_1|, |c_2|\}, \tag{18}$$

where $|c|$ represent the length of amino acid sequence c, i.e., number of amino acids in the sequence c. $\text{LCS}(c_1, c_2)$ denotes the longest common substring between c_1 and c_2, which can be computed by dynamic programming algorithm. Similarity value ranges from 0 to 1, a higher value indicates more similarity. The distance is defined as one minus similarity. The similarity is not used directly in our experiment. It is used for evaluating diversity. **(1.3) Diversity (Div)** is defined as the average pairwise distance between the CDR loops, following the definition of the diversity of molecule set [11, 12, 19]. Specifically, suppose $\mathbb{C}$ is the set of all the generated CDR amino acid sequences, diversity is defined as $\text{diversity}(\mathbb{C}) = 1 - \frac{1}{|\mathbb{C}|(|\mathbb{C}|-1)}\sum_{c_1,c_2\in\mathbb{C},c_1\neq c_2}\text{sim}(c_1, c_2)$, where $\text{sim}(\cdot,\cdot)$ is the similarity function defined above. Diversity ranges from 0 to 1, higher diversity is more desirable.
(2) 3D graph structure level metrics include: **(2.1) Root-Mean-Square Deviation (RMSD)** measures the alignment between tested conformations $G \in R^{N\times 3}$ and reference conformation $G^r \in R^{N\times 3}$, N is the number of points in the conformation, defined as $\text{RMSD}(G, \widehat{G}) = \big((1/N)\sum_{i=1}^{N}||G_i - \widehat{G}_i||_2^2\big)^{\frac{1}{2}}$, where G_i is the i-th row of conformation G. The conformation $\widehat{G}$ is obtained by an alignment function $\widehat{G} = A(G, G^r)$, which rotates and translates the reference conformation G^r to have the smallest distance to the generated G according to the RMSD metrics, which is calculated by the Kabsch algorithm [22]. For each CDR loop in test set, to evaluate RMSD, we select the closest CDR loop (in terms of RMSD) from the set of all the generated CDR loops. **(2.2) Validity Rate (%V)** is the percentage of valid CDR loops in all the generated CDRs. A CDR loop is valid if it satisfies two criteria in Eq.(3) and (4). Higher validity is more desirable.

4.3 Baseline Methods

In this section, we briefly describe the baseline methods. The implementation details are provided in Sec B.2 in the supplementary materials. All baseline methods are SOTA approaches in antibody design [21]. We use the default setup following their original papers. **(1) Reference** evaluates the metrics of real CDR loops in test set as reference. **(2) LSTM** [1, 42] leverages a long short-term memory model to generate the amino acid sequence in an autoregressive manner and does not model 3D structure. **(3) Genetic algorithm (GA)** starts from a population of CDR loops. Given the data in the current population, we leverage crossover and mutation to produce offsprings for the next generation (i.e., iteration) [28]. **(4) Iterative Refinement Graph Neural Network (IR-GNN)** was proposed

Table 2: *de novo* antibody CDR loop (including H1, H2, H3) design results on SAbDab. On each task and for each method, we conduct 5 independent runs with different random seed and data split. The average and standard deviation of the metrics are reported. LSTM only models the amino acid sequence, so RMSD and %V results are not available. We also conduct the t-test to compare the difference between CEM and the best baseline method (IR-GNN). On each task and each metric, we highlight the best result. Symbol * denotes the results pass the t-test with p-value < 0.05. The t-test results show that all improvements are significantly better than the best baseline method (IR-GNN). Several methods, including CEM achieved similar diversity measure, while no single method is consistently better.

Task	Method	PPL (↓)	RMSD (↓)	%V (↑)	Div (↑)
H1	Reference	8.10±0.08	0.0±0.00	100.0±0.0%	0.518±0.024
	LSTM	10.20±0.23	-	-	0.553±0.045
	GA	10.48±0.26	1.99±0.13	44.0±1.9%	0.635±0.047
	AR-GNN	9.55±0.25	1.97±0.11	61.7±1.7%	0.632±0.050
	IR-GNN	9.18±0.16	1.70±0.11	87.0±1.3%	0.684±0.014
	EBM	9.84±0.27	1.92±0.25	75.3±1.8%	**0.691±0.028**
	CEM	**8.48±0.19***	**1.29±0.15***	**100.0±0.0%***	0.684±0.010
H2	Reference	8.57±0.12	0.0±0.0	100.0±0.0%	0.603±0.015
	LSTM	10.86±0.35	-	-	0.633±0.030
	GA	10.25±0.26	1.93±0.19	34.3±1.8%	0.544±0.050
	AR-GNN	10.54±0.24	1.69±0.30	34.5±1.1%	**0.671±0.038**
	IR-GNN	9.65±0.16	1.12±0.17	86.9±0.9%	0.618±0.023
	EBM	10.00±0.39	1.44±0.30	45.3±1.0%	0.665±0.031
	CEM	**9.31±0.10***	**0.99±0.11**	**100.0±0.0%***	0.664±0.025
H3	Reference	9.84±0.32	0.0±0.0	100.0±0.0%	0.745±0.031
	LSTM	12.35±0.33	-	-	0.736±0.047
	GA	12.75±0.29	4.33±0.98	13.4±%	0.713±0.054
	AR-GNN	13.01±0.13	3.80±0.52	25.8±0.9%	0.754±0.025
	IR-GNN	11.45±0.25	3.02±0.24	78.4±0.7%	0.751±0.017
	EBM	10.93±0.48	3.21±0.86	62.7±1.0%	**0.798±0.043**
	CEM	**10.49±0.15***	**2.01±0.10***	**99.0±0.3%***	0.786±0.013

for 3D antibody CDR design [21]. It first unravels amino acid sequence and iteratively refines its predicted global structure. The inferred structure in turn guides subsequent amino acid choices. **(5) Auto-Regressive Graph Neural Network (AR-GNN))** [30, 53] was originally proposed for small-molecule generation and was adapted for antibody CDR design [21]. Unlike IR-GNN that uses iterative refinement mechanism, AR-GNN generates the amino acid sequence and 3D structural graph in an autoregressive manner. **(6) Unconstrained Energy Based Model (EBM)** [27] uses the same neural architecture with the CEM, but does not consider constraints and nor the constrained manifold $\mathcal{M}$. The other setups are the same as CEM (Sec B.1). It can be seen as an ablation study that explores the empirical effect of the constrained manifold on performance. LSTM, IR-GNN, AR-GNN, EBM, and our method CEM all belong to deep generative models, while genetic algorithm (GA) is a traditional heuristic searching method. These baselines are reported with competitive performance for antibody design [1, 21, 42]. Equivariant normalizing flow [44] is an ODE type of flow model, which was considered as a baseline. However, the training process is computationally expensive since the same forward operation has to be done multiple times sequentially in order to solve the ODE equation. It also exhibited some instabilities [43] and failed in our experiment, so it is not included in the results.

4.4 Results

For all the compared methods, we conducted generation tasks on H1, H2, H3 loops and report the results in Table 2. For each task and each method, we conducted 5 independent runs with different random seeds and data split. The average results of all the metrics and their standard deviation are presented. We also conduct the t-test to compare the difference between CEM and the best baseline method (IR-GNN). From Table 2, we have the following observations: (1) **our method outperforms all the baseline methods significantly** in terms of amino acid sequence level metric (PPL, perplexity) and geometry graph metrics (RMSD, root-mean-square deviation and %V, validity rate). Specifically, compared with the best baseline method (IR-GNN), on H1/H2/H3 design tasks, our method achieves 14.9%/15.0%/26.2% relative improvement in terms of %V (87.0% versus 100.0%, 86.9% versus 100.0% and 78.4% versus 99.0%) respectively, 24.1%/11.6%/33.4% relative reduction in terms of RMSD (1.70 versus 1.29, 0.99 versus 1.12 and 3.02 versus 2.01) respectively, and 7.6%/3.5%/8.4% relative reduction in terms of PPL (9.18 versus 8.48, 9.65 versus 9.31 and 11.45 versus 10.49 respectively); (2) **H1 v.s. H2 v.s. H3**: among all the three kinds of design tasks, including H1, H2, H3 loops, almost all the methods get the highest perplexity, RMSD, and lower validity in the H3 generation task. This is consistent with the existing knowledge that CDR H3 loops have the highest variability and most challenging to design [21, 39]. (3) **Diversity**: EBM, CEM, IR-GNN, and AR-GNN perform similarly in terms of diversity, validating that our method can explore the amino acid sequence space thoroughly (diversity is measured on amino acid sequence level).

Ablation study (effect of constraints). To show the empirical effect of constraints, we also compare the results of EBM in Table 2. We observe that CEM outperforms the vanilla energy based model significantly and consistently across all the three generation tasks (H1, H2, and H3), obtaining 32.8%, 120.7%, 57.9% relative improvement in terms of validity rate (% V), respectively. The key reason behind this observation is: CEM constrains the learning space to the constrained manifold $\mathcal{M}$, and only need to discriminate (i.e., assigning lower or higher energy value) the data points on the manifold. This is also supported by the theoretical results in Sec 3.4: Constrained Energy Model is more efficient than the unconstrained energy model (i.e., vanilla energy model) in terms of sample complexity. **Examples**. This section analyzes two examples of CDR H3 loop design in Figure 4. For each example, we show the 3D structures of CDR loops of CEM (ours) and reference (actual CDR

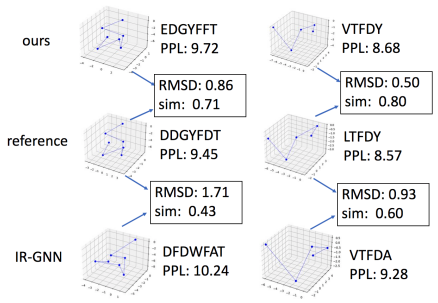

Figure 4: Two examples about CDR H3 loop design. In both examples, our method CEM outperforms the best baseline method IR-GNN in terms of PPL (perplexity), RMSD and similarity with the reference (sim).

loop in the test set) and IR-GNN (the best baseline method) as a comparison. For each visualization, the conformation (3D structure) of our method and IR-GNN are rotated and translated by an alignment function so that they have the smallest distance to the reference conformation according to the RMSD metrics, which is calculated by the Kabsch algorithm [22]. We found that on amino acid sequence level, our method is more similar to the reference than IR-GNN, e.g., in the first example (left), the similarity (defined in Eq.18) between "EDGYFFT" (generated by `CEM`) and "DDGYFDT" (the reference) is 0.71, whereas the similarity between "DFDWFAT" (IR-GNN) and "DDGYFDT" (the reference) is only 0.43. Also, `CEM` achieves lower (lower is better) perplexity than IR-GNN and are closer to the reference conformation (lower RMSD score) in both examples (0.86 versus 1.71 and 0.50 versus 0.93). These examples provide an intuitive demonstration of `CEM`.

5 CONCLUSION

We have proposed Constrained Energy Model (`CEM`) for designing 3D antibody CDR loops. We first design a constrained manifold for all the CDR loops that satisfy geometry constraints. Then we design Constrained Energy Model that learns from both positive and hallucinated samples in the constrained manifold and update hallucinated samples in the constrained manifold. Theoretical analysis shows that the sample sizes required for constrained learning on the manifold is less than about two-thirds of sample sizes for unconstrained learning. Thorough empirical studies validate `CEM`'s superiority in designing CDR H1, H2, H3 loops.

ACKNOWLEDGEMENTS

This work was supported by IQVIA, NSF award SCH-2014438, IIS-1838042, NIH award R01 1R01NS107291-01 and OSF Healthcare. We thank Wenhao Gao, Cao Xiao and Xinyu Gu for discussions.

REFERENCES

[1] Rahmad Akbar et al. 2021. In silico proof of principle of machine learning-based antibody design at unconstrained scale. *BioRXiV* (2021).
[2] Jose Juan Almagro Armenteros et al. 2020. Language modelling for biological sequences–curated datasets and baselines. *BioRxiv* (2020).
[3] Idan Attias, Aryeh Kontorovich, and Yishay Mansour. 2019. Improved generalization bounds for robust learning. In *Algorithmic Learning Theory*. PMLR.
[4] Marcus Brubaker et al. 2012. A family of MCMC methods on implicitly defined manifolds. In *AISTATS*.
[5] Yue Cao et al. 2021. Fold2Seq: A Joint Sequence (1D)-Fold (3D) Embedding-based Generative Model for Protein Design. In *ICML*.
[6] Tong Che et al. 2020. Your GAN is secretly an energy-based model and you should use discriminator driven latent sampling. *arXiv* (2020).
[7] Zak Costello et al. 2019. How to hallucinate functional proteins. *arXiv* (2019).
[8] Chen Dan et al. 2018. The sample complexity of semi-supervised learning with nonparametric mixture models. *NeurIPS* (2018).
[9] Yuanqi Du et al. 2022. MolGenSurvey: A Systematic Survey in Machine Learning Models for Molecule Design. *arXiv preprint arXiv:2203.14500* (2022).
[10] James Dunbar et al. 2014. SAbDab: the structural antibody database. *Nucleic acids research* (2014).
[11] T Fu et al. 2020. CORE: Molecule Optimization using Copy and Refine. *AAAI* (2020).
[12] T Fu et al. 2020. MIMOSA: Multi-constraint Molecule Sampling for Molecule Optimization. *AAAI* (2020).
[13] T Fu et al. 2022. Differentiable Scaffolding Tree for Molecular Optimization. *ICLR* (2022).
[14] W Gao et al. 2020. Deep learning in protein modeling and design. *Patterns* (2020).
[15] Monireh Golpour et al. 2021. The Perspective of Therapeutic Antibody Marketing in Iran: Trend and Estimation by 2025. *Advances in pharma. sci.* (2021).
[16] Ian Goodfellow et al. 2020. Generative adversarial networks. (2020).
[17] Fredrik Gustafsson et al. 2020. Energy-based models for deep probabilistic regression. In *ECCV*.
[18] SQ Han et al. 2022. ADBench: Anomaly Detection Benchmark. *arXiv* (2022).
[19] Kexin Huang et al. 2021. Therapeutics data Commons: machine learning datasets and tasks for therapeutics. *NeurIPS Track Datasets and Benchmarks* (2021).
[20] John Ingraham et al. 2019. Generative Models for Graph-Based Protein Design. *NeurIPS* (2019).
[21] Wengong Jin et al. 2022. Iterative refinement graph neural network for antibody sequence-structure co-design. *ICLR* (2022).
[22] Wolfgang Kabsch. 1976. A solution for the best rotation to relate two sets of vectors. *International Union of Crystallography* (1976).
[23] Mostafa Karimi et al. 2020. De novo protein design for novel folds using Wasserstein generative adversarial network. *J. Chem. Info. & Model.* (2020).
[24] Diederik P Kingma et al. 2013. Auto-encoding variational bayes. *arXiv* (2013).
[25] Chang Liu et al. 2016. Stochastic gradient geodesic mcmc methods. *NeurIPS* (2016).
[26] Ge Liu et al. 2020. Antibody complementarity determining region design using high-capacity machine learning. *Bioinformatics* (2020).
[27] Meng Liu et al. 2021. GraphEBM: Molecular graph generation with energy-based models. *arXiv* (2021).
[28] Xiaofeng Liu et al. 2017. Computational design of an epitope-specific Keap1 binding antibody using hotspot residues grafting and CDR loop swapping. *Sci. Rep.* (2017).
[29] Ruei-Min Lu et al. 2020. Development of therapeutic antibodies for the treatment of diseases. *Journal of biomedical science* (2020).
[30] Shitong Luo et al. 2021. 3D Generative Model for Structure-Based Drug Design. *NeurIPS* (2021).
[31] Robert M MacCallum et al. 1996. Antibody-antigen interactions: contact analysis and binding site topography. *Journal of molecular biology* (1996).
[32] Hariharan Narayanan et al. 2009. Sample Complexity of Learning Smooth Cuts on a Manifold. In *COLT*.
[33] Harini Narayanan et al. 2021. Machine learning for biologics: opportunities for protein engineering, developability, and formulation. *Trends. Pharma. Sci.* (2021).
[34] Aaron Nelson et al. 2009. Development trends for therapeutic antibody fragments. *Nature Bio.* (2009).
[35] Jaroslaw Nowak et al. 2016. Length-independent structural similarities enrich the antibody CDR canonical class model. In *MAbs*. Taylor & Francis.
[36] RJ Pantazes et al. 2010. OptCDR: a general computational method for the design of antibody CDR for targeted epitope binding. *Protein Engineering* (2010).
[37] Yifei Qi et al. 2020. DenseCPD: improving the accuracy of neural-network-based protein sequence design with DenseNet. *J. chem. info. & model.* (2020).
[38] Prajit Ramachandran et al. 2017. Searching for activation functions. *arXiv* (2017).
[39] Cristian Regep et al. 2017. The H3 loop of antibodies shows unique structural characteristics. *Proteins: Structure, Function, and Bioinformatics* (2017).
[40] Donatas Repecka et al. 2021. Expanding functional protein sequence spaces using generative adversarial networks. *Nature Machine Intelligence* (2021).
[41] D Rezende et al. 2015. Variational inference with normalizing flows. In *ICML*.
[42] Koichiro Saka et al. 2021. Antibody design using LSTM based deep generative model from phage display library for affinity maturation. *Scientific reports* (2021).
[43] Victor Satorras et al. 2021. E(n) equivariant graph neural networks. *ICML* (2021).
[44] Victor Satorras et al. 2021. E(n) equivariant normalizing flows. *arXiv* (2021).
[45] Inbal Sela-Culang, Vered Kunik, and Yanay Ofran. 2013. The structural basis of antibody-antigen recognition. *Frontiers in immunology* (2013).
[46] Chence Shi et al. 2020. GraphAF: a Flow-based Autoregressive Model for Molecular Graph Generation. In *ICLR*.
[47] Sam Sinai et al. 2017. Variational auto-encoding of protein sequences. *NeurIPS Machine Learning in Computational Biology (MLCB) workshop* (2017).
[48] Robyn L Stanfield and Ian A Wilson. 2014. Antibody structure. *Microbiology spectrum* 2, 2 (2014), 2–2.
[49] Alexey Strokach et al. 2020. Fast and flexible protein design using deep graph neural networks. *Cell Systems* (2020).
[50] Vladimir Vapnik. 1999. *The nature of statistical learning theory*. Springer science.
[51] Larry Wasserman. 2004. *All of statistics: a concise course in statistical inference*.
[52] Max Welling and Yee W Teh. 2011. Bayesian learning via stochastic gradient Langevin dynamics. In *ICML*.
[53] Jiaxuan You et al. 2018. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *ICML*.
[54] Chengxi Zang et al. 2020. MoFlow: an invertible flow model for generating molecular graphs. In *SIGKDD*.
[55] Yang Zhang. 2008. I-TASSER server for protein 3D structure prediction. *BMC bioinformatics* (2008).
[56] Yuan Zhang et al. 2020. ProDCoNN: Protein design using a convolutional neural network. *Proteins: Structure, Function, and Bioinformatics* (2020).
[57] Yue Zhao et al. 2021. Pyhealth: A python library for health predictive models. *arXiv* (2021).

A DETAILED THEORETICAL ANALYSIS

Overview. We first define some concepts required for theoretical analysis, from Definition 6-10. All the definitions above are well established in [50]. Then before presenting the main theoretical results in Lemma 2 and Theorem 3, we show some existing theoretical results on manifold learning in Theorem 1 and 2.

A.1 Proof of Lemma 1

Lemma. Suppose $\widehat{\mathcal{X}} = [\widehat{\mathbf{x}}_1, \cdots, \widehat{\mathbf{x}}_N]$ are the 3D coordinates of the graph by rotating and translating the original 3D graph whose coordinates are $\mathcal{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N]$, $E_\theta(\mathcal{Y}) = E_\theta(\mathcal{A}, \mathcal{X})$ and $E_\theta(\widehat{\mathcal{Y}}) = E_\theta(\mathcal{A}, \widehat{\mathcal{X}})$ are the corresponding scalar energy, respectively, then we have $E_\theta(\mathcal{Y}) = E_\theta(\widehat{\mathcal{Y}})$. That is, rotating or translating the 3D graph structure would not change the model's output.

Proof. Any transformation (rotation and translation) on a point in 3D Euclidean space can be represented as multiplying a 3×3 orthogonal matrix and adding a three-dimensional vector. Without loss of generalization, we suppose rotation matrix is $\mathbf{Q} \in R^{3\times3}$, which is an orthogonal matrix satisfying $\mathbf{Q}^\top\mathbf{Q} = \mathbf{Q}\mathbf{Q}^\top = \mathbf{I}_3$, where $\mathbf{I}_3 \in R^{3\times3}$ is identity matrix; we also suppose the added three-dimensional vector is $\mathbf{b} \in R^3$. Transforming (rotation and translation) 3D graph whose coordinates are $\mathcal{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N]$ results in a new 3D graph whose coordinates are

$$\widehat{\mathcal{X}} = [\widehat{\mathbf{x}}_1, \cdots, \widehat{\mathbf{x}}_N] = [\mathbf{Q}\mathbf{x}_1 + \mathbf{b}, \cdots, \mathbf{Q}\mathbf{x}_N + \mathbf{b}], \tag{19}$$

where we use the notations $\widehat{\ }$ to represent the features/variables after transformation.

Then we revisit the feedforward rule of equivariant graph neural network (EGNN) defined in Equation (10), for the original (before transformation) feature, that is, for $l = 0, 1, 2, \cdots, L-1$, we have

$$\begin{aligned}
\mathbf{w}_{ij}^{(l+1)} &= \phi_e\left(\mathbf{h}_i^{(l)} \oplus \mathbf{h}_j^{(l)} \oplus ||\mathbf{x}_i^{(l)} - \mathbf{x}_j^{(l)}||_2^2\right) \in R^d, \\
\mathbf{v}_i^{(l+1)} &= \sum_{j=1, j\neq i}^{N} \mathbf{w}_{ij}^{(l+1)} \in R^d, \\
\mathbf{x}_i^{(l+1)} &= \mathbf{x}_i^{(l)} + \sum_{j=1, j\neq i}^{N} \left(\mathbf{x}_i^{(l)} - \mathbf{x}_j^{(l)}\right)\phi_x\left(\mathbf{w}_{ij}^{(l)}\right) \in R^3, \\
\mathbf{h}_i^{(l+1)} &= \phi_h\left(\mathbf{h}_i^{(l)} \oplus \mathbf{v}_i^{(l+1)}\right) \in R^d,
\end{aligned} \tag{20}$$

For in the transformed 3D graph structure, the amino acid category feature $\mathcal{A}$ remains the same thus the node embeddings $\mathbf{h}^0$ stay the same. We prove the results using *mathematical induction*. The proof by mathematical induction usually consists of two essential steps. In the first step, we need to prove the base case (n = 0) holds. The second step (a.k.a. the induction step) proves that if the statement holds for any given case n=k, then it must also hold for the next case n=k+1. Then we have

$$\widehat{\mathbf{w}}_{ij}^{(l+1)} = \phi_e\left(\mathbf{h}_i^{(l)} \oplus \mathbf{h}_j^{(l)} \oplus ||\widehat{\mathbf{x}}_i^{(l)} - \widehat{\mathbf{x}}_j^{(l)}||_2^2\right) \in R^d, \tag{21}$$

where we use $\widehat{\mathbf{w}}_{ij}^{(l+1)}$ to denote the message vector for the edge (i, j) in the transformed graph. We expand the term $||\widehat{\mathbf{x}}_i^{(l)} - \widehat{\mathbf{x}}_j^{(l)}||_2^2$ as

$$\begin{aligned}
&||\widehat{\mathbf{x}}_i^{(l)} - \widehat{\mathbf{x}}_j^{(l)}||_2^2 = ||(\mathbf{Q}\mathbf{x}_i^{(l)} + b) - (\mathbf{Q}\mathbf{x}_j^{(l)} + b)||_2^2 \\
=&||\mathbf{Q}(\mathbf{x}_i^{(l)} - \mathbf{x}_j^{(l)})||_2^2 = (\mathbf{x}_i^{(l)} - \mathbf{x}_j^{(l)})^\top\mathbf{Q}^\top\mathbf{Q}(\mathbf{x}_i^{(l)} - \mathbf{x}_j^{(l)}) \\
=&||\mathbf{x}_i^{(l)} - \mathbf{x}_j^{(l)}||_2^2
\end{aligned} \tag{22}$$

where the third equality holds because $\mathbf{Q}$ is an orthogonal matrix. Also, translation, rotation does not change the node embeddings $\mathbf{h}_i^{(l)}$, because it is only related to the category of amino acids. Thus, for the first line in Equation 20, we have:

$$\widehat{\mathbf{w}}_{ij}^{(l+1)} = \mathbf{w}_{ij}^{(l+1)} \tag{23}$$

Then based on the definition of $\mathbf{v}_i^{(l+1)}$ (the second line in Equation 20), we have:

$$\widehat{\mathbf{v}}_i^{(l+1)} = \sum_{j=1, j\neq i}^{N} \widehat{\mathbf{w}}_{ij}^{(l+1)} = \sum_{j=1, j\neq i}^{N} \mathbf{w}_{ij}^{(l+1)} = \mathbf{v}_i^{(l+1)} \tag{24}$$

Then based on the definition of $\mathbf{x}_i^{(l+1)}$, we have:

$$\begin{aligned}
\widehat{\mathbf{x}}_i^{(l+1)} =& \widehat{\mathbf{x}}_i^{(l)} + \sum_{j=1, j\neq i}^{N} \left(\widehat{\mathbf{x}}_i^{(l)} - \widehat{\mathbf{x}}_j^{(l)}\right)\phi_x\left(\widehat{\mathbf{w}}_{ij}^{(l)}\right) \\
=&\mathbf{Q}\mathbf{x}_i^{(l)} + \mathbf{b} + \sum_{j=1, j\neq i}^{N} \left(\mathbf{Q}\mathbf{x}_i^{(l)} + \mathbf{b} - (\mathbf{Q}\mathbf{x}_j^{(l)} + \mathbf{b})\right)\phi_x\left(\widehat{\mathbf{w}}_{ij}^{(l)}\right) \\
=&\mathbf{Q}\mathbf{x}_i^{(l)} + \mathbf{b} + \sum_{j=1, j\neq i}^{N} \left(\mathbf{Q}\mathbf{x}_i^{(l)} + \mathbf{b} - (\mathbf{Q}\mathbf{x}_j^{(l)} + \mathbf{b})\right)\phi_x\left(\mathbf{w}_{ij}^{(l)}\right) \\
=&\mathbf{Q}\left[\mathbf{x}_i^{(l)} + \sum_{j=1, j\neq i}^{N} \left(\mathbf{x}_i^{(l)} - \mathbf{x}_j^{(l)}\right)\phi_x\left(\mathbf{w}_{ij}^{(l)}\right)\right] + \mathbf{b} = \mathbf{Q}\mathbf{x}_i^{(l+1)} + \mathbf{b}
\end{aligned} \tag{25}$$

In addition, based on definition of $\mathbf{h}_i^{(l+1)}$ (the fourth line in Equation 20), we have:

$$\widehat{\mathbf{h}}_i^{(l+1)} = \phi_h\left(\widehat{\mathbf{h}}_i^{(l)} \oplus \widehat{\mathbf{v}}_i^{(l+1)}\right) = \phi_h\left(\mathbf{h}_i^{(l)} \oplus \mathbf{v}_i^{(l+1)}\right) = \mathbf{h}_i^{(l+1)} \tag{26}$$

First, we consider the base case in mathematical induction. Specifically, we can prove that Equation 23, 24, 25 and 26 hold for $l = 0$.

Then we consider the second step (induction step) for $l = 0, 1, 2, \cdots, L-1$, i.e., suppose Equation 23, 24, 25 and 26 hold for l, we can prove they hold for $l + 1$. Finally, we get $\widehat{\mathbf{h}}_i^{(L)} = \mathbf{h}_i^{(L)}$. Then we extend the equality to the graph level representation (Equation 11),

$$\widehat{\mathbf{h}}_{\widehat{\mathcal{Y}}} = \sum_{i=1}^{N} \widehat{\mathbf{h}}_i^{(L)} = \sum_{i=1}^{N} \mathbf{h}_i^{(L)} = \mathbf{h}_{\mathcal{Y}} \tag{27}$$

The scalar energy (Equation 12) of the transformed 3D CDR loops is $E_\theta(\widehat{\mathcal{Y}}) = (\widehat{\mathbf{h}}_{\widehat{\mathcal{Y}}})^\top\mathbf{z} = \mathbf{h}_{\mathcal{Y}}^\top\mathbf{z} = E_\theta(\mathcal{Y})$. Proved. □

A.2 Proof of Lemma 2

Lemma. The constraint manifold $\mathcal{M}$ (Definition 4) is a (2N+1)-dimensional, smooth sub-manifold of R^{3N}, where N is the number of amino acids in the CDR loop.

Proof. As mentioned in Section 3.4, we restrict our attention to the learning of 3D coordinates $\mathcal{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N]$, where $\mathbf{x}_1, \cdots, \mathbf{x}_N \in R^3$. The dimension of the unconstrained space is $3N$. As mentioned

in Equation (5), we have $N-1$ hard constraints, i.e., $||\mathbf{x}_1-\mathbf{x}_2||_2=\kappa$; $\cdots$; $||\mathbf{x}_{N-1}-\mathbf{x}_N||_2=\kappa$. In statistics, the number of degrees of freedom is the number of values in the final statistic calculation that are free to vary [51]. Each constraint would decrease the degree of freedom by 1. On the other hand, another constraint $\epsilon_1 \le ||\mathbf{x}_1-\mathbf{x}_N||_2 \le \epsilon_2$ would not decrease the dimension of the constrained space. Thus, the dimension of the manifold is $3N-(N-1)=2N+1$. In addition, each constraint is smooth, indicating the boundary of $\mathcal{M}$ is smooth. In sum, $\mathcal{M}$ is a (2N+1)-dimensional, smooth sub-manifold of R^{3N}. Proved. □

A.3 Proof of Theorem 3

Theorem [Sample complexity] we suppose the sample complexity (i.e., number of random samples) of CEM and vanilla energy model are denoted $n^{(c)}$ and $n^{(u)}$, respectively. Under Assumption 1, given ϵ and δ, to make sure the risks $R(\alpha)$ satisfies $\mathbb{P}\big[\sup_{\alpha\in C}\frac{R(\alpha)-R_{\text{emp}}(\alpha)}{\sqrt{R(\alpha)}}>\epsilon\big]<\delta$, we have $n^{(c)}<\frac{2N+1}{3N}n^{(u)}$.

Proof. Based on Theorem 2, we can get the upper bound of the annealed entropy: $H(C,\mathcal{P},2l)\le\log\frac{\delta}{4}+\frac{\epsilon^2 l}{4}$. It holds for both Constrained Energy Model and unconstrained model. Then we use superscript (c) to denote constrained model and (u) for the unconstrained model. Then based on Theorem 1, since we are interested in minimal sample complexity, we let $n=N_p(\frac{\epsilon_r}{2})\frac{d\ln(2\sqrt{d}\rho^2/\xi)}{\xi^2}$. Based on Assumption 1, we know that the term $N_p(\frac{\epsilon_r}{2})$ is the same in both constrained and unconstrained cases. In the constrained case, the dimension $d=2N+1$; for unconstrained learning, $d=3N$. We have $n^{(c)}<n^{(u)}$ and $\xi^{(c)}>\xi^{(u)}$. Then $\frac{n^{(c)}}{n^{(u)}}=\frac{2N+1}{3N}\cdot\frac{\ln(2\sqrt{2N+1}\rho^2/\xi^{(c)})}{\ln(2\sqrt{3N}\rho^2/\xi^{(u)})}\cdot\big(\frac{\xi^{(u)}}{\xi^{(c)}}\big)^2<\frac{2N+1}{3N}$. Proved. □

B MORE DETAILS ABOUT EXPERIMENT

B.1 Implementation Details

In this section, we describe the implementation details to enhance reproducibility. We include neural network architectures, the setup of hyperparameters and software/hardware configurations.

First, we describe the neural networks' architectures. The setup of the equivariant graph neural network (EGNN) follows [43]. The latent dimension of EGNN is $d=100$ (Equation 10). The number of layers in the EGNN is set to $L=5$. The hidden dimensions of two-layer MLP (multiple-layer perceptrons, including $\phi_e(\cdot): R^{2d+1}\to R^d; \phi_x(\cdot): R^d\to R; \phi_h(\cdot): R^{2d}\to R^d$ in Equation 10) are set to 50, where the Swish activation function is used to provide non-linearity in hidden layer of MLPs [38]. In the output layer of MLP, different activation functions are leveraged. $\phi_e(\cdot)$ and $\phi_h(\cdot)$ use Swish activation function; $\phi_x(\cdot)$ uses Tanh activation function. These setups follow [43]. Sum function is leveraged as a readout function to aggregate the node-level embeddings into 3D graph-level embedding. The batch size is set to 32. The model is trained for 10 epochs using the Adam optimizer with a learning rate 10^{-4}.

On each generation task (H1, H2, H3), we generate 10K CDR loops and evaluate their VR (validity rate), PPL (perplexity), and Div (Diversity). When evaluating geometric graph metrics RMSD, for each tested CDR loop in the test set, we select the CDR loop that has the smallest RMSD with the test CDR loops from the set of all generated CDR loops and report the RMSD.

Finally, we describe the software/hardware configuration. Our method is implemented using Python 3.7, Pytorch 1.7. Both training and inference procedures are conducted on NVIDIA Pascal Titan X GPU. For each tasks (H1, H2, H3), the training procedure can be finished in 10 hours.

B.2 Implementation Details of Baseline methods

In this section, we describe the implementation details about baseline methods. All these baselines are state-of-the-art approaches in antibody design. We use the default setup following the original paper. **(1) Reference** evaluates the metrics of real CDR loops in the test set as reference. **(2) LSTM** [1, 42] leverages LSTM (long short-term memory) to generate the amino acid sequence in an autoregressive manner and does not model 3D structure. Following the original paper, it is trained by the Adam optimizer and a learning rate of 0.01. Dropout rates were chosen from 0.1 and 0.2 to regularize all layers. The number of layers in LSTM is 3. The hidden dimension of LSTM is set to 256. **(3) GA (genetic algorithm)** starts from a population of CDR loops. Given the data in the current population, to generate offsprings for the next generation, we leverage the following two operations: (i) crossover (switching subsequence between two loops) and (ii) mutation (randomly changing one amino acid) to produce a new candidate in each iteration [28]. The population size of GA is set to 100, the number of generations is set to 1K, which is sufficient for GA to converge. In each generation, when producing the offsprings, we generate 1K offsprings, where half of them are produced through crossover operation, half are produced through mutation operation. We use perplexity as the oracle to select the most promising offspring for the next generation. **(4) IR-GNN (Iterative refinement graph neural network)** was proposed for 3D antibody CDR design [21]. It first unravels amino acid sequence and iteratively refines its predicted global structure. The inferred structure, in turn guides subsequent amino acid choices. Following their original paper, for the GNN, both its structure and sequence MPN have four message passing layers, with a hidden dimension of 256 and block size b = 4. It is trained by the Adam optimizer with a dropout of 0.1 and a learning rate of 0.001. **(5) AR-GNN (auto-regressive graph neural network)** [30, 53] was originally proposed for small-molecule generation and was adapted for antibody CDR design [21]. Different from IR-GNN that uses an iterative refinement mechanism, AR-GNN generates the amino acid sequence and 3D structural graph in an autoregressive manner. Following [21, 30, 53], Adam optimizer with a dropout of 0.1 is used with a learning rate of 0.001. **(6) EBM (vanilla energy based model)** is the vanilla (i.e., unconstrained) energy based model [27] that does not consider constraints and does not use constrained manifold $\mathcal{M}$. It use the same neural architecture with the CEM. The other setups are the same as CEM (Section B.1). It can be seen as an ablation study that explores the empirical effect of the constrained manifold on performance.