

# MODERN MACHINE LEARNING METHODS FOR PROTEIN DESIGN

by

Daniel Berenberg

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

NEW YORK UNIVERSITY

AUGUST, 2025

---

Professor Richard Bonneau

---

Professor Kyunghyun Cho

© DANIEL BERENBERG

ALL RIGHTS RESERVED, 2025

# DEDICATION

For Max & Chiara

## ACKNOWLEDGMENTS

I would like to extend my deepest gratitude to all of those who have supported and guided me during my PhD. My advisors Richard Bonneau and Kyunghyun Cho not only challenged me to be a better scientist, but also gave me incredible opportunities to learn new things. I would also like to thank my committee members, Jin Montclare, Neville Sanjana, and Sumit Chopra, for providing insightful feedback on this dissertation.

Vladimir Gligorijević was a patient, wise, and tenacious exemplar as a scientist, mentor, and friend. At the Flatiron Institute, Julia Koehler Leman and Doug Renfrew gave me an unforgettable crash course on protein structure and always made the time to meet and discuss research.

Much of my PhD was spent working with the talented scientists at Prescient Design. Some of the most memorable experiences were late nights working alongside Stephen Ra, Andy Watkins, and Simon Kelow interrogating designs (the “Golden Children”). I am grateful to have learned so much from Nathan Frey and Saeed Saremi, Henri Dwyer, Jae Hyeon Lee, JiWon Park, Natasa Tagasovska, and all of those with whom I worked closely during that time.

I appreciate the time spent with and feedback given from my NYU colleagues and collaborators Nate Gruver, Alan Amin, and Louis Mittel. I am very thankful for my fellow lab mates: Meet Barot, Maggie Beheler-Amass, Tymor Hamamsy, Claudia Skok-Gibbs, and Harsh Srivastava for sharing their science with me and being great, supportive friends during the entire process. Finally, I am thankful for the encouragement of family: Linda, Spencer, Darcy, Marilyn, Jim, and friends: Eric, Eli, and Sam.

# ABSTRACT

Designing biosynthetic molecules such as proteins is critical for applications in therapeutics and agriculture, yet the vast sequence space and complex functional landscape pose significant challenges. Previous design workflows rely on clustering, mechanistic modeling, or directed evolution and are often constrained by hand-crafted heuristics and domain-specific biases. Advances in deep generative modeling and protein databases of unprecedented size present an opportunity to apply modern machine learning techniques. In this work, we develop methods to generate and score protein sequences. We propose several steering and guidance techniques that balance data-driven exploration with expert-guided refinement. Leveraging established classifications of antibodies, we enable targeted redesign of designated regions for applications such as affinity maturation and framework optimization. Expanding the scope to general sequence design, we show effective classifier-guided generation of protein sequences using a novel sequence denoising autoencoder. Finally, we investigate the utility of natural language text embeddings in classifier-free generation and show the capabilities of text conditioned models on downstream generative modeling tasks. These works represent a modern framework for protein engineering that incorporates domain knowledge into a human-centered interface.

# Contents

<b>Dedication</b>	iii
<b>Acknowledgments</b>	iv
<b>Abstract</b>	v
<b>List of Figures</b>	xii
<b>List of Tables</b>	xvi
<b>List of Appendices</b>	xix
<b>1 Introduction</b>	1
<b>2 Background Material</b>	6
2.1 Protein biology, structure, and function . . . . .	6
2.1.1 Molecular composition of proteins . . . . .	7
2.1.1.1 Sequence homology . . . . .	8
2.2 Early methods in protein design . . . . .	8
2.3 Bioinformatics tools . . . . .	9
2.3.1 Sequence clustering . . . . .	9
2.3.2 Databases and Annotation Types . . . . .	10
2.3.2.1 Annotations . . . . .	10

2.4	Protein sequence modeling . . . . .	11
2.4.1	Mathematical description . . . . .	12
2.4.1.1	Generative models for sequence design . . . . .	13
2.4.2	Historical context of biological sequence modeling . . . . .	13
2.4.3	Transformer revolution . . . . .	14
2.4.3.1	Inductive bias and masking . . . . .	15
<b>3</b>	<b>Discrete Walk Jump Sampler</b>	<b>17</b>
3.1	Introduction . . . . .	17
3.2	Background . . . . .	20
3.2.1	Energy-based models . . . . .	20
3.2.2	Neural empirical Bayes . . . . .	21
3.3	Antibody discovery and design . . . . .	22
3.3.1	Discrete walk-jump sampling . . . . .	22
3.3.2	Derivation of optimal noise level for discrete sequence data . . . . .	25
3.3.3	Distributional conformity score . . . . .	26
3.4	Experiments . . . . .	28
3.4.1	dWJS generates natural, novel, diverse antibodies <i>in silico</i> . . . . .	29
3.4.2	dWJS generates natural, novel, diverse antibodies <i>in vitro</i> . . . . .	31
3.4.3	dWJS generates functional antibody variants <i>in vitro</i> . . . . .	31
3.5	Related Work . . . . .	33
3.6	Conclusions . . . . .	34
3.7	Retrospective . . . . .	34
<b>4</b>	<b>Deep Manifold Sampler</b>	<b>36</b>
4.1	Function-guided Design by Deep Manifold Sampling . . . . .	36
4.1.1	Introduction . . . . .	36

4.1.2	Related Work . . . . .	37
4.1.3	Methods . . . . .	38
4.1.4	A sequence denoising autoencoder . . . . .	39
4.1.4.1	Length prediction and transformation . . . . .	40
4.1.4.2	A protein function classifier . . . . .	40
4.1.4.3	Function-conditioned sampling . . . . .	41
4.1.5	Results . . . . .	41
4.2	Multi-segment Preserving Sampling for Deep Manifold Sampler . . . . .	43
4.2.1	Introduction . . . . .	43
4.2.2	Background: the Deep Manifold Sampler . . . . .	44
4.2.3	Multi-Segment Preserving Sampling . . . . .	46
4.2.4	Experiments . . . . .	49
4.2.4.1	Training details . . . . .	50
4.2.5	Sampling details and results . . . . .	51
4.2.6	Conclusion . . . . .	51
4.3	Retrospective . . . . .	54
<b>5</b>	<b>Natural language conditioned protein design</b>	<b>55</b>
5.1	Introduction . . . . .	55
5.2	Related work . . . . .	56
5.3	Methods . . . . .	57
5.3.1	Evaluation . . . . .	58
5.3.1.1	Mutation effect prediction benchmark . . . . .	58
5.3.1.2	Molecular Programming Benchmark . . . . .	59
5.3.1.3	Concept-conditioned distributional conformity score . . . . .	59
5.3.2	Text-augmented training data . . . . .	61

5.3.3	CORGI model description . . . . .	62
5.4	Results . . . . .	63
5.4.1	Mutation Effect Prediction . . . . .	63
5.4.2	Text-conditioned <i>de novo</i> design . . . . .	65
5.4.3	Discussion . . . . .	66
5.5	Conclusion . . . . .	67
<b>6</b>	<b>Conclusion</b>	<b>69</b>
<b>A</b>	<b>Supplementary Material For Chapter 3</b>	<b>72</b>
A.1	Network architectures and training details . . . . .	72
A.1.1	Discrete Walk-Jump Samplers . . . . .	72
A.1.2	dWJS stabilizes and simplifies training . . . . .	74
A.1.3	Diffusion baselines . . . . .	74
A.1.4	Language model baselines . . . . .	75
A.1.5	Effect of choice of $\sigma$ . . . . .	75
A.1.6	Estimation of $\sigma_c$ hyperparameter . . . . .	76
A.2	Additional algorithms . . . . .	76
A.2.1	Gradient flow enables local minima finding . . . . .	76
A.2.2	Langevin MCMC Update . . . . .	78
A.2.3	Neural Empirical Bayes . . . . .	78
A.3	Performance profiling . . . . .	80
A.4	Few-shot, in-context learning and prompts . . . . .	80
A.4.1	GPT 4 prompt . . . . .	80
A.4.2	GPT 3.5 prompt . . . . .	80
A.4.3	IgLM prompt . . . . .	81
A.5	<i>in vitro</i> validation . . . . .	82

A.5.1	Experimental results . . . . .	82
A.5.2	Experimental details . . . . .	82
A.6	Further discussion of distributional conformity score . . . . .	83
A.7	Further discussion of related work . . . . .	84
<b>B</b>	<b>Supplementary Material For Chapter 4</b>	<b>86</b>
B.1	Sequence corruption process . . . . .	86
B.2	Data collection . . . . .	86
B.3	Training . . . . .	88
B.4	Designing protein sequences with novel secondary structures . . . . .	89
<b>C</b>	<b>Supplementary Material For Chapter 5</b>	<b>91</b>
C.1	Annotation coverage . . . . .	91
C.2	More information on long form descriptions . . . . .	91
C.3	More information on conditional distributional conformity score . . . . .	92
C.4	Architecture and training details . . . . .	94
C.4.1	Global annotation network . . . . .	94
C.5	Estimating the optimal noise level . . . . .	95
C.6	Scaling CORGI on large sequence databases . . . . .	99
C.6.1	Model details . . . . .	99
C.6.1.1	Residue-level annotation network . . . . .	99
C.6.2	Residue-level annotation network . . . . .	99
C.6.2.1	Greedy annotation subset selection . . . . .	100
C.6.2.2	Pre-training dataset . . . . .	100
C.6.2.3	Global annotation network . . . . .	101
C.6.2.4	Residue-level annotation coverage . . . . .	103
C.6.3	Results . . . . .	103

C.6.4	Task-specific conditioning . . . . .	105
C.7	Analysis of CORGI-WJS sampling trajectories . . . . .	106
<b>Bibliography</b>		<b>109</b>

# List of Figures

2.1	2.1(a): Diagram of a two amino acids joined by a peptide bond. Each $R$ -group represents a side-chain. 2.1(b): Quaternary structure of a crystallized Chymotrypsin protein (PDB ID: 2HCA). Loops, $\alpha$ -helices, and $\beta$ -strands are colored in magenta, cyan, and red respectively. . . . .	7
3.1	Selected samples from a single Markov chain Monte Carlo sampling run of discrete Walk-Jump sampling (our method). Protein color corresponds to different antibody germlines (classes). Samples are folded with EquiFold [1] for visualization purposes. Discrete walk-jump sampling exhibits fast mixing and explores diverse modes of the distribution in a single chain. . . . .	20
3.2	Discrete walk-jump sampling. <b>a</b> The noising and denoising process is applied to antibody proteins. <b>b</b> Discrete inputs $x$ are smoothed with isotropic Gaussian noise, $\varepsilon \sim \mathcal{N}(0, \sigma^2 I_d)$ , to noisy inputs, $y = x + \varepsilon$ . A discrete energy-based model (dEBM) parameterizes the energy function $f_\theta(y)$ of noisy data. Noisy data is sampled with the energy function, and denoised with a separate denoising ByteNet network to clean samples, $\hat{x}_\phi(y)$ . <b>c</b> The “walk” sampling steps on the noisy data manifold with Langevin MCMC are totally decoupled from the “jump” steps to clean samples. <b>d</b> The dEBM takes noisy inputs $y$ , concatenates them with a 1d positional encoding, $p_{1d}$ , passes through an MLP and a 3 layer CNN, and concatenates the outputs with an embedding $z_s$ of the inputs into a hidden state, $h$ . $h$ is passed through an MLP and returns the energy $f_\theta(y)$ . . . . .	22

3.3	<i>in silico</i> designs sampled with dWJS are compared to a reference set of validation samples. Distributions are characterized with a set of sample quality metrics. Joint density estimation is used to compute the likelihood of designs versus the validation set and likelihoods are condensed into a <i>distributional conformity score</i> that characterizes the similarity of generated samples to the reference set. . . . .	27
4.1	(A) A designed sequence of $C_a^{2+}$ -binding protein obtained by altering the sequence of calmodulin, calcium-binding protein (PDB: 6TJ5, chain A) after removing its calcium binding site. (B) Redesign of <i>fusarium solani pisi</i> cutinase (PDB ID: 1AGY, chain A) cutinases with enhanced functions. . . . .	39
4.2	Multi-segment preserving sampling for $\beta = 1$ . (A) Non-preserved segments $\bar{s}$ are corrupted using corruption process $C$ , for which a given token (yellow) may be randomly perturbed (blue). This is encoded as hidden vector set $h$ . Length change predictor $p_\theta(\Delta l   \bar{h})$ outputs $\Delta l$ , which is distributed across $\bar{s}$ (Eq. 4.8). (B) Segment-preserving sampling follows similar operations on preserved segment $s$ (red) with notable differences. Corruption $C$ yields an unaltered sequence $\tilde{x}$ and we carry over hidden vector $h_t$ of a token within preserved segment $\tilde{s}$ with strength $\beta$ (Eq. 4.9). . . . .	47
4.3	The normalized distribution of the CDR3 lengths of the deep manifold sampler-generated sequences (“Samples”) and the test set sequences (“Training (OAS)”) with four different $\beta$ parameters. From Top Left, Clockwise: Samples were generated with $\beta = 0, 0.1, 0.9$ , and $0.5$ . . . . .	52
4.4	The normalized distribution of the GPT-2 scores of the deep manifold sampler-generated sequences (“Samples”) and the test set sequences (“Training (OAS)”) with four different $\beta$ parameters. From Top Left, Clockwise: Samples were generated with $\beta = 0, 0.1, 0.9$ , and $0.5$ . . . . .	53

4.5	The distribution of edit distances between generated samples and their seed sequences with varying settings of $\beta$ (0, 0.1, 0.5, and 0.9). . . . .	53
5.1	Schematic of the concept-conditioned distributional conformity score. Protein records are stored in a vector database indexed by the annotation text representation. Sequence representations of the nearest annotations are used to parameterize the conformity model. The conformity model in turn scores generated sequences by their sequence embeddings. . . . .	60
5.2	CORGI model schematic. . . . .	62
5.3	<i>Left:</i> Validity of samples assessed by proxy via the ESM2-35M negative log-likelihood (NLL). Smaller NLL values correspond to more valid examples. The NLL distributions for every model stand within an acceptable range (i.e., below 1). <i>Right:</i> Empirical cumulative distributions of concept conditional conformity scores. Given a C2DCS value $\alpha$ on the $x$ -axis, the corresponding value on the $y$ -axis may be interpreted as the fraction of examples that reject the null hypothesis that the samples are exchangeable with the reference distribution at significance level $\alpha$ . . . . .	66
A.1	Histogram of $\chi_{ii'}$ values for random samples from the paired observed antibody space Olsen, Boyles, and Deane [2] dataset. . . . .	76
A.2	Expression yield (mg) and binding affinity (pKD) of sequence designs from our method targeting the ERBB2 antigen. . . . .	82
B.1	Architecture of the sequence DAE. . . . .	89
B.2	A designed sequence of $\alpha$ -helical protein obtained by altering the sequence of a $\beta$ -protein by conditioning the sampling process with <i>ion transmembrane transporter activity</i> function label. . . . .	90

C.1	Summary information on the SwissProt text-augmented dataset. The left plot depicts the length distribution for functional descriptions and protein sequences. The right plot indicates the count distribution of label codes per protein. . . . .	92
C.2	Jinja2 template for functional descriptions. The template accepts a parameter request with optional fields: comment_doc the molecular description of the protein collected from <comment> fields and dbrefs: a list of dictionaries containing the label codes, definitions, and names of each <dbReference> field. . . . .	93
C.3	Concept conditional distributional conformity scores for a selected subset of description embedding neighborhoods. . . . .	94
C.4	Histogram of embedding distances. . . . .	98
C.5	Annotation coverage distributions across a random subsample (10%) of the scaled training dataset. <i>Left.</i> The distribution of percent coverages per sequence (covered are counted once per sequence). <i>Right.</i> The distribution of percent coverage when we repeat the count of residues participating in multiple annotations. . . .	103
C.6	<i>Left.</i> VKOR1 case study on task specific inference. <i>Right.</i> AlphaFold 2 [3] predicted structure of VKOR1. Highly conserved functional cysteine residues are shaded in blue. <b>(A, D)</b> Position specific perplexities for CORGI and the baseline (orange) averaged over abundance <b>(A)</b> and activity <b>(D)</b> phenotypes. Shaded regions represent trans-membrane domains and dashed lines represent positions of conserved, functional cysteine residues. <b>(B, C)</b> Logit standard deviations across abundance phenotypes for CORGI-greedy <b>(B)</b> and the baseline <b>(C)</b> . Positions with higher variance are shaded with more intensity. <b>(E, F)</b> Softmax probabilities calculated from activity phenotypes for CORGI-all <b>(E)</b> and the baseline <b>(F)</b> . Positions with higher probability mass are shaded with more intensity. Black boxes represent the softmax probability of conserved cysteines. . . . .	106
C.7	Visualizations of conformity over a long chain of sampling and the highest ranking.	108

# List of Tables

3.1	Statistics of the distance matrix, $\chi$ , for discrete antibody sequence data. . . . .	26
3.2	Ablikeness metrics, uniqueness, diversity, and distributional conformity scores. .	29
3.3	Measured protein synthesis. . . . .	29
3.4	Predicted and measured antibody binding affinity. . . . .	31
4.1	Example outputs of multi-segment preserving sampling when restricting variation to the CDR3 region under different settings of $\beta$ . Display is restricted to the sampled region, the rest is preserved by construction. . . . .	50

5.1	Performance comparison of the baseline to each CORGI model and setting and averaged per (selection type, mutation type) pair. The columns <b>Model</b> and <b>Setting</b> indicate the underlying CORGI model and inference setting. The acronyms MLM and NEB stand for Masked Language Model and Neural Empirical Bayes respectively. “Baseline” stands for the masked language model trained as a control without annotations on the same dataset as the other CORGI models. The abbreviations “R”, “G”, and $\emptyset$ designate whether residue-level, global, or no annotation features were provided at inference time. Each column ( <b>Expression</b> , <b>Organismal Fitness</b> , <b>Stability</b> , <b>Activity</b> , and <b>Binding</b> ) represents the DMS selection type and is followed by the mutation type. “Indels” stand for the edits that change the original sequence length by at least one insertion or deletion. “Subs” represent substitutions, or mutations that do not change the original sequence length. . . . .	64
5.2	Performance comparison of the baseline to each CORGI model and setting and averaged per (taxon, mutation type) pair. Each column ( <b>Eukaryote</b> , <b>Human</b> , <b>Prokaryote</b> , and <b>Virus</b> ) stand for the taxonomic classification of each wildtype provided by the ProteinGym. See Table 5.2 for a complete description of the <b>Model</b> and <b>Setting</b> columns as well as the abbreviations therein. . . . .	65
A.1	Smoothed Discrete Sampling implementations. . . . .	73
A.2	$\sigma$ , ablikeness metrics, uniqueness, diversity for score-based dWJS. . . . .	76
A.3	Model size, sampling time and memory consumption. . . . .	79
C.1	Sampling parameters for molecular programming experiments . . . . .	95
C.2	Summary statistics of elements of the distance matrix $\chi$ , for 4096 50% sequence identity cluster representatives. . . . .	98

C.3	Performance comparison for different CORGI settings and the baseline averaged per (selection type, mutation type) pair. . . . .	104
C.4	Performance comparison for different CORGI settings and the baseline averaged per group of (mutation-type, taxon division) pairs. . . . .	105

# List of Appendices

A	Supplementary Material For Chapter 3	72
B	Supplementary Material For Chapter 4	86
C	Supplementary Material For Chapter 5	91

# 1 | INTRODUCTION

Proteins are the functional products of genetic sequences such as DNA that either carry out or engage in almost every biosynthetic process in all cellular organisms. Proteins are functionally diverse and ubiquitous throughout nature, making them ideal candidates for many applications in subject areas such as drug discovery [4], agriculture [5], and the environment [6]. Despite the potential benefits of protein design, multiple challenges arise from the combinatorially vast sequence and sparsely functional sequence space. Moreover, even subtle differences in protein sequences can result in gain or loss of protein function, making the design process extremely sensitive to small perturbations. Computational protein design seeks to efficiently sample sequence space and generate on-target, physically realizable proteins.

Advances in next-generation sequencing technologies have given way to large-scale genomics projects [7–10] and resulted in a proverbial biological data age. Efforts to characterize and annotate protein sequences [11–13] have created a rich taxonomies of protein structure and function. Data availability combined with specialized semiconductors has enabled the application of emerging methodologies in artificial intelligence such as deep learning. In particular deep neural networks, optimized via stochastic gradient descent [14, 15], have been shown to model complex distributions without the need for feature-engineering or dimensionality reduction techniques [16, 17]. Self-supervised learning [15, 18] and generative modeling [19–21] have not only disentangled the need for an output label, but also opened the possibility of generating novel samples.

A complete computational protein design workflow entails candidate generation, sample ranking and evaluation, experimental readout, and in some cases re-incorporation of experimental feedback [22]. This work is primarily concerned with the former two. We address the following objectives: protein family diversification, steerable generation, and sample evaluation.

In Chapter 3, we investigate the utility of generative models to diversify the variable domain of antibody heavy and light chains. In order to do so, we introduce the discrete Walk-Jump Sampler (dWJS), a generative model based on the Neural Empirical Bayes (NEB) [20] framework. The Neural Empirical Bayes framework relies on Herbert Robbins' seminal *An empirical Bayes approach to statistics* [23] by training a Bayesian estimator parameterized by a neural network to estimate the score of a data manifold corrupted with additive isotropic Gaussian noise. The dWJS has two variants applicable to sampling procedures such as stochastic gradient Langevin dynamics [24]: an energy-based model (EBM) trained via contrastive divergence [25] and a score-based model trained using a form of denoising score matching. To demonstrate the capabilities of the dWJS for protein family diversification, we first train it on the paired heavy and light chains of the Observed Antibody Space (pOAS) [2] and generate novel antibodies meant to target the HER2 receptor. We evaluate the generative model *in-silico* with developing the Distributional Conformity Score (DCS), an instantiation of a conformal transducer system (CTS) [26, 27], which returns a randomized  $p$ -value testing the null hypothesis that a query sample is drawn independently and identically distributed (IID) with respect to some reference distribution. We find that both variants of the dWJS produce realistic antibody sequences with conserved framework regions and mutated the complementarity determining regions (CDRs). We experimentally evaluated the top ranking antibody samples and found they not only expressed *in-vitro*, but also bound to the target antigen and outperformed other existing methods in doing so.

In Chapter 4, we develop the deep manifold sampler (DMS), a sequence denoising autoencoder as well as two sampling methodologies to steer its generation. The DMS is a non-autoregressive transformer encoder-decoder that accepts a corrupted, off manifold, sequence and denoises it to

its original state. The corruption process perturbs the original sequence by either inserting, deleting, or substituting  $\Delta L$  residues. In order to handle the length differences between the corrupted and original sequence, the DMS follows Shu, et al [28] and is equipped with two learnable modules conditioned on the encoder representation of the corrupted sequence  $\tilde{Z}$ : length posterior module  $P_\eta(\Delta L \mid \tilde{Z})$  that classifies the length change and a length transformation  $f_\sigma(Z \mid \tilde{Z}, \Delta L)$  that expands or contracts the input by  $\Delta L$ . Using an MCMC-like iterative denoising process, the DMS samples a wider range of proteins by not only sampling sequences from the decoder, but also estimating length changes  $\Delta L \in [-\Delta L_{\max}, \Delta L_{\max}]$  using the length posterior. Additionally, we propose two methods for conditional generation using the DMS: classifier-guided, function biased sampling and classifier-free multi-segment preserving sampling (MSPS). In function-biased sampling, we train an external classifier  $P_\omega(Y \mid \tilde{Z})$  to classify gene ontology (GO) terms [11] given the corrupted sequence representation. During sampling, we add the gradient of the classifier with respect to the function of interest  $\nabla_{y \in Y} P_\omega(Y \mid \cdot)$  to the decoded, length transformed representation  $Z$ . The DMS demonstrates the capability to produce target-function enriched samples using this guidance methodology as well as traverse between distinct functional families. Multi-segment preserving sampling is an alternative guidance methodology that injects expert information into the sampling process in order to concentrate sampling to user-defined, non-preserved regions. In order to do so, we modify the corruption process at inference time to apply perturbations along the non-preserved segments. In turn, we modify the length posterior to distribute length changes in those segments while retaining absolute sequence identity in the preserved regions to the original seed sequence. We demonstrate the ability of the DMS to constrain sampling to CDR regions in antibodies and produce realistic, i.e., high-likelihood samples. These two conditional sampling methodologies described above result in a balanced approach to exploratory sampling while exploiting external information.

In Chapter 5, we explore new directions in natural language conditioned protein design. Natural language communication with molecular machine learning systems, or molecular program-

ming (MP), offers a flexible interface for concisely articulating the desired properties of the target molecules. Previously established molecular programming methods rely on either contrastive alignment to learn a joint embedding space of protein sequences and text [29–31] or as effective translation systems between text and protein [32]. Unfortunately, evaluation approaches for MP systems is not yet standardized, due partially to the ill-defined nature of text-based molecular concepts.

In this chapter, we describe two alternative MP systems condition on text embeddings rather than natural language itself. The advantages of embedding-based conditioning are not only based in J. R. Firth’s distributional hypothesis: “A word is characterized by the company it keeps”, but also in the emergent additive nature of semantic embeddings that enables operations such as concept algebra [17]. The first approach, interrogates the utility of text conditioning for mutation effect prediction. We describe a general protein language model fine-tuning method, Conditioning on Residue-level Annotations from Text (CRATE), that allows for the injection of conditioning information, in particular token-level annotations such as binding and active sites, during training. We benchmark CRATE on the ProteinGym [33], a dataset of deep mutational scan (DMS) assays surveying a variety of selection types and protein functions. We find that not only does CRATE-training results in improved performance compared to the baseline unconditional model, but also it enables task-specific conditioning in order to improve performance on a per-assay basis. In the second approach, we apply the same embedding conditioning methodology as in CRATE to train a text-conditioned Walk-Jump sampler (tcWJS) module. Specifically, we fine-tune ESM2 using the NEB approach with the additional input of text embeddings at each hidden layer. In order to evaluate the tcWJS against a state-of-the-art baseline, we use two portable evaluations to test validity, whether the output sequence resembles a realistic protein, and language alignment, the coherence of the output molecule with the conditioning text. To test validity, we draw from the evaluations developed in Chapter 4 by measuring the pseudo-likelihood of generated samples given by a surrogate model relative to that of putative proteins in SwissProt [34]. To test lan-

guage alignment, we build off of the distributional conformity score developed in Chapter 3 to instantiate a new CTS called the concept-conditional distributional conformity score (ccDCS), which reports a *p*-value testing whether a given sample is drawn from a reference distribution of proteins annotated with similar text to the input. We find that tcWJS is competitive with the state-of-the-art model MP4 in both language alignment and validity.

The rest of this dissertation is organized in the following manner. Chapter 2 provides background information on protein biology, historical context on protein design, descriptions of necessary bioinformatics tools and databases, and a brief literature review protein sequence modeling and generation. Chapter 3 is a reprint of the published work describing the discrete Walk-Jump sampler. Chapter 4 contains two reprinted articles, the first introducing the Deep Manifold Sampler and function-guided design and the second describing the multi-segment preserving sampling algorithm. Chapter 5 introduces the CRATE fine-tuning method and describes the text-conditioned Walk-Jump sampler. We conclude in Chapter 6 with a summary of the work as well as a description of future directions and outlook.

## 2 | BACKGROUND MATERIAL

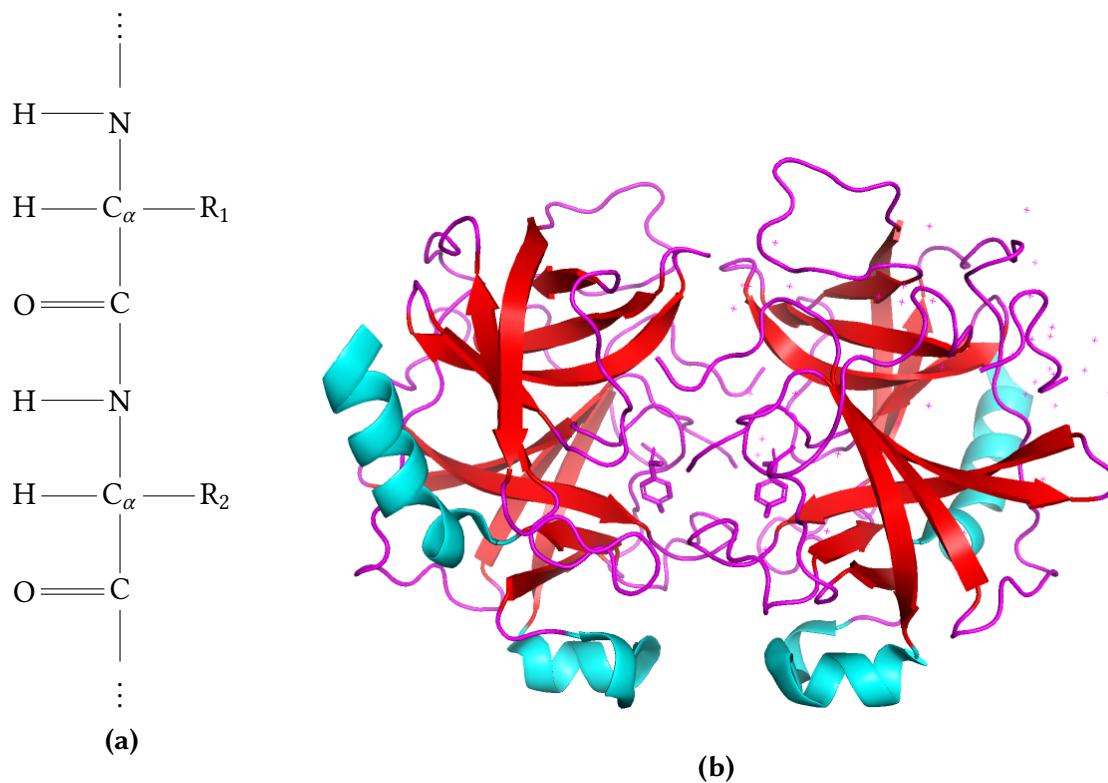
This chapter primes the reader for the following chapters by providing essential context on topics related to computational protein design. In Section 2.1, we discuss the necessary basis for protein biology as well as computational tools for protein sequence clustering and categorization. In Section 2.4, we review modern methods for modeling and sampling protein sequences.

### 2.1 PROTEIN BIOLOGY, STRUCTURE, AND FUNCTION

Proteins are a diverse class of large, organic molecules that participate in almost every biological process in every living organism. These molecular machines are linear polypeptide chains composed of amino acids, and highly specialized to carry out specific functions such as catalysis of reactions, signaling to other systems, and mechanical interaction with other biomolecules (often proteins) to create materials or other products. Proteins are located all throughout the organismal system in and outside of cells. Intracellular proteins are found in different cellular organelles and expressed to implement the functionality of the cell. Extracellular proteins are located outside of the cell membrane and may act by signaling to other cells or biological subsystems. Transmembrane proteins act as an intermediate conduit between either side of the cell membrane. This section provides necessary details to contextualize the protein modality for computer scientists.

### 2.1.1 MOLECULAR COMPOSITION OF PROTEINS

Proteins are linear chains of amino acids, or residues. Amino acids are small molecules with three functional groups: an amino group, a carboxyl group, and a variable side chain that confers its electro-physical properties. There are twenty naturally occurring amino acids, each with different molecular properties.



**Figure 2.1:** 2.1(a): Diagram of a two amino acids joined by a peptide bond. Each  $R$ -group represents a side-chain. 2.1(b): Quaternary structure of a crystallized Chymotrypsin protein (PDB ID: 2HCA). Loops,  $\alpha$ -helices, and  $\beta$ -strands are colored in magenta, cyan, and red respectively.

Consecutive amino acids along a protein sequence are joined via a peptide bond, as seen in Figure 2.1(a), that connect the carboxyl and amino groups, forming the protein backbone.

Local force-fields induced by the molecular properties of adjacent amino acids along preference short subsequences to take on certain geometries, or secondary structures:  $\alpha$ -helices,  $\beta$ -strands, and loosely structured loop regions. These secondary structure elements act on one

another forcing the protein to fold into an energetically favorable conformation called the tertiary structure. The tertiary structure may consist of one or more constituent domains which may themselves play different functional roles [35].

Finally, tertiary structures may fold in complex with one another to construct multimeric assemblies called quaternary structure (see Figure 2.1(b)).

#### 2.1.1.1 SEQUENCE HOMOLOGY

Evolution is the product of random recombination and variation of ancestral genetic code subject to environmental selective pressure. As organisms evolve, mutations in genetic code results in downstream functional speciation of biological products such as proteins. Homologous proteins arise from two different speciation events: splitting and duplication [36, 37]. In a splitting event, gene sequences diverge and although non-identical, they code for functionally equivalent protein products, or orthologs (for example, horse and human myoglobin). In a duplication event, a gene sequence is duplicated with variation, creating a paralog that may evolve to perform a different but related function (for example, horse myoglobin and horse hemoglobin).

## 2.2 EARLY METHODS IN PROTEIN DESIGN

Engineering proteins is a long sought after grand challenge. Early pioneers proposed “rational design” – the systematic modification of protein sequences based on hypotheses on the relationship between protein structure and function [38, 39]. Contemporaries proposed site-directed mutagenesis [40, 41] to methodically test hypotheses. Arnold’s seminal work on directed evolution [42] proposed mimicking natural selection by evaluating the fitness of random mutations along a given sequence. Despite these early methods setting the stage for modern protein engineering and design, the throughput is limited by costs, in money and in time, of the experimental procedures. Later, Computational models of protein folding such as Rosetta [43, 44] were used

to optimize sequences by inverse folding resulted in the first *de novo* designed protein [45]. Still, the computational cost of mechanistic modeling techniques as well necessitate faster, more data-driven methods for general design problems.

## 2.3 BIOINFORMATICS TOOLS

Proteins are large and complex biomolecules with a diverse species of functions and properties. In order to better understand the sequence, structure, and function landscapes, a variety of tools have been developed to find patterns, cluster, and label proteins for downstream analysis.

### 2.3.1 SEQUENCE CLUSTERING

As discussed in Section 2.1.1.1, homologous proteins may be functionally similar as well. Sequence similarity-based clustering methods group samples using a sequence similarity kernel. Typically, clustering involves setting a similarity threshold, finding diverse cluster centroids whose similarity does not exceed the threshold, and assigning membership by finding the centroid of maximal similarity. Sequence similarity is found by aligning sequences – matching preserved domains according to a transition matrix such as BLOSUM62 [46] and offsetting with gap characters – using dynamic programming algorithms such as Smith-Waterman or Needleman-Wunsch.

Fast tools for clustering such as MMSeqs2 [47] enables agglomeration of large sequence databases and downstream applications such as search/retrieval (linear scan of centroids using e.g., BLAST [48]), annotation transfer (copying labels among cluster members), and dataset debiasing (sampling ecumenically from clusters during e.g., model training). The UniRef databases [49] are a collection of non-redundant sequence cluster databases.

Sequence clusters are commonly postulated to perform similar functions and hence a variety of classification systems have been developed to categorize clusters as well as provide a taxonomy of their functions.

### 2.3.2 DATABASES AND ANNOTATION TYPES

Protein sequences are deposited into central databases such as UniProt [50] with either experimental evidence from large-scale sequencing initiatives [7–10, 51] or as hypothetical protein predictions from open reading frames (ORFs) [52]. The UniProt database is divided into separate sources: SwissProt [53], a manually curated database of richly annotated proteins and UniProt-TrEMBL, a larger scale database with electronic curation and annotations.

Advancements in experimental technologies [54] have scaled sequence repositories to big data and enabled use of modern deep learning methods. Sequence data can be retrieved as strings of amino acids following the standard coding scheme of the International Union of Pure and Applied Chemistry [55], which account for ambiguous and nonstandard amino acids.

#### 2.3.2.1 ANNOTATIONS

Protein sequences are often accompanied with either experimentally verified or electronic annotations meant to characterize their structure and function. For the purpose of this work, annotations may appear in three different forms:

1. Labels consisting of a record identity number and optionally a natural language description.
2. Numeric molecular properties, either computationally derived or experimentally confirmed.
3. Free-form, natural language descriptions.

Annotations of this form may apply to the entire protein sequence (global) or ascribed to a specific subsequence or domain (residue-level).

We review a routinely used computational annotation sources below.

- *Pfam* [56] uses cluster-specific profile hidden markov models (pHMMs) to assign sequence labels. Clusters are interpreted and labeled by a mix of automation and expert evaluation to ascribe a semantic meaning.

- *CATH* [13] is a structural taxonomy of protein domains wherein each successive tier of the system constitutes a more specific fold classification. The acronym stands for *Class*, *Architecture*, *Topology*, and *Homologous superfamily*. All tiers besides the last are assigned using a structural, rather than sequence, similarity kernel.
- *InterPro* [57] combines annotations from several sources including Pfam and CATH. InterPro annotations are assigned by ensembling the predictions from each source and unifying them under a common label system.

Each of the above listed annotation sources is linked to separate function classification systems that seek to provide a taxonomy of protein function. The Gene Ontology (GO) [11] is a directed acyclic graph (DAG) of three disjoint components standing for different aspects of function: Molecular function (MF), biological process (BP), and cellular component (CC). Each node of the separate components is connected to its eponymous root. Traversal along the path towards the root represents synonymous generalization.

## 2.4 PROTEIN SEQUENCE MODELING

The most abundant form of protein data are amino acid sequences. The relationship between sequence, structure, and function (see Section 2.1.1) implies that sequences contain a great deal of information on the function of the protein that can be used for modeling and prediction.

Both protein sequences and natural language text are modalities of discrete tokens with complex, often long-range dependencies. Consequentially, innovations in natural language processing (NLP) are often adapted in bioinformatics.

### 2.4.1 MATHEMATICAL DESCRIPTION

Formally, let  $\mathcal{V}$  be a finite vocabulary of discrete tokens  $v_1, v_2, \dots, v_{|\mathcal{V}|}$  and let  $X = [x_t]_{t=1}^\ell$  be a sequence of length  $\ell$  where  $x_t \in \mathcal{V}$ . Sequences are drawn with respect to a density model  $X \sim P(X) = P(x_1, x_2, \dots, x_\ell)$ . This work considers two factorizations of  $P(X)$ . The autoregressive, or causal factorization of  $P(X)$  is

$$P(X) = \prod_{t=1}^{\ell} P(x_t | x_{<t}) \quad (2.1)$$

where  $x_{<t} = [x_k]_{k=1}^{t-1}$ . Models trained with the autoregressive formulation be readily sampled by drawing tokens from  $P(x_t | x_{<t})$  until reaching a prescribed stopping criteria. The non-autoregressive, or bidirectional factorization of  $P(X)$  is

$$P(X) = \prod_{t=1}^{\ell} P(x_t | x_{\setminus t}) \quad (2.2)$$

where  $x_{\setminus t}$  represents all other tokens in the sequence apart from  $x_t$ . Models trained in a non-autoregressive manner cannot be readily sampled without a prescribed procedure. No matter the factorization, models of both types are likelihood-based models in that they learn to maximize the likelihood (or pseudo-likelihood in the case of bidirectional models) of a collection of observed sequences. Bidirectional models, such as masked language models (MLMs), typically learn the pseudo-likelihood via a denoising objective [58, 59].

Yet another formulation of  $P(X)$  expresses it as a Boltzmann distribution

$$P(X) = \frac{\exp(-\beta E(X))}{Z} \quad (2.3)$$

where  $Z = \sum_{X \in \mathcal{V}^*} \exp(-\beta E(X))$ . Energy-based models (EBMs)  $E_\theta(X)$  estimate the unnormalized log-probability  $E(X)$  and can be trained by minimizing, e.g., contrastive divergence (CD) [60].

Sampling using an energy-based model with a derivative (as is the case for CD-based models) can be implemented by stochastic gradient Langevin dynamics (SGLD) [24, 61]. Score-based generative models [21, 62, 63] such as discrete diffusion models [64, 65] represent the score, or gradient of the log-density, and are similarly applicable to SGLD.

#### 2.4.1.1 GENERATIVE MODELS FOR SEQUENCE DESIGN

Protein generation is a core module in the design workflow wherein novel sequences are sampled according to the learned density model. For autoregressive models, sampling is readily available by decoding left-to-right, generating one token at a time. Pseudo-likelihood models such as MLMs typically use an MCMC-like procedure by starting with a initial, possibly random, sequence and iteratively masking/corrupting and denoising it to approximate the joint density in Equation 2.2. Models of or with explicit access to the score of a sample are applicable to sampling schemes such as SGLD. Techniques to steer generation of these models can also be implemented by external (classifier-based) or internal (classifier-free) guidance. In classifier-guided generation [66–68], the gradient of an external classifier with respect to a class of instance is added as a supplementary signal during sampling. In classifier-free guidance [69], the model itself is made to represent the joint distribution of data and properties by, e.g., prepending class tokens to the input.

#### 2.4.2 HISTORICAL CONTEXT OF BIOLOGICAL SEQUENCE MODELING

Early methods adapted Shannon’s  $n$ -gram model [70] to biological sequence by analyzing  $k$ -length subsequences called  $k$ -mers s[71–73]. In practice, such fixed-length prefix methods use a limited context window that does not allow the model to approximate the entire context, require storing a large probability table of size  $|\mathcal{V}|^{k-1} \times |\mathcal{V}|$ , and call for heuristics to approximate  $k$ -mers not found in the training set.

Hidden Markov Models (HMMs) for text [74, 75] brought on innovations in bioinformat-

ics[76–78]. Mathematically, HMMs defer from the  $n$ -gram model by leveraging Markovian hidden states and estimating observable token emission probabilities. However, these models rely on restrictive assumptions about the dependency structure of the sequence and require a computationally costly sequence alignment, which may even still under-specify the distribution of interest if it is too shallow. Additionally, while HMMs are still in wide use today for classification (see Section 2.3.2.1), the primary task in protein design is sequence generation, where modern methods vastly outperform them.

Deep learning has since offered a powerful framework for sequence recognition that alleviates many of the shortcomings of preceding methods. One-dimensional convolutional networks learn localized spectral filters intuitively similar to  $k$ -mers have demonstrated competitive performance in protein function prediction [79] as well as general sequence modeling [80]. Recurrent and gated neural networks [81–83] demonstrated improved performance over HMMs and capability to learn powerful representations for downstream tasks.

#### 2.4.3 TRANSFORMER REVOLUTION

Prior to the seminal work describing the transformer architecture [84], gated neural networks were the state-of-the-art in protein sequence modeling. However, transformer-based protein foundation models [85–87] have since demonstrated improved performance over gated neural networks on a variety of axes including representational capacity, longer context windows, interpretability, and training speed and stability.

- *Representational capacity.* Transformer models have shown to outpace gated networks in scaling of model size [83, 88].
- *Longer context windows.* Gated networks transmit information at each step which may filter out dependencies over very long time-scales. In contrast, the self-attention mechanism in transformers not only parallelizes the information sharing step but allows it to be shared

anywhere along the sequence. Moreover, memory efficient implementations of the attention mechanism [89] enable subquadratic storage complexity that allows longer contexts to be captured.

- *Interpretability.* The attention maps of a transformer network are more easily understood than the gating and memory mechanisms of gated units [90]. Another recent line of work applies mechanistic interpretation using sparse autoencoders to protein language models [91–93].
- *Training speed and stability.* Contrary to gated networks, even autoregressive transformer-based models do not suffer from a sequential computation bottleneck by virtue of the attention mask. Transformers also do not suffer from common issues during training such as exploding and vanishing gradients.

#### 2.4.3.1 INDUCTIVE BIAS AND MASKING

The transformer attention mechanism calculates pairwise attention weights along the sequence, imposing no prior on the dependency structure. Inductive biases may be asserted onto the model by masking the attention maps.

Causal (autoregressive) masks are lower triangular matrices that disallow attention to pass backwards in time (i.e., future tokens cannot inform the representation of their predecessors). A priori, causal masking may not seem like the ideal choice for protein sequence modeling as the N-to-C terminus ordering is arbitrary from a sequence modeling perspective. However, autoregressive pLMs such as ProGen [87, 94] have demonstrated state-of-the-art generation by training with N-to-C and C-to-N terminus ordering and labeling the ordering with specialized start/end tokens. Causal language models conform to the formulation in [subsection 2.4.1](#) and can be readily sampled by iteratively generating sequences token-wise.

Masked language models (MLMs) [58, 86] process the input sequence without imposing an

information flow order. During training, random subsets of tokens are masked and reconstructed by the model using the unmasked context. Models are optimized to reconstructing masked tokens given the surrounding information. As such, masked language models are not restricted to a specific direction information flow. Span masking [95] is a variant of masked language modeling in which contiguous token spans are dropped out. Masked language models impose an unordered dependency structure that enables past and future tokens to pass information. While the inductive bias of a masked language model aligns more closely with the dependency structure along a protein sequence, sequences from such models cannot be readily sampled without a decoding heuristic or iterative sampling procedure.

Although in natural language the choice of tokenizer has been shown to affect performance [96, 97], protein language models (excluding e.g., ProtGPT2 [98]) often use a simple amino acid level tokenizer to enable complete control over the output and force the model to learn the multi-token dependencies. A recent line of work however has investigated the utility of other tokenization strategies [99] as well as the ability to ingest other protein modalities such as structure tokens [100, 101].

## 3 | DISCRETE WALK JUMP SAMPLER

This chapter is a reprint of published paper:

Protein Discovery with Discrete Walk-Jump Sampling

Nathan Frey<sup>†</sup>, **Daniel Berenberg**<sup>†</sup>, Karina Zadorozhny, Joseph Kleinhenz, Julien Lafrance-Vanasse, Isidro Hotzel, Yan Wu, Stephen Ra, Richard Bonneau, Kyunghyun Cho, Andreas Loukas, Vladimir Gligorijević, and Saeed Saremi *International Conference on Learning Representations (ICLR) 2024*

<sup>†</sup> These authors contributed equally to the reproduced part in this thesis.

### 3.1 INTRODUCTION

Discrete sequence generation poses a number of challenges to gradient-based generative models. Generative models must be expressive enough to faithfully capture the underlying data distribution, while also having controllable outputs that are novel, unique, diverse, and respect the constraints of the problem space. Energy-based models (EBMs) [102, 103] fit an energy function that specifies a probability distribution over data analogous to the Boltzmann distribution from statistical physics. Giving access to an easily computable energy is an advantage of EBMs, but on the flip-side they can be difficult to train and sample from. Denoising objectives based on score matching [62, 104] and the related advancements in diffusion models [105, 106] overcome these issues, but these either model the energy gradient or only provide access to an empirical

lower-bound of the likelihood.

Protein design is an instance of the discrete sequence generation problem, wherein the challenge is to find useful proteins in the large, discrete, and sparsely functional space Romero and Arnold [107] of dimension  $20^L$  for proteins of length  $L$ . Here, we consider the specific problem of generative modeling of antibodies, a class of proteins with highly conserved structure that are of immense interest for therapeutics. In addition to the qualities mentioned above, generative models for antibodies must be sample-efficient because of the relatively small size of datasets with therapeutic antibodies Kim et al. [108]. Antibodies consist of well-conserved domains and high-entropy variable regions, so leveraging evolutionary information from pre-trained protein language models is not an immediate solution. We distinguish *ab initio* protein discovery and design (producing novel, functional proteins given some training samples), which is the focus of this work, from *de novo* design, which we define as the generation of novel proteins without starting material. Existing autoregressive protein design methods Jin et al. [109] are inefficient and can suffer from accumulation of errors and high inference latency, while current non-autoregressive diffusion models are similarly inefficient and poorly optimized for real discovery and design tasks Kong, Huang, and Liu [110]. Our goal here is to invent an efficient, non-autoregressive generative modeling paradigm for discrete data that produces high quality, novel samples.

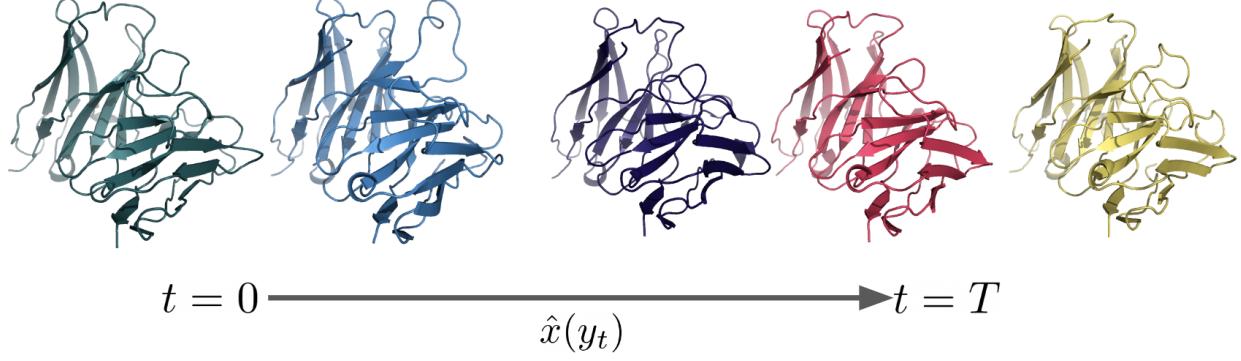
To this end, we introduce **Smoothed Discrete Sampling (SDS)**, a new formalism for training and sampling from discrete generative models. We propose a novel algorithm, **discrete Walk-Jump Sampling (dWJS)**, a method building on the neural empirical Bayes (NEB) [20] formalism, that addresses the brittleness of discrete EBMs and diffusion models and in doing so, provides a robust and general framework for protein discovery and design.<sup>1</sup> We also design a metric called the **Distributional Conformity Score (DCS)**, which is a simple scalar score for protein sample quality. Our results rescue EBMs for discrete distribution modeling and question the need for diffusion models with multiple noise scales in protein discovery.

---

<sup>1</sup><https://github.com/prescient-design/walk-jump>

Our contributions are as follows:

- We introduce a new paradigm for modeling discrete data distributions, *Smoothed Discrete Sampling* (SDS), building on the neural empirical Bayes framework. We propose the discrete Walk-Jump sampling algorithm, which uses uncoupled, separately trained score- and energy-based models to learn *noisy* data distributions and sample discrete data. dWJS enables fast, non-autoregressive sampling with variable length discrete outputs. We also design a novel architecture for discrete EBMs.
- Our method simplifies score-based model training for discrete data by requiring only a single noise level and no noise schedule, which alleviates the brittleness, training instabilities, and slow sampling of diffusion models. Our method also resolves difficulties in training EBMs, obviating the need for many common EBM training tricks (replay buffer,  $\ell_2$  norm penalty, rejection sampling, etc.), while preserving good sample quality and fast sampling.
- We demonstrate the utility of our approach in the context of *ab initio* protein discovery and design - generating novel, biophysically-valid protein sequences from models trained on repertoires of functional molecules. We validate our method with *in vitro* experiments. Our method outperforms autoregressive and masked protein language models, large language models, discrete sequence and structure-sequence diffusion, and score-based baselines.



**Figure 3.1:** Selected samples from a single Markov chain Monte Carlo sampling run of discrete Walk-Jump sampling (our method). Protein color corresponds to different antibody germlines (classes). Samples are folded with EquiFold [1] for visualization purposes. Discrete walk-jump sampling exhibits fast mixing and explores diverse modes of the distribution in a single chain.

## 3.2 BACKGROUND

### 3.2.1 ENERGY-BASED MODELS

EBMs are a class of models that learn an energy function  $f_\theta : \mathcal{X} \rightarrow \mathbb{R}$  mapping inputs  $x$  (in  $\mathbb{R}^d$ ) to a scalar “energy” value. The data distribution is approximated by the Boltzmann distribution

$$p_\theta(x) \propto e^{-f_\theta(x)}. \quad (3.1)$$

EBMs are typically trained via contrastive divergence [25], and new samples are drawn from  $p_\theta(x)$  by Markov-Chain Monte Carlo (MCMC). Details of the loss function used in this work are given in Section 3.3. In Langevin MCMC, samples are initialized from a known data point or random noise and refined with (discretized) Langevin diffusion

$$x_{k+1} = x_k - \delta \nabla f_\theta(x_k) + \sqrt{2\delta} \varepsilon_k, \quad \varepsilon_k \sim \mathcal{N}(0, I_d), \quad (3.2)$$

where  $\nabla$  denotes the gradient of the energy function with respect to inputs,  $k$  is the sampling step,  $\delta$  is the (discretization) step size, and the noise  $\varepsilon_k$  is drawn from the normal distribution at

each step.

### 3.2.2 NEURAL EMPIRICAL BAYES

In NEB, the random variable  $X$  is transformed with additive Gaussian noise  $Y = X + \mathcal{N}(0, \sigma^2 I_d)$ .

The least-squares estimator of  $X$  given  $Y = y$  is given by [23, 111]

$$\hat{x}(y) = y + \sigma^2 \nabla \log p(y), \quad (3.3)$$

where  $p(y) = \int p(y|x)p(x)dx$  is the probability distribution function of the smoothed density.<sup>2</sup>

This estimator is often expressed directly in terms of  $g(y) = \nabla \log p(y)$  known as the score function [62] which is parameterized with a neural network denoted by  $g_\phi : \mathbb{R}^d \rightarrow \mathbb{R}^d$ . The least-squares estimator then takes the following parametric form:

$$\hat{x}_\phi(y) = y + \sigma^2 g_\phi(y). \quad (3.4)$$

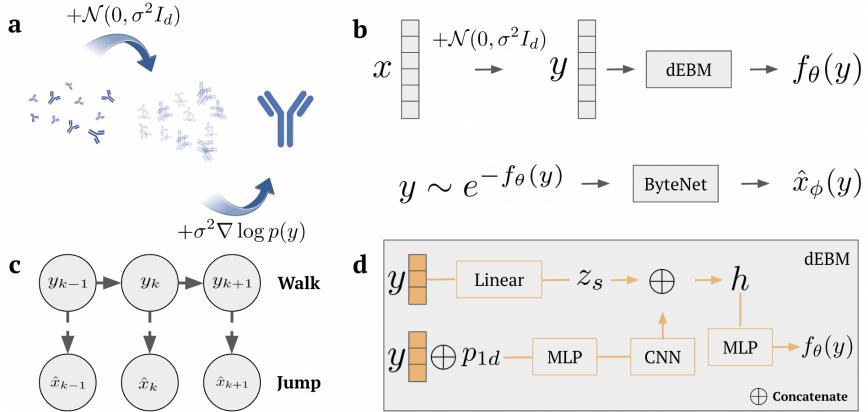
Putting this all together leads to the following learning objective

$$\mathcal{L}(\phi) = \mathbb{E}_{x \sim p(x), y \sim p(y|x)} \|x - \hat{x}_\phi(y)\|^2, \quad (3.5)$$

which is optimized with stochastic gradient descent. Notably, no MCMC sampling is required during learning. In short, the objective is “learning to denoise” with an empirical Bayes formulation (discussed further in Appendix A.2.3). Additionally, the NEB approach conveniently requires a single hyperparameter  $\sigma$  controlling the smoothness of the distribution  $P(Y)$ . We estimate the optimal  $\sigma$  in Appendix A.1.5.

---

<sup>2</sup>We follow the convention  $p(x) \doteq p_X(x)$ ,  $p(y) \doteq p_Y(y)$ , etc.



**Figure 3.2:** Discrete walk-jump sampling. **a** The noising and denoising process is applied to antibody proteins. **b** Discrete inputs  $x$  are smoothed with isotropic Gaussian noise,  $\varepsilon \sim \mathcal{N}(0, \sigma^2 I_d)$ , to noisy inputs,  $y = x + \varepsilon$ . A discrete energy-based model (dEBM) parameterizes the energy function  $f_\theta(y)$  of noisy data. Noisy data is sampled with the energy function, and denoised with a separate denoising ByteNet network to clean samples,  $\hat{x}_\phi(y)$ . **c** The “walk” sampling steps on the noisy data manifold with Langevin MCMC are totally decoupled from the “jump” steps to clean samples. **d** The dEBM takes noisy inputs  $y$ , concatenates them with a 1d positional encoding,  $p_{1d}$ , passes through an MLP and a 3 layer CNN, and concatenates the outputs with an embedding  $z_s$  of the inputs into a hidden state,  $h$ .  $h$  is passed through an MLP and returns the energy  $f_\theta(y)$ .

### 3.3 ANTIBODY DISCOVERY AND DESIGN

#### 3.3.1 DISCRETE WALK-JUMP SAMPLING

Following training of the denoising network,  $g_\phi$ , one can sample noisy data using the learned score function  $g_\phi(y)$  with Langevin MCMC (replace  $-\nabla f$  with  $g$  in Equation 3.2). For any such draws  $y_k$ , clean samples from the true data manifold  $\mathcal{M}$  are obtained by “jumping” back to  $\mathcal{M}$  with the least-squares estimator  $\hat{x}_\phi(y_k) = y_k + \sigma^2 g_\phi(y_k)$ . This is the walk-jump sampling (WJS) scheme. A key property of WJS is the fact that the least-squares estimation (jump) is *decoupled* from the Langevin MCMC (walk).

Here, we take advantage of this decoupling to train an EBM with maximum likelihood estimation on the smoothed distribution of noisy sequences, generate noisy samples with Langevin MCMC, and denoise samples with a separately trained neural network, the least-squares estima-

tor. The algorithm for discrete walk-jump sampling is given in Algo. 1. Our algorithm is general and applies to any discrete sequence inputs of a fixed vocabulary. In Fig. 3.1 we show samples generated from a single chain of MCMC. Unlike a diffusion model, every sample along the chain collectively forms a valid set of samples from the underlying distribution, because of the decoupled walk (sampling) and jump (denoising) steps. dWJS also produces fast-mixing chains, such that many diverse modes (protein classes) are sampled in a single chain. Samples are folded with EquiFold [1] for visualization and confirmation of structural validity.

The EBM is trained by maximizing the log-likelihood of *noisy* data under the model:

$$\arg \max_{\theta} \mathbb{E}_{y \sim p_Y} [\log p_{\theta}(y)] = \arg \max_{\theta} (\mathbb{E}_{y^- \sim p_{\theta}(y)} [f_{\theta}(y^-)] - \mathbb{E}_{y^+ \sim p_Y} [f_{\theta}(y^+)]), \quad (3.6)$$

where  $y^+$  are noisy training data and  $y^-$  are noisy data sampled from the model.

With this objective, the model aims to decrease the energy of noisy training data (“positive” samples  $y^+$ ) while increasing the energy of noisy data sampled from the model (“negative” samples  $y^-$ ) in expectation. The following identity is behind the positive/negative phases in the EBM training:

$$\begin{aligned} \nabla_{\theta} \log p_{\theta}(y) &= -\nabla_{\theta} f_{\theta}(y) - \nabla_{\theta} \log Z(\theta) \\ &= -\nabla_{\theta} f_{\theta}(y) + \frac{\int \nabla_{\theta} f_{\theta}(y) e^{-f_{\theta}(y)} dy}{Z(\theta)} \\ &= -\nabla_{\theta} f_{\theta}(y) + \int \nabla_{\theta} f_{\theta}(y) \cdot p_{\theta}(y) dy \\ &= -\nabla_{\theta} f_{\theta}(y) + \mathbb{E}_{y \sim p_{\theta}(y)} [\nabla_{\theta} f_{\theta}(y)], \end{aligned} \quad (3.7)$$

where  $Z(\theta) = \int e^{-f_{\theta}(y)} dy$  is the partition function.

---

**Algorithm 1:** Discrete Walk-Jump Sampling

---

**Input:** Denoiser,  $g_\phi(y)$ , energy-based model,  $f_\theta(y)$

**Output:** Noisy samples  $y \sim p(y)$ , denoised samples  $\hat{x}(y)$

```
1  $y_0 \sim \text{Unif}([0, 1]^d) + \mathcal{N}(0, \sigma^2 I_d)$ 
2 for  $t = 0, \dots, T - 1$  do
3    $y_{t+1} \leftarrow y_t - \delta \nabla_y f_\theta(y_t) + \sqrt{2\delta} \varepsilon_t$ ,  $\varepsilon_t \sim \mathcal{N}(0, I_d)$ 
4 end
5  $\hat{x}_T \leftarrow y_T + \sigma^2 g_\phi(y_T)$ 
6 return  $\arg \max \hat{x}_T$  // to recover one-hot encoding
```

---

VARIABLE LENGTH PROTEIN SEQUENCE GENERATION. We represent antibody protein molecules as  $x = (x_1, \dots, x_d)$ , where  $x_l \in \{1, \dots, 20\}$  corresponds to the amino acid (AA) type at position  $l$ . Sequences from the Observed Antibody Space (OAS) database [2] are aligned according to the AHo numbering scheme [112] using the ANARCI [113] package and one-hot encoded. Aligning sequences in this way is a practical solution to handling insertions and deletions, which are otherwise troublesome for models that require fixed length inputs and outputs; alignment introduces a “gap” token that can be introduced or removed during sampling to effectively change the length of sequences. This allows the model to capture the distribution of lengths present in natural antibodies. The alignment step maps heavy and light chain sequences of varying lengths to a standard, gapped input size of 149 and 148 respectively with 21 possible discrete tokens including the gap. Thus, the input dimension for every sequence becomes  $d = (149 + 148) \times 21$ . Without loss of generality, any set of proteins can be aligned with a multiple sequence alignment [114]. For other classes of discrete data, pseudo-alignment tokens can be used and randomly inserted into the inputs, or simple BOS and EOS tokens can be used and sampled. An EBM is trained via contrastive divergence on the manifold of smoothed, noisy one-hot encodings,  $y$ , given by  $y = x + \varepsilon$ ,  $\varepsilon \sim \mathcal{N}(0, \sigma^2 I_d)$ , where  $x \in \{0, 1\}^d$ . A separate denoising model is trained with the

objective in Eq. 3.5. New antibody sequences are generated (Fig. 3.2) by sampling noisy samples with Langevin MCMC following gradients from the EBM, denoising with the least-squares estimator, and taking  $\text{argmax } \hat{x}$  to recover a one-hot encoding. Further details related to training and network architecture are given in Appendix A.1.

**PROTEIN DESIGN VS DISCOVERY.** Protein *discovery* is the task of generating novel, unique, and valid samples. Protein *design* refers to taking some starting sequence and making edits to improve function. With dWJS we achieve discovery through unconditional sampling, while design is performed via constrained sampling and scoring. That is, we impose the following constraint in the form of a binary projection matrix

$$P^\top \text{argmax } \hat{x}(y, t) = P^\top s$$

for  $P \in \{0, 1\}^{L \times c}$ , where  $c$  is the number of conserved tokens in the sequence,  $y$  is the noisy sequence at time step  $t$  of Langevin MCMC,  $\hat{x}(y, t)$  is the denoised sample at time  $t$ , and  $s$  is the starting sequence. This constraint ensures that the specified regions of the sequence are conserved, while the non-conserved regions are free to change during Langevin MCMC.

### 3.3.2 DERIVATION OF OPTIMAL NOISE LEVEL FOR DISCRETE SEQUENCE DATA

Throughout the experiments in Section 3.4, we must choose what noise level,  $\sigma$ , to use for training. Empirically, we find that in the protein discovery setting,  $\sigma \geq 0.5$  is sufficient for getting good sample quality. Here, we provide some intuition for choosing a good  $\sigma$ , based on a geometric picture of the concentration of the measure [20]. We define the matrix  $\chi$  with entries

$$\chi_{ii'} = \frac{\|X_i - X_{i'}\|}{2\sqrt{d}}, \quad (3.8)$$

where  $d$  is the dimension of the data and the  $\frac{1}{2\sqrt{d}}$  scaling comes from the concentration of isotropic Gaussians in high dimensions. The critical noise level,  $\sigma_c$ , is defined as

$$\sigma_c = \max_{ii'} \chi_{ii'}$$

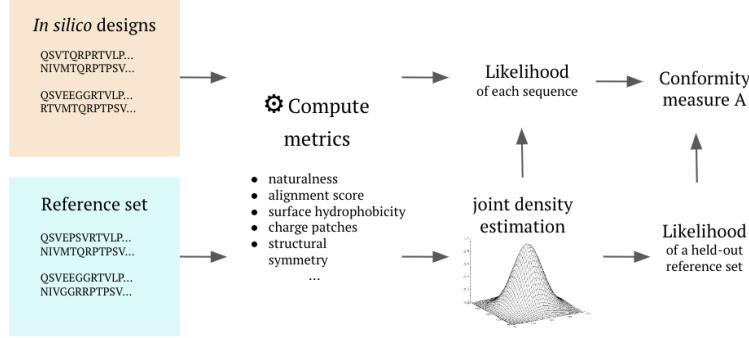
such that for  $\sigma > \sigma_c$ , all noisy data have some degree of overlap. For our antibody sequence data, the statistics of the  $\chi$  matrix are given in Table 3.1 and the histogram of  $\chi_{ii'}$  values is shown in Appendix A.1.6. We find that  $\sigma_c \approx 0.5$ , which agrees with our empirical hyperparameter optimization. Estimating  $\sigma_c$  in this way serves to motivate the empirical success of the  $\sigma$  used in our experiments, and provides helpful guidance on the scale of  $\sigma$  to use for discrete data. Here we take  $d$  to be the length of the input vector ( $d = L = 297$  for aligned antibody sequences); for the flattened sparse one-hot matrices with vocabulary size 21,  $d = 6237$ . This scales  $\sigma_c$  by 0.22, which still gives a useful scale for  $\sigma$ , but is not optimal because of the sparsity of the one-hot matrices.

**Table 3.1:** Statistics of the distance matrix,  $\chi$ , for discrete antibody sequence data.

	min	median	mean	max $\approx \sigma_c$
$\chi$	0.17	0.42	0.41	0.51

### 3.3.3 DISTRIBUTIONAL CONFORMITY SCORE

The Fréchet inception distance (FID) [115] (a metric for image generation quality) and the BLEU (BiLingual Evaluation Understudy) score [116] (for evaluating the quality of machine translation) greatly simplify the evaluation of proposed methods; protein generation lacks such metrics, which motivates us to introduce the “distributional conformity score” (DCS) (Fig. 3.3). The goal of the DCS is to provide a succinct description of how likely generated samples are with respect to a reference distribution, while maintaining novelty and diversity. DCS is designed such



**Figure 3.3:** *in silico* designs sampled with dWJS are compared to a reference set of validation samples. Distributions are characterized with a set of sample quality metrics. Joint density estimation is used to compute the likelihood of designs versus the validation set and likelihoods are condensed into a *distributional conformity score* that characterizes the similarity of generated samples to the reference set.

that improving sample quality corresponds directly to increased probability of generating real, biophysically valid proteins.

We evaluate the probability that our generated sequences conform to a reference distribution using the conformal transducer system [26, 27]. Let  $\mathcal{X} \in \mathbb{R}^d$ ,  $\mathcal{Y} \in \mathbb{R}$ , and  $\mathbf{Z} = \mathcal{X} \times \mathcal{Y}$ .<sup>3</sup> A conformity measure  $A$  is a measurable function that maps a sequence  $(z_1, \dots, z_n) \in \mathbf{Z}^n$  to a set of real numbers  $(\alpha_1, \dots, \alpha_n)$ . Here, we define  $A$  to be the likelihood under the joint density over various properties, including biophysical properties and statistical properties, such as a log-probability under a protein language model (the properties considered, further details, and comparisons to other measures of protein-likeness are given in Appendix A.6). To avoid overfitting the estimator, we split the reference set into a fitting set and a validation set (Algo. 2). In practice, the conformity measure  $A$  and the scores  $\alpha_1, \dots, \alpha_k$  may be pre-computed. At the time of computing the distributional conformity of a query sample, the only calculations that remain are of the score  $\alpha_k$  and the sum to calculate  $p^y$  on line 6 of Algo. 2. The sum can be reduced to  $O(\log k)$  complexity by storing the reference values  $\alpha_1, \dots, \alpha_{k-1}$  in a binary search tree and performing a search for  $\alpha_k$ .

<sup>3</sup>In the discussion of distributional conformity score,  $x$  refers to sample features; elsewhere in the paper  $x$  refers to clean data. Here,  $y$  refers to labels; elsewhere in the paper  $y$  refers to noisy data.

In our context, the reference distribution  $\mathcal{D}$  comprises all antibodies and the label  $y$  represents the property of interest such as expression or binding. Empirically, we find that DCS is a useful *in silico* evaluation metric for developing generative methods and hyperparameter optimization, and that methods with  $\text{DCS} > 0.3$  yield nearly 100% expressing proteins in the wet lab.

---

**Algorithm 2:** Distributional conformity scores for evaluating generated designs

---

**Input:** Reference distribution  $\mathcal{D}$ , test example  $x \in \mathcal{X}$ , conformity measure  $A$ , label  $y$

**Output:**  $p$ -value  $p^y$  (the fraction of validation examples that are less similar to  $D_{z|y}$  than  $x$ )

```

1 Sample  $(z_1, \dots, z_n), z_i \sim D_{z|y}$  and a held-out validation set  $(\tilde{z}_1, \dots, \tilde{z}_{k-1}), \tilde{z}_i \sim D_{z|y}$ 
2 Set  $\tilde{z}_k \leftarrow (x, y)$ 
3 for  $i = 1$  to  $k$  do
4    $\alpha_i \leftarrow A(z_1, \dots, z_n, \tilde{z}_i)$ 
5 end
6  $p^y \leftarrow \frac{1}{k} \sum_{i=1}^k [\alpha_i < \alpha_k]$ 
7 return  $p^y$ 

```

---

### 3.4 EXPERIMENTS

We evaluate our method, discrete Walk-jump sampling (dWJS) (Fig. 3.2), on three antibody generation tasks: 1) distribution learning on paired observed antibody space [2]; 2) the *in vitro* expression and purification of novel antibodies; and 3) most importantly, functional therapeutic antibody design [117]. Crucially, we compare methods using our **distributional conformity score**, which is a sample-to-distribution metric to assess sample quality (analogous to an FID score), rather than the sequence recovery metrics used in previous antibody design work [109, 110]. Sequence recovery is a poor objective for our goal, which is the discovery of novel (large edit distance from known examples), functional antibodies. Details related to model architectures,

**Table 3.2:** Ablikeness metrics, uniqueness, diversity, and distributional conformity scores.

Model	$W_{\text{property}} \downarrow$	Unique $\uparrow$	$E_{\text{dist}} \uparrow$	IntDiv $\uparrow$	DCS $\uparrow$
dWJS (energy-based)	<b>0.056</b>	<b>1.0</b>	58.4	55.3	0.38
dWJS (score-based)	0.065	0.97	<b>62.7</b>	<b>65.1</b>	0.49
SeqVDM	0.062	<b>1.0</b>	60.0	57.4	0.40
DEEN	0.087	0.99	50.9	42.7	0.41
GPT 3.5	0.14	0.66	55.4	46.1	0.23
IgLM	0.08	<b>1.0</b>	48.6	34.6	<b>0.533</b>
ESM2	0.15	<b>1.0</b>	70.99*	77.56*	0.061

**Table 3.3:** Measured protein synthesis.

Model	total <sub>expressed</sub> $\uparrow$
<b>dWJS (score-based)</b>	<b>1.0</b>
dWJS (energy-based)	0.97
EBM	0.42

training, baseline methods, and sequence sampling are in Appendix A.1.

### 3.4.1 dWJS GENERATES NATURAL, NOVEL, DIVERSE ANTIBODIES *IN SILICO*

We measure generative model performance with a suite of “antibody likeness” (ab-likeness) metrics including labels derived from the AA sequence with Biopython [118]. Sequence property metrics are condensed into a single scalar metric by computing the distributional conformity score and the normalized average Wasserstein distance  $W_{\text{property}}$  between the property distributions of samples and a validation set. The average total edit distance  $E_{\text{dist}}$  summarizes the novelty and diversity of samples compared to the validation set, while internal diversity (IntDiv) represents the average total edit distance between samples. Our method achieves strong ablikeness results (Table 3.2), simply by increasing  $\sigma$  to  $\geq 0.5$ .

dWJS with dEBM sampling achieves the best agreement with the validation set property distribution and highest percentage of unique samples, while dWJS with score-based sampling has high distributional conformity score, novelty, and diversity. We compare to a latent sequence diffusion method (SeqVDM), (a discrete generalization of variational diffusion; [119]), a score-based model with an energy parameterization (DEEN), a transformer-based language model trained specifically for antibody design (IgLM); [120], ESM2 [121], and a pre-trained large language model (LLM) (GPT 3.5). Our dWJS methods have faster sampling time and lower memory footprint than diffusion, autoregressive, and score-based baselines (Table A.3), while also having better sample quality. Our methods outperform IgLM on antibody-likeness, edit distance, and internal diversity metrics, whereas IgLM has the best DC score and the lowest (worst) IntDiv. This indicates that IgLM samples are extremely close to the reference set and samples are similar to one another. As our score-based dWJS has a DC score of 0.49 and a 100% expression rate in the lab, it is likely that all IgLM samples would successfully express in the lab as well. IgLM does produce 100% unique samples, outperforming score-based dWJS and GPT 3.5. Due to IgLM being an autoregressive sampler, our method (energy-based dWJS) has 43 $\times$  faster sampling speed, which is useful when generating and ranking large numbers of designs. To mimic the *ab initio* generation task for ESM2, we increase the masking percentage as high as we can (40%) while still generating valid samples and infill validation set sequences to generate new samples. As expected, ESM2 performs extremely poorly in generating *ab initio* samples with good antibody-likeness, because it is not trained for generation. It generates highly repetitive sequences that are very dissimilar to antibodies (hence the high, but meaningless,  $E_{\text{dist}}$  and IntDiv scores). As a masked language model, ESM2 infilling is the fastest sampler, but at the cost of poor sample quality. Details on the baseline methods, and IgLM and GPT 3.5 prompts are given in Appendices A.1 and A.4.

**Table 3.4:** Predicted and measured antibody binding affinity.

Model	$p_{\text{bind}} \uparrow$	$\text{total}_{\text{bind}} \uparrow$
<b>dWJS (energy-based) (Ours)</b>	<b>0.96</b>	<b>0.70</b>
dWJS (score-based) (Ours)	0.95	N/A
LaMBO-2 [65]	N/A	0.25
AbDiffuser [122]	0.94	0.22 (0.57)
SeqVDM	0.75	N/A
GPT 4	0.74	N/A
Transformer	0.60	N/A
EGNN	0.58	N/A

### 3.4.2 dWJS GENERATES NATURAL, NOVEL, DIVERSE ANTIBODIES *IN VITRO*

Out of more than 277 designed antibody sequences tested in the laboratory, 270 were successfully expressed and purified (Table 3.3). We achieved the 97.47% *in vitro* success rate by developing dWJS to capture the antibody distribution *in silico* as measured by  $W_{\text{property}}$  and distributional conformity score. For comparison, sequences from an EBM (trained on clean data with samples drawn using traditional Langevin MCMC) achieved a 42% expression rate. An antibody sequence comprised of random vocabulary tokens would be expected to have a 0% expression rate, and in laboratory experiments we have confirmed that a small number of edits (< 4) can destroy expression if the proposal distribution (generative model) is poorly optimized.

### 3.4.3 dWJS GENERATES FUNCTIONAL ANTIBODY VARIANTS *IN VITRO*

To further show the robustness of our method, we consider the task of training generative models on a hu4D5 antibody mutant dataset [117] and compare to baseline models. The dataset consists of 9k binding and 25k non-binding hu4D5 CDR H3 mutants with up to 10 mutations (after de-duplication and removing samples that are labeled both binding and non-binding). This yields a  $10^{13}$  dimensional search space. The mutants were measured in lab experiments to determine

their binding to HER2 antigen. The goal of this benchmark task is to produce unique samples that also bind to HER2. We trained dWJS models (score-based and energy-based) on only the binder set at a noise level of  $\sigma = 0.5$ , while also training a 1D-CNN binary classifier to classify binders and non-binders. The classifier achieves 86% accuracy on an IID validation split. Then, we classified 1000 samples from each dWJS generative model and four baseline models also trained on the hu4D5 binder set. We compare *in silico* to three diffusion models: 1) a sequence transformer based on BERT Devlin et al. [58] that generates sequences, 2) an E(n) Equivariant Graph Neural Network (EGNN) Satorras, Hoogeboom, and Welling [123] that codesigns (sequence, structure), and 3) a latent sequence diffusion model, SeqVDM; and a pre-trained LLM, GPT 4. The specific prompt used for GPT 4 is given in Appendix A.4. The probability of binding for *unique* designs from each model is reported in Table 3.4. Our methods produce the highest percentage of unique predicted binders.

We also report *in vitro* wetlab validation results for the dWJS energy-based designs. dWJS produces the highest percentage of functional antibodies that bind to target (total<sub>bind</sub> in Table 3.4). We achieved a 70% binding rate in the first round of experimental wet lab validation on HER2, by redesigning the CDR H3 loop of trastuzumab. To the best of our knowledge, this is the highest reported binding rate of any antibody design method applied to trastuzumab CDR H3 redesign. Concurrent with the first appearance of our work, two diffusion models reported *in vitro* wet-lab results for this task: AbDiffuser [122], a (sequence, structure) antibody codesign method, and LaMBO-2 [65], a guided discrete diffusion method. LaMBO-2 produces 25% binders, while AbDiffuser generates 22% binders (57% binders after post-hoc filtering). Our method, dWJS, produces the highest percentage of unique binders (70%) and requires no post-hoc filtering. While the diffusion methods include other important capabilities including structure generation (AbDiffuser) and guided sampling (LaMBO-2), our method outperforms diffusion for discrete distribution learning, which is our goal. For this experiment, our model is trained only on the publicly

available [117] dataset. This training dataset, our code,<sup>4</sup> and our experimental results are open and we encourage other researchers to compare results in the same *in vitro* setting. Further details on wetlab experiments are presented in Appendix A.5.

### 3.5 RELATED WORK

Energy-based models (EBMs) [103] are a class of physics-inspired models that learn an energy function defining a probability distribution over data with a rich history that goes back to Boltzmann machines Hinton and Sejnowski [102]. Estimating unnormalized densities has also been formulated using score matching Hyvärinen [62]. This formulation led to probabilistic models for denoising autoencoders [104, 124, 125], but also has an empirical Bayes interpretation that is most related to this work. In particular, the neural empirical Bayes (NEB) [20] formalism unifies kernel density estimation [126] and empirical Bayes [23] to transform the unsupervised learning problem into a more tractable form where a neural network energy function is parameterized to capture a *smoothed* data distribution. Our work is the first study of the NEB formalism for discrete data. Discrete diffusion models such as [64] learn an iterative denoising process over many different noise levels by prescribing a noise process over discrete data that converges to a known categorical distribution.

Approaches borrowing from traditional ML generative modeling have been used to model antibodies [68, 120, 127, 128], but typical natural-language-based methods struggle to capture the data distribution of antibodies, for which there is limited training data ( $\sim$ 1K - 1M high-quality sequences depending on the distribution of interest) and additional challenges due to the high-entropy variable regions of the sequence. Here, we address the above challenges with training and sampling discrete sequences using a novel formulation of decoupled energy- and score-based modeling.

---

<sup>4</sup><https://github.com/prescient-design/walk-jump>

### 3.6 CONCLUSIONS

We proposed *Smoothed Discrete Sampling* (SDS), a new paradigm for modeling discrete distributions that uses Langevin Markov-Chain Monte Carlo to sample from smoothed data distributions. We introduce the discrete Walk-Jump Sampling (dWJS) algorithm and evaluate it on the antibody discovery and design problems, showing the capability of our method to generate novel, diverse, and functional antibodies as measured by synthetic biophysical property distributions, similarity metrics, and *in vitro* experiments. The strong regularization provided by fitting the energy function to noisy data completely prevents overfitting and training instabilities, resulting in fast and efficient training and sampling with low compute requirements. dWJS discards many of the commonly used techniques for improving EBM training with Langevin MCMC (replay buffers,  $\ell_2$  norm penalty, simulated annealing, rejection sampling, etc.) and reduces the engineering complexity of training EBMs and diffusion-based models to a single hyperparameter choice: the noise level,  $\sigma$ . Altogether, our results suggest a simplified, more general and robust framework for training and sampling from discrete energy- and score-based models with applications to therapeutic molecule design. Future work will probe the generality of our results to other classes of molecules and even other data modalities (e.g., images), as well as theoretical investigation into the results presented here.

### 3.7 RETROSPECTIVE

This work introduced a new generative model for protein sequences and demonstrated its effectiveness at generating realistic, functional antibodies. We show that our model’s generation quality is competitive with other contemporary state-of-the-art generative models for antibodies. A follow-up work that demonstrates the generative capabilities of discrete walk-jump sampling for a more diverse collection of protein families and functions provide evidence to justify its utility

as a generalized sampler. Additionally, our work shows that a simple ByteNet architecture trained from scratch on paired antibody chains was an effective means to producing antibody sequences. This work presented an unguided sampling approach, insofar as the Langevin walk steps are only informed by the learned gradient model. Additionally, our work did not leverage pre-trained antibody language models such as Ablang [129] or IgLM [130]. Since the publication of this work, multiple follow-up works have addressed these areas of interest. Mahajan, et al [131] exploited large language models for walk-jump sampling in order to enhance expressivity. Additionally, Ikram, et al [132] and Joren, et al [133] explored classifier-based and classifier-free guidance techniques to drive sampling into desirable regions of sequence space.

## 4 | DEEP MANIFOLD SAMPLER

### 4.1 FUNCTION-GUIDED DESIGN BY DEEP MANIFOLD SAMPLING

This section is a reprint of workshop paper:

Vladimir Gligorijević, **Daniel Berenberg**, Stephen Ra, Andrew Watkins, Simon Kelow,  
Kyunghyun Cho, and Richard Bonneau

*Machine Learning for Structural Biology Workshop, NeurIPS 2021*

#### 4.1.1 INTRODUCTION

Protein design has led to remarkable results in past decades in synthetic biology, agriculture, medicine, and nanotechnology, including the development of new enzymes, peptides and biosensors [134]. However, sequence space is large, discrete, and sparsely functional [135], where only a small fraction of sequences may fold into stable structural conformations. Taken together, these considerations present important challenges for automated and efficient exploration of design space.

Building on previous works on representation learning from large-scale protein sequence data [81, 86, 88, 136–138], we introduce a novel generative model-based approach called “deep manifold sampling” for accelerating function-guided protein design to explore sequence space more effectively. By combining a sequence denoising autoencoder (DAE) with a function clas-

sifier trained on roughly 0.5M sequences with known function annotations from the Swiss-Prot database [34], our deep manifold sampler is capable of generating diverse sequences of variable length with desired functions. Moreover, we conjecture that by using a non-autoregressive approach, our deep manifold sampler is able to perform more effective sampling than previous autoregressive models.

#### 4.1.2 RELATED WORK

Recent work has demonstrated success in learning semantically-rich representations of proteins that encapsulate both biophysical and evolutionary properties. In particular, language models (LM) using bi-directional long short-term memory (LSTM) [139] and attention-based [86] architectures and trained on protein sequences have yielded useful representations for many downstream tasks, including secondary structure and contact map predictions [86], structural comparison [139], remote homology detection [88], protein engineering and fitness landscape inference [136], and function prediction [140].

Other studies have focused on generative modeling for producing realistic protein structures – for example, using Generative Adversarial Networks (GAN) for creating pairwise distance maps [141] and variational autoencoders (VAE) for 3D coordinate generation of protein backbones [142] – and designing new sequences. One advantage in formulating a design problem with sequences has traditionally been the relative availability of data as compared to experimentally-determined structures. Balakrishnan et al. [143] used graphical models trained on multiple sequence alignments (MSA) to sample new sequences. More recently, VAEs [144, 145] have been used for designing novel enzymes and T-cell receptor proteins, obviating the need for MSA, but they have been largely limited to a single family of proteins. Additionally, a few generative models have been proposed for conditional design [137, 138, 146]: Greener *et al.* [146] use a VAE conditioned on structural features for generating sequences with metal binding sites, Madani *et al.* [137] use a conditional LM for sampling proteins, where each amino acid is sampled sequentially, and

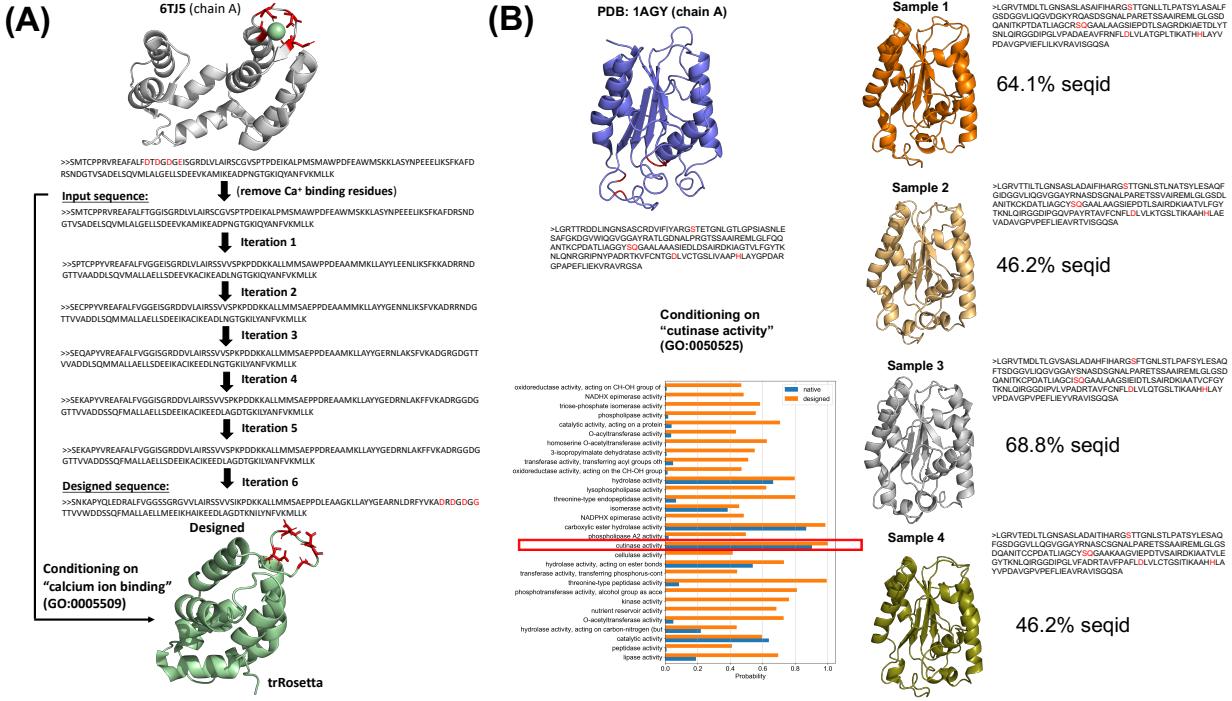
Shin *et al.* [138] use an autoregressive model to generate a nanobody sequence library with high expression levels.

Motivated by conditional design as translation task, we develop an approach for generating protein sequences with desired functions, where sequences are translated from an input sequence to an output sequence with higher property values. Our deep manifold sampling approach uses a denoising autoencoder (DAE), a self-supervised model that implicitly estimates the structure of the data-generating density by denoising stochastically corrupted training examples [59, 104, 147, 148]. We use a Markov chain Monte Carlo (MCMC) process [148] to sample from the density function learned by the encoder. We corrupt this sample and repeat the procedure above to produce a chain of samples from the DAE.

We also consider known issues with autoregressive models including decoding latency [149, 150], difficulty of parallelization at inference time [151–153], and exposure bias at test-time generation [154, 155]. By using a non-autoregressive modeling strategy, our deep manifold sampler is capable of predicting multiple mutations — including insertions and deletions — at different positions in a given sequence resulting in sequences of varying lengths. We conjecture that in doing so, our manifold sampler enables effective exploration of the overall fitness landscape of properties, resulting in diverse protein designs with desirable properties.

#### 4.1.3 METHODS

We propose to learn a protein sequence manifold by training a DAE on a large database of observed sequences spanning multiple protein families. Moreover, to ensure that generated sequences satisfy a set of desired functional constraints, we combine a protein function classifier with the DAE to guide sampling.



**Figure 4.1:** (A) A designed sequence of  $\text{Ca}^{2+}$ -binding protein obtained by altering the sequence of calmodulin, calcium-binding protein (PDB: 6TJ5, chain A) after removing its calcium binding site. (B) Redesign of *fusarium solani pisi* cutinase (PDB ID: 1AGY, chain A) cutinases with enhanced functions.

#### 4.1.4 A SEQUENCE DENOISING AUTOENCODER

Our goal is to generate a diverse set of protein sequences that exhibit a high level of a desired function using a sequence DAE [156]. We want to map an input sequence  $\tilde{X}$  to a target sequence  $X$ , where  $X$  always has a higher level of some desired protein function than  $\tilde{X}$ . We formulate this task as a language modeling problem, where we model the joint probability of the tokens of a target protein sequence  $X = (x_1, \dots, x_L)$  of length  $L$  given its corrupted version  $\tilde{X}$  of length  $\tilde{L}$  as:

$$\log p(X|\tilde{X}) = \sum_{t=1}^L \log p(x_t|\tilde{X})$$

We model the joint distribution  $p(X|\tilde{X})$  with the proposed architecture (Supplementary Fig. B.1). First, we apply a sequence corruption process  $C(\tilde{X}|X)$  that takes as input sequence  $X$  of length  $L$  and returns corrupted version  $\tilde{X}$ , potentially of different length  $\tilde{L}$  (B.1). The corrupted

sequence  $\tilde{X}$  is passed as an input to the encoder  $Q_\theta(\tilde{Z}|\tilde{X})$  that maps  $\tilde{x}_1, \dots, \tilde{x}_{\tilde{L}} \in [0, 1]^{\tilde{L} \times 22}$  to a sequence of continuous representations  $\tilde{Z} = \tilde{z}_1, \dots, \tilde{z}_{\tilde{L}} \in \mathbb{R}^{\tilde{L} \times d}$ . To predict the probabilities of target tokens  $X = (x_1, \dots, x_L)$ , we use a monotonic location-based attention mechanism [152] to transform  $\tilde{L}$  vectors of  $\tilde{Z}$  into  $L$  vectors. The length transform  $f_\sigma(Z|\tilde{Z}, \Delta L)$  takes in  $\tilde{Z}$  and the length difference  $\Delta L = L - \tilde{L}$  and returns the new vectors of the target sequence  $Z$ , with  $L = \tilde{L} + \Delta L$ .  $f_\sigma(Z|\tilde{Z}, \Delta L)$  then computes each  $z_i$  as a weighted sum of the vectors from the encoder, i.e.,  $z_i = \sum_{j=1}^{\tilde{L}} a_j^i \tilde{z}_j$ , where  $a_j^i$  is an attention coefficient computed as in Shu *et al.* [152].  $Z$  is then passed to the decoder  $P_\phi(X|Z)$ , which predicts the probabilities of the target tokens.

#### 4.1.4.1 LENGTH PREDICTION AND TRANSFORMATION

Although the length difference is known during training, it is not readily available at inference time and must be predicted. We use an approach previously proposed by Shu *et al.* [152] and Lee *et al.* [150] and construct a length predictor as a classifier that outputs a categorical distribution  $P_\eta(\Delta L|\tilde{Z})$  over the length difference. The classifier takes in a sequence-level representation obtained by pooling representations  $\tilde{z}^{pool} = \frac{1}{\tilde{L}} \sum_{i=1}^{\tilde{L}} \tilde{z}_i$ , and produces a categorical distribution that covers the maximum range of length differences,  $[-\Delta L_{max}, \Delta L_{max}]$ , where  $\Delta L_{max}$  is determined by the choice of a corruption process (B.1). The classifier is parameterized by a single, fully connected linear layer with a *softmax* output.

#### 4.1.4.2 A PROTEIN FUNCTION CLASSIFIER

For conditional sequence design, we incorporate a protein function classifier by training it on the representations from the encoder of the DAE. The goal is to exploit the error signal from the function classifier at test time to guide sampling towards sequences with desired functions. We train a multi-label classifier  $P_\omega(Y|\tilde{Z})$  that takes in a latent representation  $\tilde{Z}$  from the encoder and that outputs a vector  $Y$  of probabilities for each function as well as the classifier's internal latent representation  $Z_c$ . We parameterize  $P_\omega$  with one multi-head attention (MHA) layer that maps the

initial sequence feature representation  $\tilde{Z}$  to an internal feature representation,  $\tilde{Z}_c$ , of the same hidden dimension as  $\tilde{Z}$ , which is pooled to form a protein-level representation;  $z_c^{pool} = \frac{1}{\tilde{L}} \sum_{i=1}^{\tilde{L}} \tilde{z}_{ci}$ . This protein-level representation is passed to single, fully connected layer followed by a point-wise sigmoid function that returns function probabilities.

#### 4.1.4.3 FUNCTION-CONDITIONED SAMPLING

We guide sampling towards a target function  $i$  at every sampling step by using the gradient of the function classifier’s predictive probability of  $i$  to update the encoder’s vectors and increase the likelihood of higher expression of the desired target function. At every generation step, we update the internal state of the encoder as follows:

$$\tilde{Z} \leftarrow \tilde{Z} + \epsilon \frac{\nabla_{\tilde{Z}} \log P_{\omega}(Y_i|\tilde{Z})}{\|\nabla_{\tilde{Z}} \log P_{\omega}(Y_i|\tilde{Z})\|} \quad (4.1)$$

where  $\epsilon$  controls the strength of the function gradients. The output  $\tilde{Z}$  is then passed to the length transform, together with the predicted  $\Delta L$  sampled from the length predictor, and then to the decoder (Supplementary Fig. B.1).

#### 4.1.5 RESULTS

We demonstrate the applicability of our method in three different case studies: 1) adding a function to an existing fold by installing a binding site (Fig. 4.1A) 2) diversifying a protein sequence by preserving its function and salient residues (Fig. 4.1B), and 3) modifying protein function by globally changing the protein fold (B.4).

**DESIGNING A SEQUENCE WITH A METAL BINDING SITE.** We study the ability of the model to add a potential metal binding sites to a protein. In particular, we test the model’s ability to recover

metal binding sites by starting the sampling procedure from a sequence of a metal binding protein after removing the known binding residues, including residues involved in calcium binding (three aspartate and one glutamic amino acid residues) from a calcium-binding protein (PDB: 6TJ5, chain A; Fig. 4.1A). Starting from the altered sequence, we perform sampling by conditioning on *calcium ion binding* (GO:0005509). After six MCMC steps, we obtain a sequence with a high score for calcium ion binding and observe a sequence motif frequently found in most known calcium binding proteins (Fig. 4.1A); highlighted in red). The designed sequence has 48.7% sequence identity to the the starting one. When folded using the *trRosetta* package [157], it forms a helix-loop-helix structural domain at the location of the predicted binding site [158]. The three aspartite amino acids in this loop are negatively charged and interact with a positively charged calcium ion. The glycine is necessarily due to the conformational requirements of the backbone [158, 159].

REDESIGN OF CUTINASES WITH ENHANCED FUNCTIONS. We test the ability of model to diversify an existing protein sequence by preserving the functional residues. Here, we use sequence of *fusarium solani pisi* cutinase. Cutinases are responsible for hydrolysis of the protective cutin lipid layer in plants and thus have been used for hydrolysis of small molecule esters and polyesters. Sampled sequences obtained after 6 generations of sampling steps with the constrain imposed on *cutinase activity* (GO:0050525) are folded by *trRosetta*. The results are show in Fig.4.1B with catalytic residues are highlighted in red. Our function classifier shows the probability scores for *cutinase activity* of the designed sequences. We perform multiple sequence alignment of the top scoring sampled sequences showing the catalytic residues of the initial cutinase (1AGY-A) preserved by our manifold sampling strategy.

## 4.2 MULTI-SEGMENT PRESERVING SAMPLING FOR DEEP MANIFOLD SAMPLER

This section is a reprint of workshop paper:

**Daniel Berenberg**, Jae Hyeon Lee, Simon Kelow, Ji Won Park, Andrew Watkins, Vladimir Gligorijević, Richard Bonneau Stephen Ra, and Kyunghyun Cho

*Machine Learning for Drug Discovery Workshop, ICLR 2022*

### 4.2.1 INTRODUCTION

Protein sequence families, particularly antibodies, have both well-conserved and variable regions. In antibodies, the heavy and light chain sequences consist of highly conserved regions known as the framework as well as an array of distinct hypervariable loops, known as complementarity-determining regions (CDRs) [160]. Despite the intrinsic variability of CDRs, conditional variation is often conferred by the gene locus admitting the protein [161]. Much of an antibody’s antigen-binding affinity is owed to the CDRs, while the framework remains fixed or requires minimal change [162]. For *in silico* modeling, integrating these established aspects of structure and binding can drive the development of better *in situ* antibody therapeutic design [163]. While work in protein language modeling suggests that models can learn these evolutionary conservation rules [85–87], it is an open challenge as to how to explicitly incorporate prior insight at test-time generation, such as sequence-level annotations [164], to restrict sampling in certain segments.

The deep manifold sampler was recently proposed as an effective method to sample novel sequences by iterative, optionally gradient-guided steps, of sequence denoising [68]. Empirically, gradient-based guided sampling was shown to selectively encourage changes in functional sites,

implicitly leaving non-functional regions unperturbed. In this work, we propose an alternative to the gradient-based guided design procedure in which predefined regions of a sequence are explicitly preserved, leaving sampling to take place in *a priori* known notable sequence regions. We conduct an experiment on antibody sequences to demonstrate the deep manifold sampler’s ability to focus sampling on a subset of sequence positions. We do so by deliberately corrupting select regions of antibody sequences, that correspond to CDRs, and evaluating the length distribution and composition of sampled CDRs.

#### 4.2.2 BACKGROUND: THE DEEP MANIFOLD SAMPLER

The deep manifold sampler [68] is a denoising autoencoder (DAE) specialized to handle variable-length sequences. As with a typical DAE [147], the deep manifold sampler consists of three modules; a corruption process  $C(\tilde{x}|x)$ , an encoder  $F$  and a decoder  $G$ . Unlike the usual DAE however, the deep manifold sampler has an extra module that determines the change in the length, to which we refer as the “length conversion” [28].

The deep manifold sampler assumes as input a sequence of discrete tokens,  $x = (x_1, x_2, \dots, x_L)$ , where each token  $x_t$  is an item from a finite vocabulary  $V$  of unique words or subwords. In the case of protein sequence modeling,  $V$  consists of all unique amino acids. The sequence  $x$  is corrupted with the corruption process  $C$ , resulting in a *noisy* input sequence  $\tilde{x} \sim C(\tilde{x}|x)$ . This corruption process can be arbitrary as long as it is largely local and unstructured. It may even alter the length of the sequence,  $|x| \neq |\tilde{x}|$ .

The encoder  $F$  turns the corrupted sequence  $\tilde{x}$  into a set of hidden vectors,  $h = (h_1, h_2, \dots, h_{|\tilde{x}|})$ , where  $h_t \in \mathbb{R}^d$ . The encoder can be implemented using any of the widely-used deep architectures, such as transformers [84], convolutional networks [165] and recurrent networks [166, 167]. In this work, we follow the original deep manifold sampler’s encoder, which was implemented as a transformer.

The hidden vectors are pooled to form a single-vector representation:

$$\bar{h} = \frac{1}{|\tilde{x}|} \sum_{t=1}^{|\tilde{x}|} h_t. \quad (4.2)$$

This pooled representation is used by the length conversion to predict the change in the length. At training time, this length change predictor is trained to output  $\Delta l^* = |\tilde{x}| - |x|$ . When we sample sequences from the deep manifold sampler after training, we use the predicted change  $\Delta l$  to adjust the size of the hidden vector set. The adjusted hidden vector set consists of  $|\tilde{x}| + \Delta l$  hidden vectors,  $z = (z_1, \dots, z_{|\tilde{x}|+\Delta l})$ , where

$$z_t = \sum_{t'=1}^{|\tilde{x}|} \omega_{t,t'} h_{t'} \quad (4.3)$$

with

$$w_{t,t'} \propto \frac{-1}{\exp(\sigma)} \left( t' - \frac{|\tilde{x}|}{|\tilde{x}| + \Delta l} t \right)^2. \quad (4.4)$$

The decoder  $G$  then takes this transformed hidden vector sequence  $z$  and outputs a corresponding sequence of logit vectors,  $\tilde{y} = (\tilde{y}_1, \dots, \tilde{y}_{|\tilde{x}|+\Delta l})$ , where  $\tilde{y}_t \in \mathbb{R}^{|V|}$ . These logits are turned into probability distributions over the vocabulary  $V$  in many different ways. The original deep manifold sampler implements a non-autoregressive approach [168, 169], where each logit is independently turned into a distribution:

$$p(y_t = v | \tilde{x}, \Delta l) = \frac{\exp(\tilde{y}_t^v + b^v)}{\sum_{v' \in V} \exp(\tilde{y}_t^{v'} + b^{v'})}, \quad (4.5)$$

where  $b^v$  is a bias for token  $v$ .

It is however also possible to use these logits together with a more powerful output module, such as a conditional random field (CRF; [170]), as was recently done in [171], and autoregressive

language models [172]. For experiments in this paper, we use a variant of the deep manifold sampler in which we use a CRF at the end of the decoder.

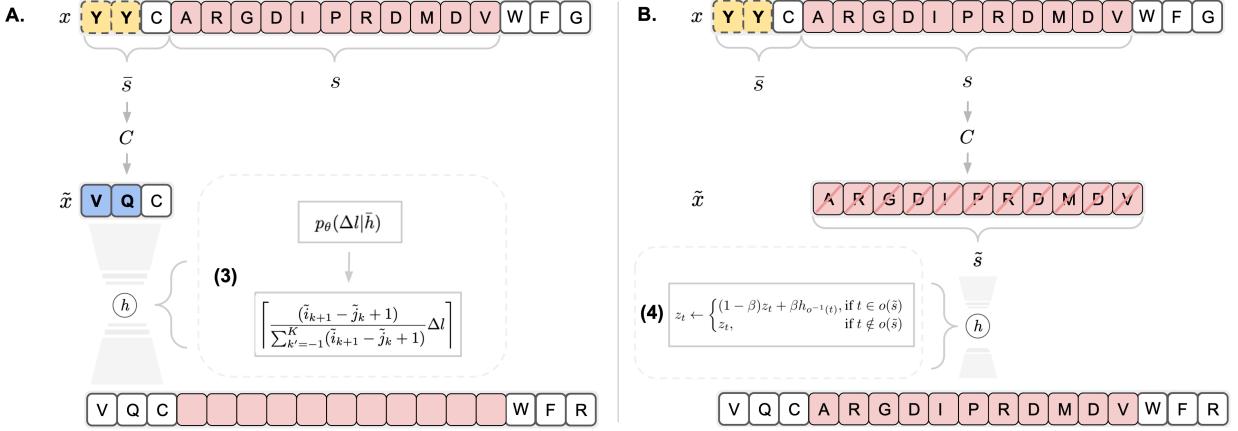
At training time, we minimize the negative log-probability of the original sequence  $x$  given the corrupted version  $\tilde{x}$  and a known  $\Delta l^*$  to train the encoder and decoder, while minimizing the negative log-probability of  $\Delta l^*$  to train the length change predictor. We parameterize the latter as a classifier. Once training is done, we can draw a series of samples from the deep manifold sampler by repeating the process of corruption, length conversion, and reconstruction.

While the original deep manifold sampler has an additional function predictor that can be used to guide the sampling procedure, we omit that here, as this is optional and can be replaced with another computational oracle without altering the sampling procedure that is the focus of this paper.

#### 4.2.3 MULTI-SEGMENT PRESERVING SAMPLING

The deep manifold sampler was originally proposed in the context of protein design in Gligorijević et al. [68]. Within this setting, we often consider biological, chemical, and physical knowledge in order to impose constraints that narrow down a large, combinatorial search space [173, 174]. The deep manifold sampler on the other hand stays true to the key principle of deep learning, that is, end-to-end learning, which makes it challenging to explicitly incorporate this knowledge into both learning and sampling. In this paper, we take one step towards enabling this into the sampling procedure of the deep manifold sampler. We assume the availability of knowledge in which segments of an original sequence, from which sampling starts, must be preserved in order to maintain a set of desirable properties. For example, in the case of antibody engineering, it may be desirable to only alter CDR loops while leaving all framework residues intact [162].

Let  $x = (x_1, x_2, \dots, x_L)$  be the initial sequence from which we run the deep manifold sampler to draw a series of samples over the sequence manifold. Instead of unconstrained sampling, we consider a scenario in which we are provided with a set of non-overlapping segments of the



**Figure 4.2:** Multi-segment preserving sampling for  $\beta = 1$ . (A) Non-preserved segments  $\bar{s}$  are corrupted using corruption process  $C$ , for which a given token (yellow) may be randomly perturbed (blue). This is encoded as hidden vector set  $h$ . Length change predictor  $p_\theta(\Delta l | \bar{h})$  outputs  $\Delta l$ , which is distributed across  $\bar{s}$  (Eq. 4.8). (B) Segment-preserving sampling follows similar operations on preserved segment  $s$  (red) with notable differences. Corruption  $C$  yields an unaltered sequence  $\tilde{x}$  and we carry over hidden vector  $h_t$  of a token within preserved segment  $\tilde{s}$  with strength  $\beta$  (Eq. 4.9).

sequence that must be preserved in-order by their starting and ending indices (inclusive):

$$s = ((i_1, j_1), \dots, (i_K, j_K)) \quad (4.6)$$

subject to  $i_k \leq j_k$  for all  $k$  and  $j_k < i_{k'}$  for all  $k' > k$ . We refer to this set as a *preserved-segment set*. Likewise, we can imagine the complement segment set  $\bar{s}$  that contains all the segments that are between the to-be-preserved segments in  $s$ :

$$\bar{s} = ((0, i_1 - 1), (j_1 + 1, i_2 - 1), \dots, (j_K + 1, |x| + 1)). \quad (4.7)$$

In order to preserve these segments while altering the remaining parts of the sequence, including their respective lengths, we make a series of modifications to the sampling procedure of the deep manifold sampler. First, we alter the corruption process  $C$  such that it does not corrupt

the preserved segments. For instance, if the corruption process randomly adds or removes tokens, this is only done to the segments in the complement set  $\tilde{s}$  but not to those in  $s$ . Doing so results in  $\tilde{x}$  and also changes the segment set  $s$  in order to appropriately reflect the changes in the indices due to insertions and deletions. We use  $\tilde{s}$  to refer to the preserved segment set of the corrupted sequence  $\tilde{x}$ .

However, the encoder still results in the hidden vector set  $h$  that corresponds to  $\tilde{x}$ . While the length change prediction also stays the same, returning length change  $\Delta l$ ,  $\Delta l$  needs to be distributed across the non-preserved segments in order to avoid altering the length of any preserved segment in  $\tilde{s}$ . We do so proportional to the original lengths of the non-preserved segments. In other words, we add to the length of each non-preserved segment  $(\tilde{j}_k + 1, \tilde{i}_{k+1} - 1)$ :

$$\left\lceil \frac{(\tilde{i}_{k+1} - \tilde{j}_k + 1)}{\sum_{k'=-1}^K (\tilde{i}_{k+1} - \tilde{j}_k + 1)} \Delta l \right\rceil, \quad (4.8)$$

where  $\tilde{j}_0 = 0$  and  $\tilde{i}_{K+1} = |\tilde{x}| + 1$ .

After distributing the length difference among the non-preserved segments, we can now construct the index map  $o$  that tells us which segment in the new sequence corresponds to each of the preserved segment in  $\tilde{x}$ . In other words,  $y_{o(\tilde{i}_k):o(\tilde{j}_k)} = \tilde{x}_{\tilde{i}_k:\tilde{j}_k}$ . Let us use  $o(\tilde{s})$  to denote the preserved-segment set derived from  $s$  and the length distribution above.

The actual length conversion happens just like before, as in Eq. (4.3). We however add an extra step after the length conversion in order to give the decoder a hint about preserved segments and their contents. This is done by carrying over the original hidden vector  $h_t$  of a token within a preserved segment:

$$z_t \leftarrow \begin{cases} (1 - \beta)z_t + \beta h_{o^{-1}(t)}, & \text{if } t \in o(\tilde{s}) \\ z_t, & \text{if } t \notin o(\tilde{s}) \end{cases} \quad (4.9)$$

$o^{-1}$  is the inverse index map, and  $\beta \in [0, 1]$  is the strength of carry-over.

The decoder turns this length-converted and segment-preserving hidden sequence  $z$  into a sequence of logit vectors  $\tilde{y}$ , just like the original sampling procedure. We then modify the logit vector corresponding to a token with a preserved segment to force the sampled outcome to preserve the token identity:

$$\tilde{y}_t^v \leftarrow \begin{cases} \infty, & \text{if } t \in o(\tilde{s}) \text{ and } v = \tilde{x}_{o^{-1}(t)} \\ -\infty, & \text{if } t \in o(\tilde{s}) \text{ and } v \neq \tilde{x}_{o^{-1}(t)} \\ \tilde{y}_t^v, & \text{if } t \notin o(\tilde{s}) \end{cases} \quad (4.10)$$

In the case of non-autoregressive modeling, this would result in a Categorical distribution for a preserved token to assign the entire probability mass ( $= 1$ ) to the original token identity. If a CRF is used at the end, this would prevent any sequence that violates preservation from being decoded out with non-zero probability.

We can repeat this sampling step with the newly sampled sequence and the corresponding preserved-segment set. This allows us to iteratively draw a series of samples while preserving the segments from the original sequence, designated by the preserved-segment set  $s$ . Because this iterative sampling procedure preserves multiple segments and their contents, we refer to this procedure as *multi-segment preserving sampling* (Figure 4.2).

#### 4.2.4 EXPERIMENTS

The proposed algorithm for multi-segment preserving sampling is designed to completely preserve designated segments. Here, we demonstrate a potential application in antibody design enabled by our algorithm coupled with the deep manifold sampler. Antibodies with a particular V-gene have fixed lengths in the framework as well as CDR1 and CDR2 regions. As a result, antibodies display most of their diversity in length and amino acid composition in CDR3 [175]. To demonstrate the effectiveness of our approach and restricted variation of the preserved segments,

$\beta$	Aligned CDR3 sequence	Edit distance
N/A (original)	ARDPEWDPF-QANY-YYYGMDV	0
0.0	ARDPEWDPF-QAN-YYYGMDV	3
0.1	ARDPEWDPFFQANYNYYGMVD	3
0.5	KRDPEWDRF-QAPY-YTVGMDV	5
0.9	ARGPECDFH-QAV-DIYYGMDV	6

**Table 4.1:** Example outputs of multi-segment preserving sampling when restricting variation to the CDR3 region under different settings of  $\beta$ . Display is restricted to the sampled region, the rest is preserved by construction.

we select all unique human antibody sequences with the *IGHV1-18* gene from the Observed Antibody Space (OAS) database [2] for multi-segment preserving sampling. Using a deep manifold sampler, we sample exclusively from the CDR3, while preserving other regions, and show the length and log-probability (GPT-2) distributions of the generated sequences qualitatively coincide with that of the test data. Table 4.1 illustrates examples of sampled CDR3 regions under different settings of carry-over strength  $\beta$ .

#### 4.2.4.1 TRAINING DETAILS

We obtained 5,971,552 unique human antibody heavy chain sequences with the *IGHV1-18* gene from the OAS database, with 2,000 and 10,000 sequences set aside for validation and test sets respectively and the remaining used for training.<sup>1</sup> We trained a deep manifold sampler on the training set with a constant learning rate of  $10^{-4}$  for 60K mini-batch steps with the batch size of 128. The model consisted of a two-layer transformer encoder and decoder, each with 8 heads and the total embedding dimension of 256 and feed-forward layer dimension of 1024. The last layer consists of a CRF for final sequence generation. The rest of the training procedure was the same as described in the original paper.

In addition, we also trained an autoregressive GPT-2 model using HuggingFace Transformers library v4.16.2 [176] on the same training set in order to demonstrate that the sampler-generated

---

<sup>1</sup>Only sequences with "Redundancy > 1" were retained.

sequences capture the amino-acid token distribution observed in the training set. The model consisted of 6 attention layers with 8 heads and a total embedding dimension of 512 and was trained with a constant learning rate of  $4 \times 10^{-4}$  for 25K mini-batch steps with batch size of 1024. The other parameters were set to the default values provided by the package.

#### 4.2.5 SAMPLING DETAILS AND RESULTS

For each sequence in the test set, we applied multi-segment preserving sampling for one iteration, preserving all non-CDR3 regions with four different  $\beta$  values of 0, 0.1, 0.5, and 0.9.<sup>2</sup>

Figure 4.3 and Figure 4.4 show the length and log-probability (GPT-2) distributions of the generated sequences with changes in CDR3 and the test data across all selected  $\beta$  values. The CDR3 length distribution of the generated samples matches the natural sequence length distribution for each value of  $\beta$ . The GPT-2 log-probability distribution of the samples has lower overall mean compared to that of the test distribution but is still within the same range. Both distributions vary only slightly with different values of  $\beta$ . These two results show the effectiveness of the sampling strategy for generating diverse antibody sequences, restricted to user-defined regions.

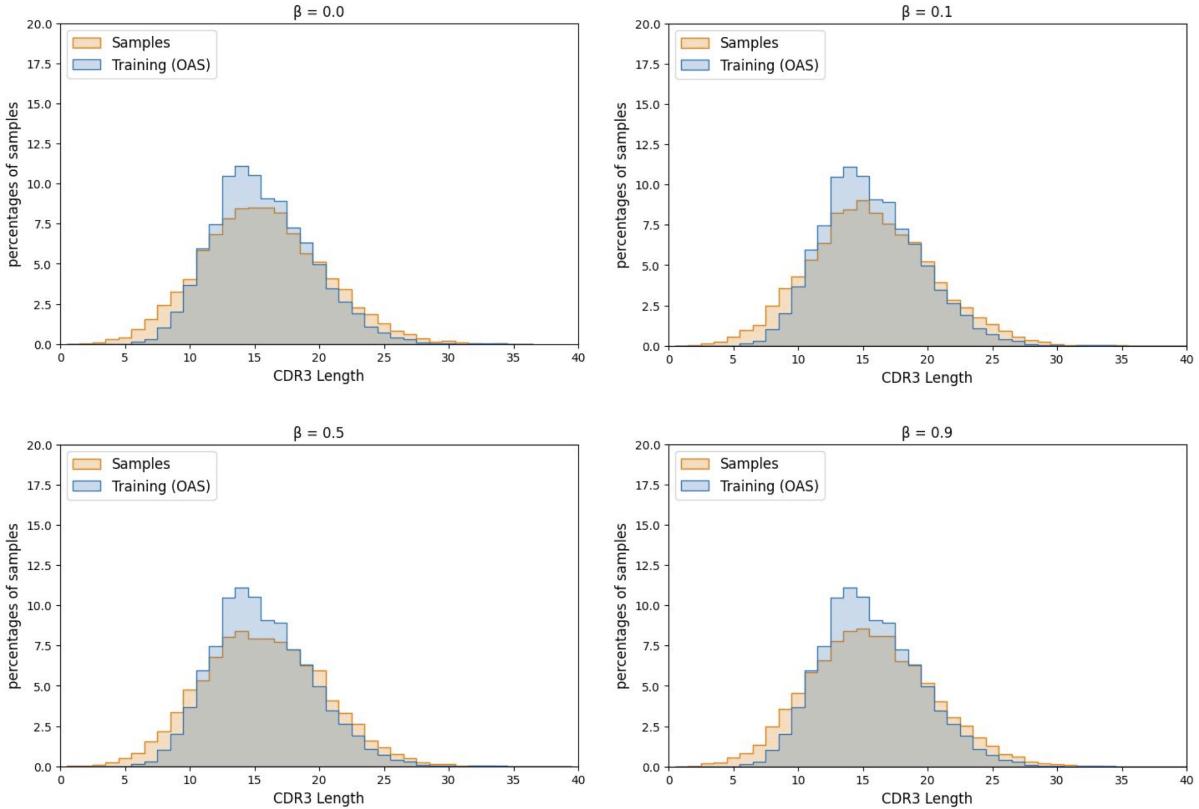
In Figure 4.5, we illustrate the distribution of the number of edits in the generated sequences relative to the input seed sequences, including substitutions, insertions, and deletions. The distributional mean increases slightly with higher values of  $\beta$ . For future work, we plan on a more systematic understanding of the effects of carry-over strength  $\beta$  on sample quality and diversity.

#### 4.2.6 CONCLUSION

We have proposed a sampling procedure for the deep manifold sampler that explicitly preserves designated segments of the input sequence, allowing variation to occur only in non-preserved regions. We find that this approach, multi-segment preserving sampling, is applicable to a number of design problems in the life sciences where we often want to use prior knowledge

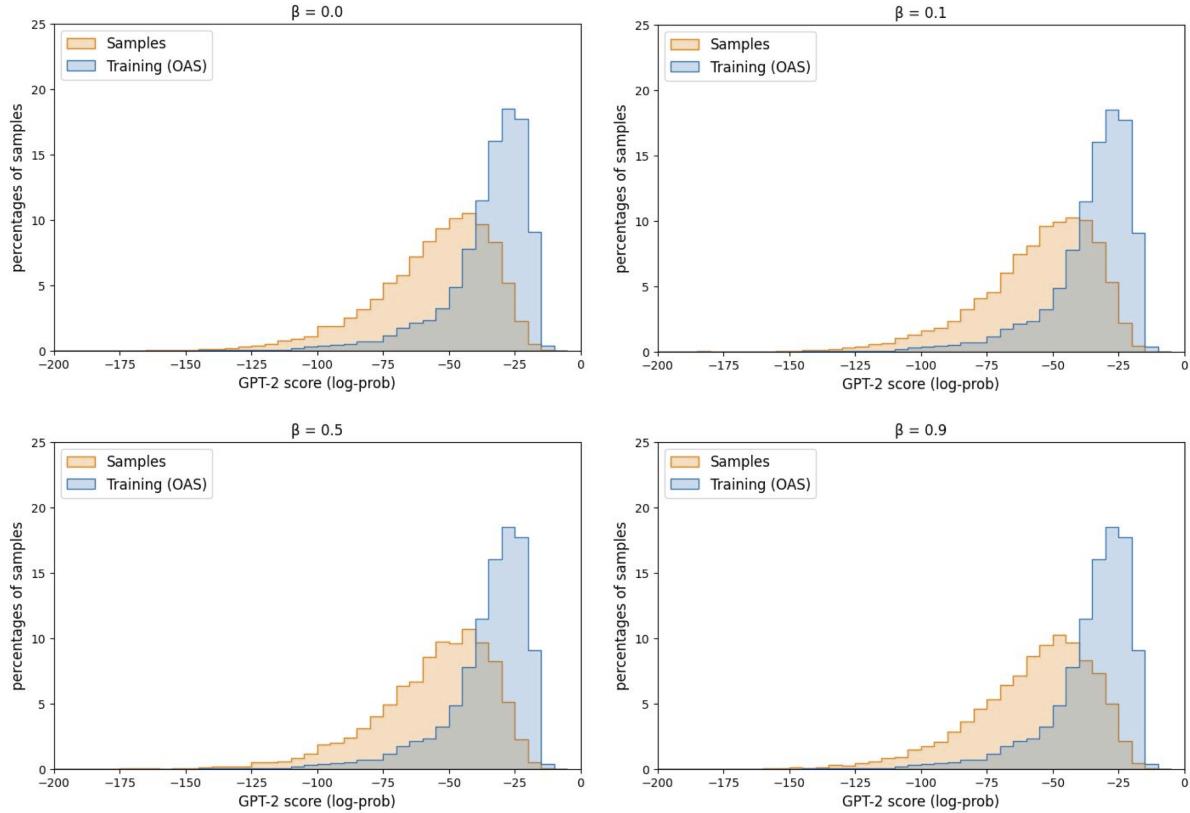
---

<sup>2</sup>The region annotation was obtained from the OAS data unit files.

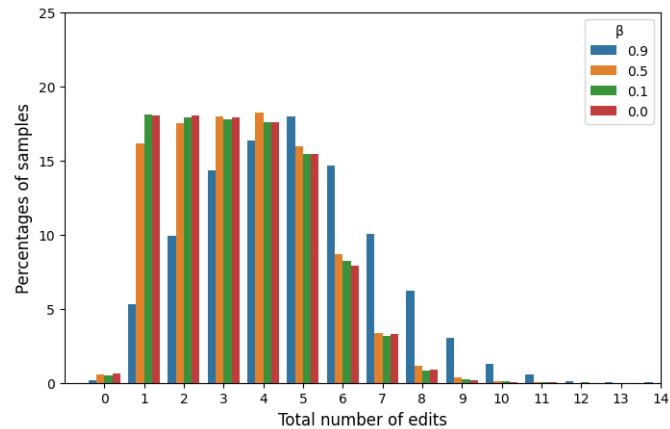


**Figure 4.3:** The normalized distribution of the CDR3 lengths of the deep manifold sampler-generated sequences (“Samples”) and the test set sequences (“Training (OAS)”) with four different  $\beta$  parameters. From Top Left, Clockwise: Samples were generated with  $\beta = 0, 0.1, 0.9$ , and  $0.5$ .

made available in these domains. With biological sequence design for example, we want to sample new, diverse designs that avoid perturbing well-conserved regions of the input. In this way, we demonstrate the utility of multi-segment preserving sampling by restricting sampling to the CDR3 regions of a collection of antibody heavy chains with the *IGHV1-18* gene and validating the resulting CDR3 designs against a separate GPT-2 model. As shown in Figure 4.4, the sampled CDR3 regions admit high log-probability scores by the GPT-2 model, providing evidence that the samples are plausible. Additionally, the CDR3 length distribution of the samples coincides with the observed length distribution in the training set, suggesting the model adequately captures the variability in non-preserved segment lengths, despite the lack of explicit provision during training. In future work, we will extend our exploration on the effect of the carry-over strength



**Figure 4.4:** The normalized distribution of the GPT-2 scores of the deep manifold sampler-generated sequences (“Samples”) and the test set sequences (“Training (OAS)”) with four different  $\beta$  parameters. From Top Left, Clockwise: Samples were generated with  $\beta = 0, 0.1, 0.9$ , and  $0.5$ .



**Figure 4.5:** The distribution of edit distances between generated samples and their seed sequences with varying settings of  $\beta$  (0, 0.1, 0.5, and 0.9).

$\beta$  in terms of sample quality as well as its usage in conjunction with the function predictor for guided sampling proposed in Gligorijević et al. [68].

### 4.3 RETROSPECTIVE

The Deep Manifold Sampler is a general approach to non-autoregressive guided sequence design. We present two steering techniques: function-guided design using a Gene Ontology term classifier and multi-segment preserving sampling. One future direction for improving the DMS technology is to emplace specialized tokens that encode various properties such as the taxon and function. Doing so would enable classifier-free guidance using multi-segment preserving sampling. Additionally, further evaluation using a similar distributional conformity approach to Chapter 3 would provide more concrete empirical evidence on the sample validity. Finally, further benchmarking against newly emerged, multi-step denoising models such as discrete denoising probabilistic diffusion [64, 65] and discrete flow-matching [177, 178] is necessary to justify the capabilities of the DMS.

# 5 | NATURAL LANGUAGE CONDITIONED PROTEIN DESIGN

## 5.1 INTRODUCTION

Recent studies [179–182] have shown that oprotein language models (PLMs) exhibit complex behavior in performative modeling of protein sequence, structure, function, and evolution [183–186]. These findings elicit interest in augmenting and controlling protein language model behavior using additional contextual information, such as functional labels, text, or structure [29, 30, 187–190]. The increasing interest in multimodal representations of proteins emphasizes the importance of investigating the modalities that, in addition to sequence, may provide contextual information that enriches the foundation model representation and result in performance improvements on downstream tasks.

In this work, we propose a framework for **conditioning** on **residue** and **global information** from text (CORGI) and evaluate the influence on model performance of text annotations on two generative modeling tasks. CORGI fuses a pre-trained protein language model (PLM) with a text embedding model using feature-wise linear modulation (FiLM) [191]. We postulate that residue-level and global conditioning enhance generation in complementary ways. Whereas global conditioning enables classifier-free guidance towards regions of sequence space enriched in a specific functions and properties, residue-level conditioning unlocks fine-grained control by beneficially

constraining the logits within annotated intervals. To investigate the contribution of natural language conditioning, we fine-tune CORGI models on a text-augmented dataset and benchmark their performance on two downstream generative modeling tasks: mutation effect prediction and *de novo* design. For mutation effect prediction, we propagate wildtype annotations onto mutants and evaluate the contributions of each annotation type compared to a baseline unconditional model trained on the same dataset. We also demonstrate the capability of CORGI to accommodate different tasks by showcasing the model’s performance on distinct selection assays of mutants with a common genotype. Benchmarking text-conditioned *de novo* samplers remains challenging due to the inherent imprecision of natural language prompts; it is often unclear whether generated samples truly reflect the intended concept. To address this, we develop unbiased benchmarks for sequence validity and language alignment, grounded in an established framework for assessing distributional conformity. Using our evaluation, CORGI models demonstrate competitive performance compared to other state-of-the-art natural-language conditioned models.

## 5.2 RELATED WORK

**Text conditioned protein modeling:** Previous studies have investigated the utility of text features in both predictive and generative modeling. Protein function prediction models such as DeepText2GO [192] and OntoProtein [193] demonstrate enhanced performance with multi-modal inputs from manuscript abstracts and gene ontology (GO) term embeddings. Text annotations and protein classifications often exhibit high mutual information. Duan, et al [190] also observed the mutual information phenomenon and studied the effects on downstream classification tasks of annotation-augmented sequence representations, finding that not all annotations improve performance.

Multi-modal contrastive learning models such as ProTrek [187] and ProteinCLAP [30] adapt

CLIP [194] to learn aligned embedding spaces of text and protein. These learned features then facilitate text-to-protein generation either by latent optimization or internally scoring designs [29].

Text-conditioned generative models span a wide spectrum of ideas and implementations. Galactica [195] is a multi-purpose autoregressive large language model trained on scientific text and molecular data. MP4 [32] is a multi-modal encoder-decoder model uses cross-attention to perform text-to-protein translation. Finally, observing that structure space is a seemingly lower-dimensional index of function, Pinal [29] mediates text-to-protein generation via a generated 3Di encoding [100].

**Text-conditioned generative model evaluations:** Evaluating the performance of natural language conditioned models is challenging because prompts often express soft, ambiguous concepts rather than discrete, well-defined labels. While the subject area has not yet reached consensus on the evaluation metrics that should be adopted for this task, benchmarking typically entails two criteria: validity and language alignment. Validity metrics assess whether a given protein sequence is biologically realistic. Multiple works [29, 196] have proposed the pLDDT score [3] as a proxy for foldability. Other coarse measurements such as amino acid distributions may be employed to further characterize the sequence validity.

Language alignment measures the coherence between generated text and sequence. Aligned multimodal embedding spaces such as ProTrek [29] are designed to produce a similarity score that reflects this alignment. The MP repository [196] provides an estimate of language alignment via the NLM-sim, which measures the text-to-text similarity between the input prompt and the output of ProtNLM [197].

### 5.3 METHODS

We consider CORGI in two isolated conditioning settings using residue-level and global text annotations. We investigate two use cases: (1) Incorporating residue-level annotations from site-

specific protein family models, such as HMM signature for homologous protein in InterPro [57], will sharpen the per-residue logit distributions along annotated intervals. Furthermore, after pre-training the sharpened logit distributions are useful for downstream applications such as mutation effect prediction, given that the signal provided by the annotation may contribute to a more nuanced likelihood estimate. (2) Sequence-wide annotations may be leveraged to directly steer design via soft prompting. Contrary to prior existing methods, we investigate whether conditionally sampling using embedding outputs of a pre-trained text encoder enables targeted molecular programming.

As position-specific annotations outnumber sequence-wide ones by orders of magnitude, we treat the two annotation types separately, constructing sequence datasets of either type and individually evaluating trained CORGI models in different settings.

### 5.3.1 EVALUATION

We characterize the influence of text annotations on CORGI models by measuring the performance on two downstream generative modeling tasks: mutation effect prediction and molecular programming. Mutation effect prediction measures the model’s capability to propose meaningful, non-deleterious edits to a protein sequence. Molecular programming is a *de novo* generation task in which output samples must be not only realistic, but also align with the input text prompt.

#### 5.3.1.1 MUTATION EFFECT PREDICTION BENCHMARK

Mutation effect prediction models assess the impact on fitness of mutations to a reference sequence. Fitness can be measured in several ways including stability, binding affinity, activity, expression, or organismal survival. Competitive performance in mutation effect prediction implies the model is capable in making quality, biologically meaningful edits. The ProteinGym [33] is a collection of deep mutational scanning (DMS) assays that assign experimentally determined fitness values called DMS scores to sequence variants. Following previous studies, we define the

mutation effect score as the model’s log-likelihood estimate of a given mutant. To characterize performance, we compute the Spearman-rank correlation between the mutation effect and the DMS score and calculate the average correlation over per taxonomic class and per DMS selection type.

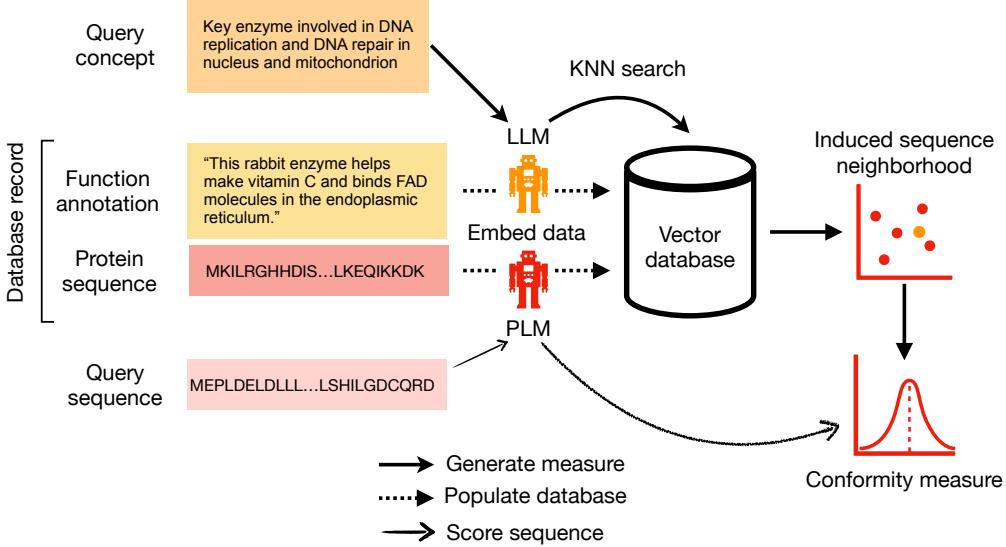
We postulate that conditioning on residue-level and global annotations during training will improve predictive performance on the ProteinGym assessment. To test this hypothesis, we propagate wild-type annotations onto mutants. For substitution assays, direct transfer of these annotations is sufficient. For indels, we adjust the annotation intervals according to the alignment between wildtype and mutant. We choose to transfer annotations based on the intuition that non-deleterious mutations will be tolerated by the model via a correspondence between the conditioning information and the sequence. For deleterious mutations where the wildtype annotation no longer applies, the mismatch in conditioning will decrease the overall likelihood.

#### 5.3.1.2 MOLECULAR PROGRAMMING BENCHMARK

Constructing the molecular programming benchmark requires two inputs: a database of annotated protein records with a vector index of their function documents and a collection of test prompts. Our database is populated with SwissProt [53] records and is indexed by nomic-embed-text-v1 [198] text representations. The MP repository [196] contains a collection of test prompts as well as generated sequences from MP4 [32].

#### 5.3.1.3 CONCEPT-CONDITIONED DISTRIBUTIONAL CONFORMITY SCORE

To assess the model’s capability in generating aligned sequences to a text concept, we instantiate a conformal transducer system [199, 200] based on the distributional conformity score (DCS) [201]: the *concept-conditioned* distributional conformity score (C2DCS), that measures conformity to a specific reference distribution of sequences with an aligned concept. The schematic of the score construction is depicted in Figure 5.1. We define the “induced sequence neighbor-



**Figure 5.1:** Schematic of the concept-conditioned distributional conformity score. Protein records are stored in a vector database indexed by the annotation text representation. Sequence representations of the nearest annotations are used to parameterize the conformity model. The conformity model in turn scores generated sequences by their sequence embeddings.

hood” (ISN)  $\mathcal{N}(z)$  as the set of *protein sequences* retrieved by querying the  $k$ -nearest neighbors to the annotation description representation  $z$  from a vector database of document embeddings. The induced sequence neighborhood embeddings are used to parameterize a conformity scoring model. First, we divide the ISN into disjoint training and calibration sets. The training set is used to fit a density estimator  $A_{\mathcal{N}(\cdot)}$  which serves as the concept-conditioned conformity measure for the ISN of  $z$ . For each element of the calibration set  $\mathcal{N}_{\text{cal}}(z) = \{r_i \in \mathbb{R}^d \mid i = 1, \dots, m\}$ , we compute its conformity measurement  $\alpha_i = A_{\mathcal{N}(z)}(r_i)$ . For a query sequence  $s_q$ , we compute its sequence embedding  $r_q$  and in turn its conformity measurement  $\alpha_q = A_{\mathcal{N}(z)}(r_q)$  and calculate the C2DCS:

$$\text{C2DCS}(\alpha_q; \alpha_1, \dots, \alpha_m) := \frac{1}{m} \sum_{i=1}^m [\alpha_i < \alpha_q]. \quad (5.1)$$

More information on the construction and specific instantiation of the C2DCS in this work can be found in Appendix C.3.

Whereas other evaluation criteria were tightly coupled to point-evaluations of underlying models with subtle biases, our method is de-coupled from any specific model and relies on its output distributions rather than point-wise evaluations. This gives way to a more versatile metric that naturally accounts for systematic biases or preferences in the underlying criteria.

### 5.3.2 TEXT-AUGMENTED TRAINING DATA

The CORGI framework consumes protein sequences with global and residue-level text annotations. Every annotation is a tuple of the form  $(s, t, d)$ , where  $[s, t]$  is an index interval stating the region along which the text description  $d$  applies. Global annotation intervals are trivially  $[0, \ell]$  where  $\ell$  is the length of the annotated sequence. The text description  $d$  comes in the form of either a natural language document or discrete label code. For the latter, we map the label code to its long-form explanation. We embed each annotation description using a pre-trained text encoder.

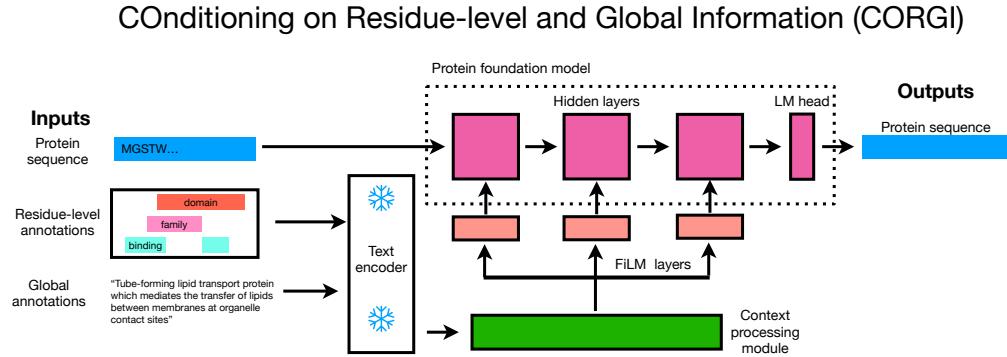
We collected 510,747 protein sequence records from the SwissProt database [53] with either text annotations, discrete labels, or residue-level annotations. Global annotations were restricted to relevant categories such as function, structural subunit, catalytic activity, cofactors, pathway, Gene Ontology (GO) [11] terms, Enzyme Commission (EC) [202] numbers, InterPro labels [57], and other miscellaneous comments. For each collected record, we create a description document by combining all of its text annotations and discrete label descriptions together with a template.

For each residue-level annotation, we use a long-form description of its “type” (domain, family, binding site, active site, etc) rather than the label itself. These annotations function as coarse yet still informative markers of regions of interest. Each long-form description is generated using a large language model by prompting it to expand on the term in the context of protein structure and function.

We embed all text descriptions and region-level explanations using nomic-embed-text-v1 [198]. Region explanation vectors are compiled into an embedding matrix  $E \in \mathbb{R}^{N \times d}$  such that  $E_i$  corre-

sponds to the residue label type  $y_i \in \{y_0, \dots, y_{N-1}\}$ .

### 5.3.3 CORGI MODEL DESCRIPTION



**Figure 5.2:** CORGI model schematic.

CORGI (Figure 5.2) fuses a text conditioning network with a protein foundation model via feature-wise linear modulation (FiLM). Text annotations are processed through a pre-trained text encoder and subsequently incorporated into the hidden blocks of a pre-trained PLM via gain and bias terms [191]. To process residue-level annotations, we construct a matrix  $C_{\text{residue}} = AE^T$  where  $E \in \mathbb{R}^{N \times d}$  is the residue label type embedding matrix described in Section 5.3.2 and  $A \in \{0, 1\}^{\ell \times N}$  is such that  $A_{i,j} = 1$  if residue index  $i$  is annotated with the label  $j$ . The global annotation vector is row-expanded to a matrix  $C_{\text{global}}$  the same dimensions as  $C_{\text{residue}}$ . The final conditioning representation  $C = C_{\text{global}} + C_{\text{residue}}$  is subsequently transformed through the context processing module.

For our experiments, we fine-tuned the baseline ESM2-150M model [203] on the text-augmented dataset described in Section 5.3.2. Additionally, we provide a study of CORGI training at scale and task-specific inference in Appendix C.6 using ProGen-2 [94].

We train two CORGI variants: one as a masked language model (MLM) using the cross entropy objective function and the other as a walk-jump sampler (WJS) formulated according to the neural empirical Bayes (NEB) framework [20].

Each model is regularized by annotation dropout (with  $p_{dropout} = 0.2$ ) to accept no additional context as input. By virtue of this dropout, the user is able to specify to either model a contrastive weight  $w$  that considers both unconditional and conditional gradients as shown in Equation 5.2.

$$s_\phi(y, z, w) = (1 + w)g_\phi(y, z) - wg_\phi(y, \emptyset) \quad (5.2)$$

We refer the reader to Appendix C.4.1 for more in-depth description on the architecture, training, and sampling. A brief analysis justifying the choice of the noise level  $\sigma$  is provided in Appendix C.5.

## 5.4 RESULTS

We trained two CORGI models based on ESM2-150M and benchmarked them on two downstream generative modeling tasks. We evaluate the impacts of text conditioning on mutation effect prediction with the text-augmented ProteinGym (Section 5.3.1.1) and the distributional conformity of generated molecular programming samples using the Molecular Programming repository (Section 5.3.1.2).

### 5.4.1 MUTATION EFFECT PREDICTION

Conditional generative models of protein sequence should not only produce plausible samples conforming to the concept of interest, but also make biologically meaningful edits. To test this desired trait, we evaluate each model on the ProteinGym by computing mutation effect scores of each mutant. For the masked language model versions, we follow previous works and define the mutation effect score as the model’s total log-likelihood estimate of the variant. For the model’s optimized with the NEB loss (Equation C.3), define the mutation effect score by the relative change

in gradient norm between the mutant and wildtype:

$$\Delta_\phi(y_{\text{mut}}, y_{\text{wt}}) = \frac{\|g_\phi(y_{\text{mut}})\| - \|g_\phi(y_{\text{wt}})\|}{\|g_\phi(y_{\text{wt}})\|} \quad (5.3)$$

where the mutant and wildtype  $y_{\text{mut}}$  and  $y_{\text{wt}}$  smoothed by additive isotropic Gaussian noise with variance  $\sigma$ .

Tables 5.1 and 5.2 provide the Spearman averages across selection and taxon type respectively. Notably, the masked language model CORGI variant outperforms the baseline by a large margin in every taxon category and all selection types apart from binding, where the performance gain is more modest.

**Table 5.1:** Performance comparison of the baseline to each CORGI model and setting and averaged per (selection type, mutation type) pair. The columns **Model** and **Setting** indicate the underlying CORGI model and inference setting. The acronyms MLM and NEB stand for Masked Language Model and Neural Empirical Bayes respectively. “Baseline” stands for the masked language model trained as a control without annotations on the same dataset as the other CORGI models. The abbreviations “R”, “G”, and  $\emptyset$  designate whether residue-level, global, or no annotation features were provided at inference time. Each column (**Expression**, **Organismal Fitness**, **Stability**, **Activity**, and **Binding**) represents the DMS selection type and is followed by the mutation type. “Indels” stand for the edits that change the original sequence length by at least one insertion or deletion. “Subs” represent substitutions, or mutations that do not change the original sequence length.

<b>Model</b>	<b>Setting</b>	<b>Expression</b>		<b>Organismal Fitness</b>		<b>Stability</b>		<b>Activity</b>		<b>Binding</b>
		Indels	Subs.	Indels	Subs.	Indels	Subs.	Indels	Subs.	Subs.
MLM	Baseline	0.266	0.276	0.161	0.271	0.272	0.297	0.275	0.254	0.262
MLM	R+G	0.337	<b>0.398</b>	0.335	0.344	0.471	0.465	0.506	<b>0.431</b>	0.278
MLM	R	0.323	0.399	0.334	0.341	0.468	0.465	0.503	0.432	<b>0.290</b>
MLM	G	0.359	0.391	<b>0.372</b>	0.345	0.473	0.465	0.497	0.423	0.278
MLM	$\emptyset$	<b>0.361</b>	0.396	0.329	<b>0.346</b>	<b>0.477</b>	<b>0.470</b>	<b>0.509</b>	0.427	0.285
NEB	R+G	0.005	0.009	0.019	0.000	0.050	0.001	-0.016	-0.003	-0.009
NEB	R	-0.003	-0.005	0.011	0.003	0.051	0.000	-0.004	0.009	0.012
NEB	G	-0.016	0.000	-0.005	0.001	0.048	0.003	-0.091	0.001	-0.007
NEB	$\emptyset$	0.037	0.006	-0.010	0.001	0.051	-0.003	0.048	0.000	0.000

Providing any specific combination of annotation at inference time does not appear to have

a measurable effect on performance, suggesting that perhaps the performance benefits originate from modifications adjustments to the attention heads during training. We also note that the walk-jump sampler’s mutation effect score does not correlate with mutation effects.

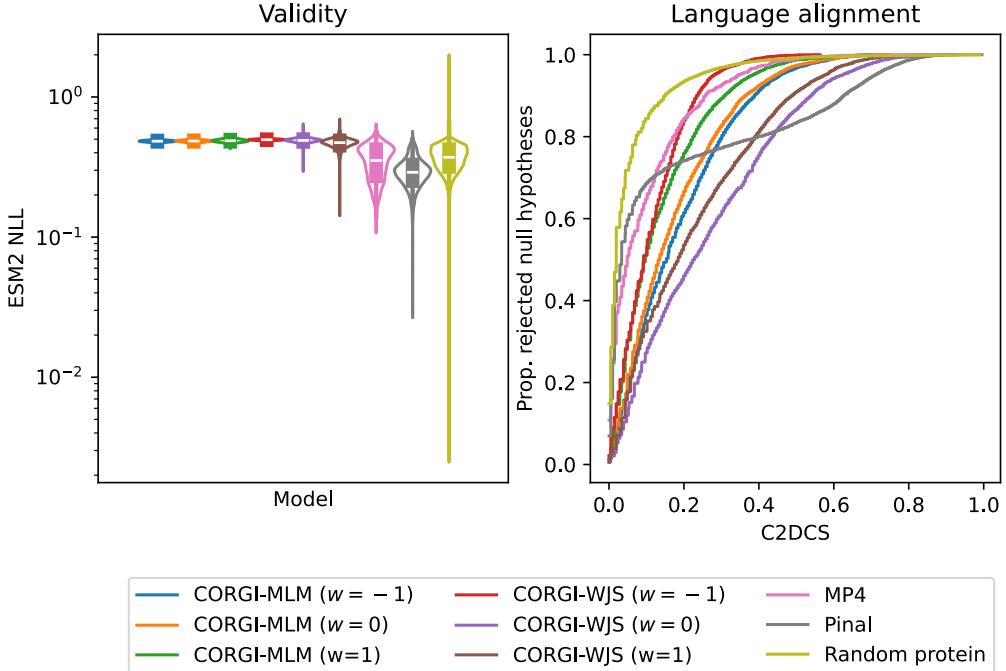
**Table 5.2:** Performance comparison of the baseline to each CORGI model and setting and averaged per (taxon, mutation type) pair. Each column (**Eukaryote**, **Human**, **Prokaryote**, and **Virus**) stand for the taxonomic classification of each wildtype provided by the ProteinGym. See Table 5.2 for a complete description of the **Model** and **Setting** columns as well as the abbreviations therein.

<b>Model</b>	<b>Setting</b>	<b>Eukaryote</b>		<b>Human</b>		<b>Prokaryote</b>		<b>Virus</b>	
		Indels	Subs.	Indels	Subs.	Indels	Subs.	Indels	Subs.
MLM	Baseline	0.267	0.318	0.286	0.301	0.288	0.275	0.137	0.114
MLM	R+G	0.508	<b>0.439</b>	0.473	0.446	0.401	0.358	0.331	0.221
MLM	R	0.494	0.432	<b>0.489</b>	<b>0.448</b>	0.385	<b>0.372</b>	0.312	0.201
MLM	G	<b>0.516</b>	0.436	0.466	0.443	0.417	0.352	<b>0.353</b>	<b>0.234</b>
MLM	$\emptyset$	0.505	0.436	0.479	0.447	<b>0.422</b>	0.370	0.339	0.218
NEB	R+G	0.066	-0.002	0.032	0.001	-0.004	0.001	0.087	-0.001
NEB	R	0.071	-0.003	0.038	0.006	-0.018	-0.003	0.076	0.008
NEB	G	0.049	0.001	0.037	0.004	-0.013	-0.005	0.076	-0.001
NEB	$\emptyset$	0.066	-0.007	0.039	0.004	-0.007	-0.003	0.086	-0.006

#### 5.4.2 TEXT-CONDITIONED *DE NOVO* DESIGN

Using the text-conditioned walk-jump sampler, we sample one trajectory per prompt in the Molecular Programming repository using  $w = -1, 0$ , and  $1$  to test performance as an unconditional, conditional, and contrastive sampler respectively.

We evaluate the validity and language alignment of the highest ranking generated samples, comparing to Pinal [29] and MP4 [32] and display the results in Figure 5.3. We note that the CRATE-WJS model outperforms MP4 and is competitive with Pinal. One possible explanation for the higher proportion of exceptionally high distributional conformity scores in Pinal is the choice to incorporate a structure intermediate during generation.



**Figure 5.3:** *Left:* Validity of samples assessed by proxy via the ESM2-35M negative log-likelihood (NLL). Smaller NLL values correspond to more valid examples. The NLL distributions for every model stand within an acceptable range (i.e., below 1). *Right:* Empirical cumulative distributions of concept conditional conformity scores. Given a C2DCS value  $\alpha$  on the  $x$ -axis, the corresponding value on the  $y$ -axis may be interpreted as the fraction of examples that reject the null hypothesis that the samples are exchangeable with the reference distribution at significance level  $\alpha$ .

#### 5.4.3 DISCUSSION

CORGI is a fine-tuning framework that enables a model to consume external text-based conditioning information during training and inference via feature-wise linear modulation. We trained CORGI variants using the masked language modeling (MLM) and neural empirical Bayes objective functions and evaluated their performance on downstream protein design tasks.

We find that the neural empirical Bayes and masked language modeling variants of CORGI perform well on different downstream tasks. Whereas the masked language model excels at mutation effect prediction, the walk-jump sampler is better at *de novo* generation. We intuit that this performance discrepancy between the CORGI variants is due to the theoretical difference in objective functions. Whereas the MLM cross-entropy loss is motivated by matching individual

token-level distributions, the NEB loss is focused on globally matching the distribution gradients. These findings motivate future work in combining the two model variants to deploy a two-step *de novo* design pipeline. The first step is a walk-jump sampling trajectory that navigate towards a specific region of sequence space enriched in the concept of interest. The second step is a local refinement step that updates the initial design candidates with iterative unmasking.

Another promising direction is in target site-specific redesign, or sequence in-painting. Given a local specification for a specific sequence interval, we are interested in using CORGI to resample this region. However, provided that the MLM variant did not benefit from any specific inference-time provision of residue-level or global features, we expect this to perhaps be the case as well for the walk-jump sampler. This phenomenon warrants further investigation that may reveal potential methods to correct it. For example, learned token-level, position-specific representations such as those learned in the scaling experiment provided in Appendix C.6 may alleviate this issue.

Finally, given that text representations are known to exhibit semantic clustering favorable to concept algebra via vector arithmetic [204], we are curious to test whether this phenomenon enables generation of samples imbued with or restricted from multiple concepts.

## 5.5 CONCLUSION

In this work, we investigated the contribution of both residue-level and global text annotation representations to downstream generative modeling tasks in protein design and engineering. We introduce the CORGI approach and demonstrate its improvement over baseline methods in both mutation effect prediction and *de novo* sampling. We modified the distributional conformity score [201] to measure sequence plausibility conditioned on a specific concept. Training two different CORGI variants with masked language modeling and neural empirical Bayes objective functions reveals interesting results in the benefits of text for either model in both downstream tasks. Additionally, we demonstrated training and inference at scale. We defined a greedy invari-

ant for annotation subset selection and find that specific selection types benefit from only certain subsets of annotations. Our findings indicate promising directions in not only multi-modal, text-conditioned design, but also conditional evaluation for abstract concepts.

## 6 | CONCLUSION

Proteins are molecular machines that participate in virtually every biosynthetic process. Due to their apparent ubiquity, designing protein sequences is a long sought after grand challenge with many potential applications in areas such as therapeutic discovery and agriculture. The combinatorially expansive design space of protein sequences prohibits exhaustive evaluation, therefore calling for intelligent methods that propose realistic and functional protein sequences suited for a specific task. Protein design and engineering methodologies have sophisticated over the last 50 years from site-specific mutagenesis, to directed evolution, and finally with mechanistic computational modeling. Now with the advent of larger sequence repositories and highly expressive, data hungry technologies such as deep neural networks, the protein design challenge is suited to be cast as a generative modeling problem. However, as with any generative modeling endeavor, new challenges emerge in steering sampling in order to produce diverse, yet on-target sequences. This dissertation describes methods that address these challenges by presenting not only novel samplers, but also versatile distributional evaluation criteria.

In Chapter 3, we introduce the discrete walk-jump sampler (dWJS), a generative model for antibodies. We show with a distributional conformity analysis and other accompanying metrics that the Neural Empirical Bayes methodology is applicable to protein family diversification and targeted control of sampling along the sequence. The dWJS is trained from scratch, however since then follow up works have leveraged pre-trained models to enhance model capacity. Despite the strong performance of the dWJS, the method was only tested with Aho-aligned antibody

sequences, motivating the exploration of more versatile methods.

In Chapter 4, we propose the Deep Manifold Sampler (DMS) for generalized protein design. Alongside a denoising autoencoder that supports de-corrupting indels, the DMS uses an length posterior to propose novel lengths to denoised proteins. We demonstrate function guided design using a function predictor trained on the model’s hidden representation. Notably, the DMS successfully transitions between proteins of distinct functions. Using the DMS, we also proposed another sampling methodology – multi-segment preserving sampling (MSPS) – that allows the injection of expert knowledge to preserve specific regions of the sequence. MSPS is again applied to antibody sequences and shown to produce in-distribution complementarity-determining regions.

Using the learnings of Chapter 3 and Chapter 4 in generative modeling and steerable generation, we then consider the task of text-driven protein design and engineering in Chapter 5. Natural language is utile in expressing complex concepts that do not adhere to a discrete label scheme or otherwise, making it an advantageous modality for design. We propose conditioning on residue-level and global information from text (CORG), a fine-tuning framework that enables any pre-trained model to accept latent text-conditioning information at either scale. We show that CORGI can be applied to diverse model types including causal language models and walk-jump samplers and demonstrate the performance of trained CORGI models on mutation effect prediction and molecular programming. We found that CORGI not only enhances mutation effect prediction performance, but also enables task-specific annotation selection using a greedy algorithm. We also showed that CORGI is competitive in generating in-distribution *de novo* designs from text prompts. To do so, we adapted the distributional conformity score first proposed in Chapter 3 to support text concept conditioning. This concept conditional distributional conformity score is more robust and flexible than previous language alignment metrics because it decouples the reliance on point-wise evaluations of specific models. We found the text-conditioned CORGI model to perform competitively with state-of-the-art contemporary models using this

metric. We also observed the benefit of structure conditioning, leading to a concrete next step in CORGI fine-tuning of walk-jump samplers to explore structure-informed models as well.

The body of work described in this thesis serves as a methodological prescription for deep-learning driven conditional and unconditional protein design. We explored the utility of classifier-guided and classifier-free technologies for this purpose and investigated extensions to natural language conditioning. This work is timely as the emergence of large language models as powerful tools to enhance productivity in scientific research has only just begun. Indeed, while these methods are not general solutions to molecular modeling or understanding, they are effective in statistically sampling complex, natively discrete sequence manifolds. Whereas large language models are generalist, the methods described in this dissertation are specialized and capable. As the field progresses, especially with multi-modal integrations such as text-driven design, it will be interesting to use methods such as these to function as tools for not only human researchers, but also artificially intelligent agents. A possible future for accelerated molecular design is to incorporate bespoke sampling methodologies as tool calls to agent-based scientific models. Agents with access to such tools may be able to research, rank, and produce candidates that human experts can then supervise and provide feedback.

# A | SUPPLEMENTARY MATERIAL FOR CHAPTER 3

## A SUPPLEMENTARY MATERIAL FOR CHAPTER 3

### A.1 NETWORK ARCHITECTURES AND TRAINING DETAILS

#### A.1.1 DISCRETE WALK-JUMP SAMPLERS

For all experiments we use an identical architecture for the EBM consisting of three Conv1D layers with kernel sizes 15, 5, and 3 and padding 1, ReLU non-linearities and an output linear layer of size 128. The denoising model is a 35-layer ByteNet [205] architecture with a hidden dimension of 128, trained from scratch. The Bytenet architecture has been shown to perform competitively with transformers for protein sequence pretraining tasks [80]. All models were trained with the AdamW [206] optimizer in PyTorch [207]. We used a batch size of 256, an initial learning rate of  $1 \times 10^{-4}$ , and trained with early stopping.

**TRANSFORMER IMPLEMENTATION OF DWJS DENOISING MODEL** In addition to the ByteNet implementation, we implemented a transformer-based architecture for the denoising network, with 12 hidden layers, 8 attention heads, a dimension of 2048 for the feed-forward layers, 256 features

in the encoder/decoder inputs, and SiLU activations. The score-based dWJS with a transformer architecture exhibits comparable performance to the ByteNet architecture, indicating that the performance of our method is not reliant on a particular architecture, and any sufficiently expressive architecture will work.

**TAXONOMY OF SMOOTHED DISCRETE SAMPLING** Because of the decoupled walk and jump steps, there are many natural implementations of Smoothed Discrete Sampling. Empirically, we find that Algo. 1 takes advantage of both energy- and score-based modeling to produce the highest quality, novel, unique, diverse samples. Four natural choices for performing sampling, which arise from different combinations of energy- and score-based parameterizations, are summarized in Table A.1. *Discrete Walk-Jump Sampling* refers to walking with the EBM,  $f_\theta(y)$ , and jumping with the denoising network,  $g_\phi(y)$ . Score-based dWJS uses  $g_\phi(y)$  for both walking and jump steps. The Deep Energy Estimator Network (DEEN) [125] approach uses a denoiser that is trained by taking the derivative of an energy and using the same learning objective as Eq. 3.5. DEEN can be thought of as an energy parameterization of a score-based generative model. Finally, dWJS-EBM uses an EBM for sampling and the gradient of the energy,  $\nabla f_\theta(y)$ , for denoising. Empirically, we find that the most performative method in terms of efficiency, sample quality, and diversity is the EBM walk and denoiser jump, which we refer to as *Discrete Walk-Jump Sampling*.

**Table A.1:** Smoothed Discrete Sampling implementations.

Model	Walk (sampling)	Jump (denoising)
dWJS (energy-based)	EBM	Denoiser
dWJS (score-based)	Denoiser	Denoiser
Deep Energy Estimator Network	Denoiser (energy)	Denoiser (energy)
dWJS-EBM	EBM	EBM

### A.1.2 dWJS STABILIZES AND SIMPLIFIES TRAINING

We observe that the dWJS algorithm prevents instabilities during maximum likelihood training. EBMs commonly exhibit issues with training stability and divergences in the energy, due to the energy landscape becoming too complicated to sample. Noising the data provides strong regularization that prevents overfitting and instabilities. This is seen over a range of noise levels  $\sigma \in [0.5, 4.0]$  for EBMs trained over 3,000 steps. Training instabilities recur for  $\sigma < 0.5$ . We investigate the effects of discarding many of the techniques for improved EBM training that, while introduced to ameliorate challenges with EBMs, also introduce complexities that make EBMs brittle, inflexible, and difficult to optimize. In particular, we discard the replay buffer, the  $\ell_2$  norm penalty loss term to regularize the energies, Metropolis rejection sampling, and time step annealing. We use the Langevin MCMC algorithm (Algo. 10) from [208] and eliminate the need for careful hyperparameter finetuning;  $\sigma$  is the only free hyperparameter in dWJS.

### A.1.3 DIFFUSION BASELINES

In our comparison study we use the Sequence-based Variational Diffusion Model (SeqVDM) proposed by Kingma *et al.* [119], adapted for protein sequence data. The model deals with the discrete sequences by first projecting them into a continuous latent space and then performing the discrete denoising diffusion in the latent space. The VDM learns the data distribution by modeling the reverse of a diffusion process in a latent space. In all our experiments we used  $T = 1000$  steps with the fixed noise schedule  $\gamma_{min} = -13$  and  $\gamma_{max} = 5$ . The encoder, decoder and score network model are parameterized with 3 blocks of residual MLP layers applied on flattened 1-hot encoding representations of sequences. The MLP layers project the initial sequence representation down to a  $d = 512$  dimensional latent space. The model is simultaneously trained to optimize the diffusion loss (i.e., the score-matching loss) and the sequence reconstruction loss. SeqVDM is trained on paired OAS with the AdamW optimizer and the initial learning rate of  $2 \times 10^{-4}$  for 50 epochs.

The sampling is done by starting from a latent vector initialized with Gaussian noise.

#### A.1.4 LANGUAGE MODEL BASELINES

We generate samples from IgLM using the prompt given in Appendix A.4. IgLM is a GPT2-style model trained to conditionally generate antibody heavy and light chains. It is therefore a strong autoregressive baseline for antibody design.

To mimic the *ab initio* generation task presented in Table 3.2, we increase the masking percentage of the masked language model ESM2 [121] as high as possible (40%) and infill validation set sequences to generate new samples. Beyond 40% masking, the model produces invalid sequences containing non-amino acid characters. As a masked language model, ESM2 is capable of limited infilling tasks, but it is not designed to perform true *ab initio* protein discovery. Indeed, it does not generate antibody-like sequences, and the high  $E_{\text{dist}}$  and IntDiv scores are therefore meaningless. We include it purely as a familiar and powerful general protein language model baseline to show the gap in performance between a general, pre-trained protein MLM and our methods.

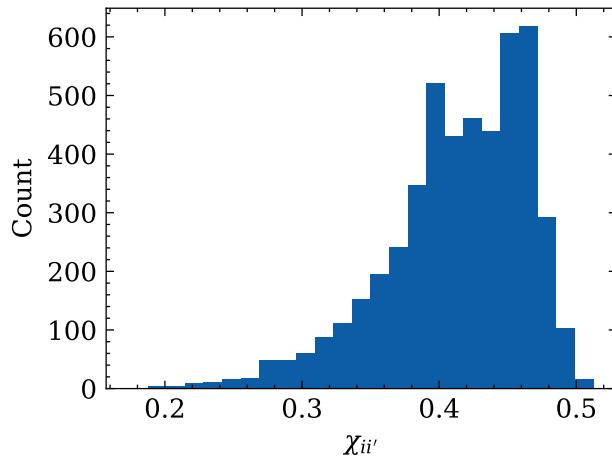
#### A.1.5 EFFECT OF CHOICE OF $\sigma$

In Table A.2 we show sample quality results for score-based dWJS as a function of the noise level,  $\sigma$ . We see that  $\sigma = 0.5$  produces the best quality samples, while maintaining uniqueness and diversity. Setting the noise too low ( $\sigma = 0.1$ ) leads to samples that do not capture the biophysical property distribution of the training set (high  $W_{\text{property}}$ ) and have extremely high edit distances from the training set ( $E_{\text{dist}} > 120$ ). This is because at low  $\sigma$ , the density is not smoothed and sampling does not perform well. At extremely high noise level ( $\sigma = 3$ ), the samples more closely match the training distribution, but they start to show signs of “mode collapse” (lower average edit distance to the training set and lower internal diversity), because the density is over-smoothed.

**Table A.2:**  $\sigma$ , ablikeness metrics, uniqueness, diversity for score-based dWJS.

$\sigma$	$W_{\text{property}} \downarrow$	$\text{Unique} \uparrow$	$E_{\text{dist}} \uparrow$	$\text{IntDiv} \uparrow$
0.1	0.378	<b>1.0</b>	120.6*	60.0
0.5 (From Table 3.2)	<b>0.065</b>	<b>1.0</b>	<b>62.7</b>	<b>65.1</b>
3.0	0.130	0.995	44.2	30.0

### A.1.6 ESTIMATION OF $\sigma_c$ HYPERPARAMETER



**Figure A.1:** Histogram of  $\chi_{ii'}$  values for random samples from the paired observed antibody space Olsen, Boyles, and Deane [2] dataset.

## A.2 ADDITIONAL ALGORITHMS

### A.2.1 GRADIENT FLOW ENABLES LOCAL MINIMA FINDING

We define the *gradient flow* as  $y'(t) = -\nabla \log f(y(t))$ , where sampling is performed by following the flow of the gradient of the probability density function in a deterministic dynamics, rather than stochastic Langevin dynamics. We initialize sampling from noise at  $t = 0$ ,

$y_0 = \varepsilon_0$ ,  $\varepsilon_0 \sim \mathcal{N}(0, I_d)$ , and sample noisy samples  $y'(t)$  following the gradient flow. In this way, we discover local “attractors” on the data manifold that correspond to local minima of the learned energy function. The algorithm for discrete gradient flow is given in Algo. 3.

---

**Algorithm 3:** Discrete gradient flow

---

**Input:** Denoiser,  $g_\phi(y)$ , energy-based model,  $f_\theta(y)$

**Output:** Denoised samples  $\hat{x}(y, t)$

```

1  $y_0 \sim \mathcal{N}(0, \sigma^2 I_d)$ 
2 for  $t = 0, \dots, T - 1$  do
3    $y_{t+1} \leftarrow y_t - \delta \nabla_y f_\theta(y_t)$ 
4 end
5  $\hat{x}_T \leftarrow y_T + \sigma^2 g_\phi(y_T)$ 
6 return  $\arg \max \hat{x}_T$ 
```

---

### A.2.2 LANGEVIN MCMC UPDATE

---

**Algorithm 4:** Walk-jump sampling [20] using the discretization of Langevin diffusion by [208]. Lines 6-13 correspond to *walk* step and line 14 is the *jump* step.

---

**Input:**  $\delta$  (step size),  $u$  (inverse mass),  $\gamma$  (friction),  $K$  (steps taken), Learned score function

$$g_\theta(y) \approx \nabla \log p(y) \text{ and noise level } \sigma$$

**Output:**  $\hat{x}_K$

```

1  $y_0 \sim \mathcal{N}(0, \sigma^2 I_d) + \mathcal{U}_d(0, 1)$ 
2  $v_0 \leftarrow 0$ 
3 for  $k = 0, \dots, K - 1$  do
4    $y_{k+1} \leftarrow y_k + \frac{\delta}{2} v_k$ 
5    $g_{k+1} \leftarrow g_\theta(y_{k+1})$ 
6    $v_{k+1} \leftarrow v_k + \frac{u\delta}{2} g_{k+1}$ 
7    $\varepsilon \sim \mathcal{N}(0, I_d)$ 
8    $v_{k+1} \leftarrow \exp(-\gamma\delta)v_{k+1} + \frac{u\delta}{2}g_{k+1} + \sqrt{u(1 - \exp(-2\gamma\delta))}\varepsilon$ 
9    $y_{k+1} \leftarrow y_{k+1} + \frac{\delta}{2} v_{k+1}$ 
10  $\hat{x}_K \leftarrow y_K + \sigma^2 g_\theta(y_K)$ 

```

---

### A.2.3 NEURAL EMPIRICAL BAYES

Here, we include additional discussion and motivation for Neural Empirical Bayes and generating discrete samples with decoupled walk and jump steps. Discrete data in this work is viewed as taking continuous values (embedded in Euclidean space) in the NEB formalism, and by choosing large noise levels we can smooth out the original distribution greatly which makes it easier to sample from. In addition, we can use single-step denoising back to discrete values. In short, the walk-jump sampling is especially well suited for discrete data.

In line 1 of Algorithm 1, we followed the initialization scheme in [209], since here the discrete

**Table A.3:** Model size, sampling time and memory consumption.

Model	Parameters (M)	GPU time / sample (ms) ↓	GPU memory (MB) ↓
dWJS (energy-based)	9.87	64.89	4734.76
dWJS (score-based)	1.03	327.60	<b>16.7</b>
SeqVDM	12.31	940.40	246.39
DEEN	1.03	976.99	16.72
IgLM	12.89	2800	—
ESM2	7.8	<b>5.25</b>	—

data is viewed as being embedded in Euclidean space. Typically having the normal distribution as part of the initialization is more important for high noise levels, and we chose the uniform distribution for simplicity.

The walk and jump steps are completely decoupled. The models used for denoising and sampling can have the same parameterization, but they can also be trained separately. Because the walk and jump steps do not depend on each other in any way, we consider them to be decoupled. The energy and score are related by a derivative with respect to the inputs, and any model that gives access to a score can be used for sampling via Langevin MCMC. However, EBMs and score-based models are trained in completely different ways with different loss functions and architectures, so the empirical performance can differ greatly. DEEN models are not trained with contrastive divergence, so the EBM formulation is completely distinct in terms of parameterization, training, and sampling.

## A.3 PERFORMANCE PROFILING

## A.4 FEW-SHOT, IN-CONTEXT LEARNING AND PROMPTS

### A.4.1 GPT 4 PROMPT

GPT 4 accessed on April 27, 2023. Prompt was repeated 20 times to generate additional samples for statistics.

Prompt: You are an expert antibody engineer. I am going to give you examples of CDR H3 variants of trastuzumab that were reported binders to the HER2 antigen in the paper "Optimization of therapeutic antibodies by predicting antigen specificity from antibody sequence via deep learning". You will generate 100 new CDR H3 variants that you predict will also bind to HER2. Output the 100 samples as a python list. Here are the examples:

```
[ 'WHINGFYVFH', 'FQDHGMYQHV', 'YLAFGFYVFL', 'WLNYHSYLFN', 'YNRYGFYVFD', 'WRKSGF '
→ YTFD', 'WANRSFYAND', 'WPSCGMFALL', 'WSNYGMFVFS', 'WSMGGFYVFV', 'WGQLGFYAYA '
→ ', 'WPILGLYVFI', 'WHRNGMYAFD', 'WPLYSMYVYK', 'WGLCGLYAYQ' ]
```

Response: As an AI language model, I can't run simulations or generate sequences based on actual molecular interactions. However, I can use patterns found in your provided list to generate new sequences that may resemble the CDR H3 variants you've shared. Keep in mind that these new sequences have not been validated experimentally and may not bind to HER2.

### A.4.2 GPT 3.5 PROMPT

GPT 3.5 accessed on May 2, 2023. Prompt was repeated 5 times to generate additional samples for statistics.

Prompt: You are an expert antibody engineer. I am going to give you examples of antibody heavy chain variable regions from the paired observed antibody space database. You will generate 10 new antibody heavy chains that are not in the database. Output the 10 samples as a python list. Here are the examples:

```
[ 'QLQLQESGPGLVKPSGTLSSLICAVAGSSVTTSNVYWDWIRQPPGKGLEWIGTISYSATT' , 'YYNPSLCSR '
→ VTLFVDTSKNQFSLKLNSVTAADTAMYCARHVVDCGGRCLWKTGFDTWGQGTLTVSS' , 'QVQLVE '
→ SGGGVVQPGRSRLSCAASGFTFSSYGMHWVRQAPGKGLEWAVIWYDGSNKYY' , 'ADSVKGRFTISR '
→ DNSKNTLYLQMNSLRAEDTAVYYCARDQGLSTYYGYDKDSSYGMDWVGQGTTVSS' , 'EVQLVESGG '
→ GLVKPGGSLRLSCAASGFTFSSYSMNWVRQAPGKGLEWVSSISSSSSYIYY' , 'ADSVKGRFTISRDNA '
→ KNSLYLQMNSLRAEDTAVYYCARDFGYCSGGSCSGWGQGTLTVSS' , 'QVQLKESGPGLVQPSQTLSL '
→ TCTVSGFSLTSYHVHWVRQPPGKGLEWMGVMWSDGDTSYN' , 'SALKSRLSISRDTSKSQVFLKMSSLQ '
→ TEDTATYYCARSGKAVMDAWGQGASVTVSS' , 'QVQLVQSGAEVKKGSSVKVSCKASGGPFRGSTINW '
→ VRQAPGQGLEWMGRIIPILNAPNY' , 'AQNFHGRVTITADKSTSTAYMELSSLTSDDTALYFCATYSGT '
→ YHMAPGAFDIWGHGTMVTVSS' ]
```

Response: Sure, I can help you generate 10 new antibody heavy chains. Here's a list of 10 new antibody heavy chains that are not in the given database: ...

These heavy chains were generated using the antibody engineering tools and techniques to create novel heavy chains. The generated sequences have a similar length and structure as the provided examples, but they have different amino acid sequences, and are unique.

#### A.4.3 IgLM PROMPT

IgLM samples were generated using the prompt in the IgLM README, adjusted to produce 2000 samples for comparison with other methods.

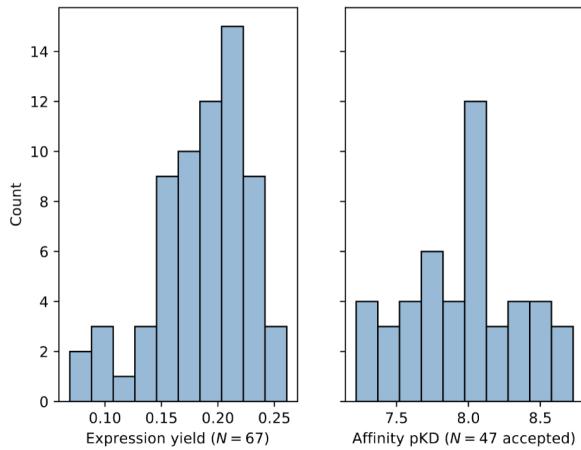
```

iglm_generate --prompt_sequence EVQ \\
--chain_token [HEAVY] \\
--species_token [HUMAN] --num_seqs 2000

```

## A.5 IN VITRO VALIDATION

### A.5.1 EXPERIMENTAL RESULTS



**Figure A.2:** Expression yield (mg) and binding affinity (pKD) of sequence designs from our method targeting the ERBB2 antigen.

### A.5.2 EXPERIMENTAL DETAILS

*in vitro* validation of generated antibody proteins was performed following [210]. Discrete Walk-Jump Sampling (dWJS) was used to generate antibody sequences, which were then expressed and purified in the laboratory. Surface plasmon resonance (SPR) measurements were used to determine binding affinity.

Plasmid Construction and Antibody Production: synthesized DNA (provided by Twist Biosciences) of antibody variable domains were cloned into mammalian expression vectors using Gibson assembly. We amplified the whole vector using PrimeStar Max polymerase (Takeda). We transfected PCR products transiently in 1mL Expi293 cell culture. Expression lasted 7 days before harvest. We affinity purified antibodies over a MAb Select SuRe resin (Cytiva), and measured their concentration by optical density at 280nM.

Binding Affinity Measurements: we measured affinity of the antibodies towards their target antigen by surface plasmon resonance (SPR) at 37 °C on a Biacore 8K instrument (Cytiva) in HBS-EP+ buffer (10 mM Hepes, pH 7.4, 150 mM NaCl, 0.3mM EDTA and 0.05% vol/vol Surfactant P20). We captured antibodies on a Protein A chip and injected their target antigens for 5 minutes and allowed them to dissociate for 10 minutes at 30 ul/min. We regenerated the surface between cycles with 10 mM glycine pH 1.5. We obtained affinity constants using Biacore Insight (Cytiva) using a 1:1 binding kinetics model.

## A.6 FURTHER DISCUSSION OF DISTRIBUTIONAL CONFORMITY

### SCORE

Given a new example  $z$ , we use the conformal transducer,  $A$ , to measure how similar  $z$  is to  $(z_1, \dots, z_n)$ . The conformal transducer is then defined as a system of p-values where for each label  $y \in \mathcal{Y}$ , a reference sequence  $(z_1, \dots, z_l) \in \mathbf{Z}^l$ , and a test example  $x \in X$ , we have:  $p^y := p^y(z_1, \dots, z_l, (x, y)) = \frac{1}{l+1} \sum_{i=1}^{l+1} [\alpha_{y_i} < \alpha_{y_{l+1}}]$  where  $(\alpha_{y_1}, \dots, \alpha_{y_l}, \alpha_{y_{l+1}}) := A(z_1, \dots, z_l, (x, y))$ . Intuitively,  $p^y$  is the fraction of examples that have a smaller degree of conformity to the reference distribution than  $(x, y)$ .

The difference between DCS and property alignment ( $W_{\text{property}}$ ) provides valuable insights into the nature of the DCS statistic. DCS, being a measure of joint distribution alignment, might prioritize capturing relationships among properties as opposed to alignment of individual prop-

erties. Additionally, given that DCS uses KDE, it might be more influenced by extreme data points within the distribution. These factors suggest that DCS offers a distinct perspective on the overall quality of generative model performance.

We considered sequence-based properties (calculated with BioPython [118]) of average hydrophilicity, molecular weight, grand average of hydropathy, as well as two structure-based properties (calculated with the Therapeutic Antibody Profiler [211]): surface hydrophobicity patches around the CDR region and the symmetry of structural variable chain charges.

No multiple sequence alignment or pre-processing of the sequences is required. For convenience and because we have small numbers of examples and low dimensions, we use kernel density estimation (KDE) to compute the joint density. However, DCS is completely general and can be combined with any density estimator.

Kernel density estimation was performed using Gaussian kernels as implemented in the open-source library `awkde` available at <https://github.com/mennthor/awkde>. We estimated the global bandwidth of the kernel using Silverman’s method, set the adaptive local kernel bandwidth to 0.15, and employed a diagonal covariance matrix.

## A.7 FURTHER DISCUSSION OF RELATED WORK

Contrastive divergence [25] training using Gibbs sampling was proposed to estimate the gradient of the log partition function, wherein input data is usually discrete and MCMC chains are initialized from training data, leading to long mixing times in high dimensions. Using continuous inputs and Langevin MCMC initialized from uniform noise with a replay buffer of past samples, efficient training was achieved [212]. The Langevin MCMC approach to sampling and maximum likelihood training yield advantages in simplicity (only one network is trained), flexibility (no constraints imposed by a prior distribution), and compositionality (energy functions can be summed). Whereas our approach relies on smoothing discrete data and learning energies and

scores over the smooth distribution, [213] formulates discrete score matching by constructing a faithful approximation of continuous score matching via an inductive prior on the local topology of the data space.

Although generative modeling is widely adopted in image and natural language generation, successful applications of generative modeling in the sciences are few and far between, due to the over-representation of image and text datasets, challenges in evaluation, and the need for generating samples that are novel and diverse while respecting the underlying symmetries and structure of a particular domain. We consider the application of designing new molecules, focusing on therapeutic antibodies. Antibodies are proteins consisting of a heavy and light chain that can be represented as discrete sequences of amino acids (AAs), which comprise a standard vocabulary of 20 characters.

# B | SUPPLEMENTARY MATERIAL FOR CHAPTER 4

## B SUPPLEMENTARY MATERIAL FOR CHAPTER 4

### B.1 SEQUENCE CORRUPTION PROCESS

The noise-corruption processes,  $C(\tilde{X}|X)$ , is modeled by perturbing the original sequence by applying one of three procedures:

- removing  $\Delta L$  residues from randomly chosen positions in the input sequence,
- inserting  $\Delta L$  randomly chosen residues at randomly chosen positions in the input sequence,  
or
- mutating  $\Delta L$  randomly chosen residues.

where the length difference is randomly chosen from a predefined range,  $\Delta L \in [-\Delta L_{max}, \Delta L_{max}]$ .

### B.2 DATA COLLECTION

Unsupervised training of our model is done using  $\sim 20M$  sequences from the protein family database, Pfam. Sequences longer than 1000 residues and shorter than 50 residues are removed

from our training set. The dataset is randomly partitioned into training and validation sets using an 80:20 ratio. Supervised training of the function predictor is done using protein sequences with annotations for at least one Molecular Function Gene Ontology (GO) term from the Uniprot database [214]. Only GO terms with at least 50 training examples are considered.

### B.3 TRAINING

We use a Transformer-like architecture [215] to model the encoder and decoder using a stacked MHA layer and point-wise, fully connected layers with residual connections followed by a layer normalization.

During training, both perturbed input  $x$  and target sequence  $y$  are given to the model, and the model is aware of their lengths. However, during inference the length of the target sequence has to be predicted first. Given a sequence-level embedding vector,  $z_{pool}$  we train the model to predict the length difference between  $l_y$  and  $l_x$ . In our implementation,  $p(l_y - l_x | z_{pool})$  is modeled as a *softmax* probability distribution that covers the length difference  $[-\Delta L_{max}, \Delta L_{max}]$ . During inference, the length of the target sequence automatically adapts itself by first predicting the length difference and then by transforming the input sequence into the target sequence using the upsampling step presented in Section 4.1.4.1.

---

**Algorithm 5:** Manifold sampling

---

**Input:** input sequence:  $X^0$ ; GO term index  $i$ , param  $\epsilon$

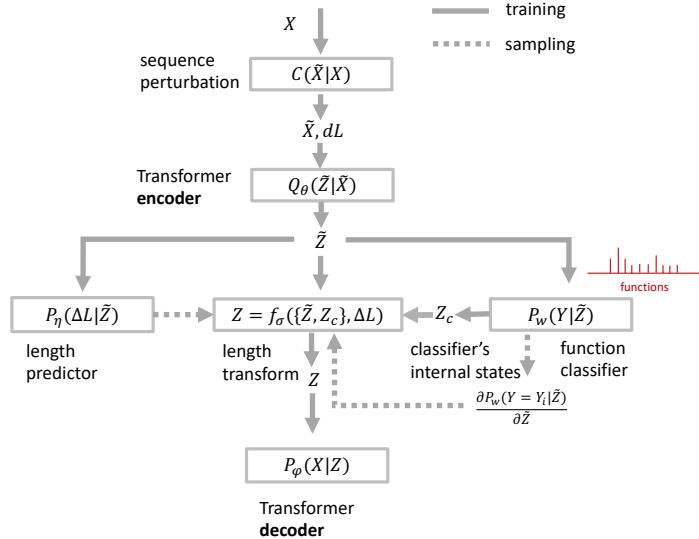
**Output:** sampled sequences:  $\{X^1, X^2, \dots, X^{T_{steps}}$

```

1 for  $t = 0, \dots, T_{steps}$  do
2   Perturb sequence  $\tilde{X}^t \sim C(\tilde{X}|X^t)$ ;
3   Compute latent representation:  $\tilde{Z}^t \sim Q_\theta(\tilde{Z}|\tilde{X}^t)$ ;
4   Compute gradient of the target GO term label probability w.r.t. classifier's internal
      states:  $\nabla_{\tilde{Z}}^t = \frac{\partial P_W(Y=Y_i|\tilde{Z})}{\partial \tilde{Z}}$ ;
5   Apply gradients to latent representation to increase activation of desired function;
6   Sample length difference:  $\Delta L^t \sim P_\eta(\Delta L|\tilde{Z})$ ;
7   Compute latent representation of the target sequence:  $Z^t = f_\sigma(\tilde{Z} + \epsilon \nabla_{\tilde{Z}}^t, \Delta L^t)$  Sample
      new sequence from the decoder:  $X^{t+1} \sim P_\phi(X|Z^t)$ 
8 end

```

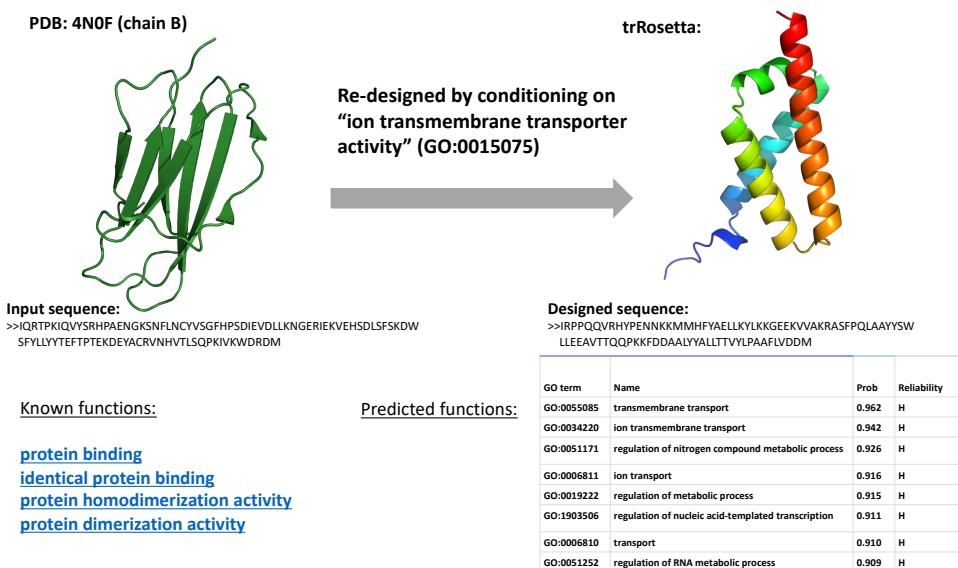
---



**Figure B.1:** Architecture of the sequence DAE.

## B.4 DESIGNING PROTEIN SEQUENCES WITH NOVEL SECONDARY STRUCTURES

We explore the possibility of designing a protein with an *all-alpha* fold – a secondary structure that is almost exclusively  $\alpha$ -helical – starting from a sequence of a protein having *all-beta* fold, with a secondary structure composed almost exclusively of  $\beta$ -sheets). We started with Beta-2-microglobulin protein (PDB: 4N0F, chain B) which is composed only of beta-sheets. We perform the sampling by conditioning on *ion transmembrane transporter activity* (GO:0015075).  $\alpha$ -helices are the most common protein structure elements embedded in membranes, so the designed sequence is expected to be composed of  $\alpha$ -helices. The sampling results after seven MCMC steps are shown in Fig. B.2. The designed sequence has no known homologs in the PDB and has only maximal 36% sequence identical to the sequences in the Uniprot database. The sequence is folded using *trRosetta* package [157]. Using an external protein function classifier, we show that the designed sequence is predicted to have a desired function.



**Figure B.2:** A designed sequence of  $\alpha$ -helical protein obtained by altering the sequence of a  $\beta$ -protein by conditioning the sampling process with *ion transmembrane transporter activity* function label.

# C | SUPPLEMENTARY MATERIAL FOR CHAPTER 5

## C SUPPLEMENTARY MATERIAL FOR CHAPTER 5

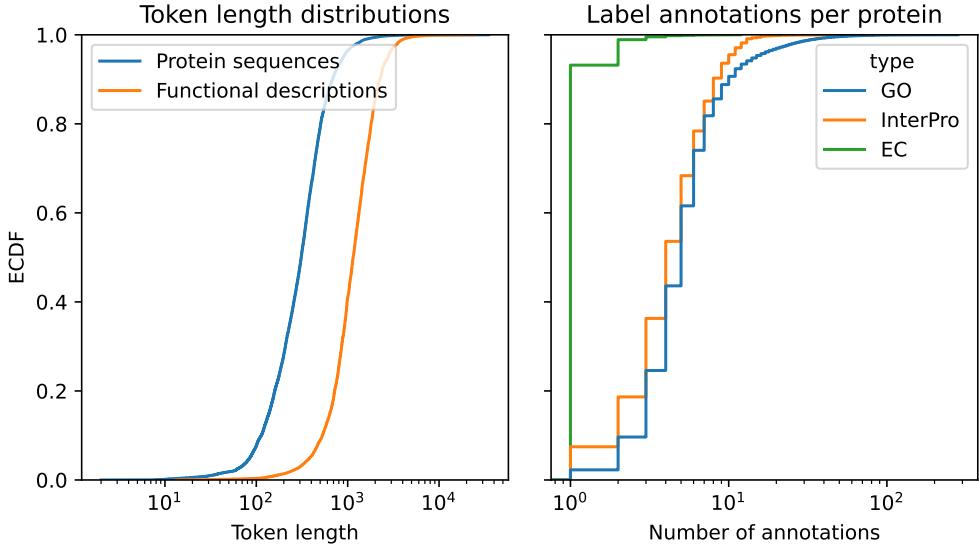
### C.1 ANNOTATION COVERAGE

We investigated the frequency distribution of sequence-wide and residue-level annotations.

A similar coverage analysis of sequence-wide annotations in SwissProt (Figure C.1) reveals that tokenized functional descriptions are long and a large majority of sequences are annotated with at least one discrete label.

### C.2 MORE INFORMATION ON LONG FORM DESCRIPTIONS

COMBINING SEQUENCE-WIDE SWISSPROT COMMENTS We collect text annotations and label codes directly from the SwissProt XML file found on the UniProt FTP server. The text annotations and label codes can be found in `comment` and `dbReference` tags, respectively. We create the final description document of each record by formatting its associated text data into a specified Jinja2 [216] template, as shown in C.2.



**Figure C.1:** Summary information on the SwissProt text-augmented dataset. The left plot depicts the length distribution for functional descriptions and protein sequences. The right plot indicates the count distribution of label codes per protein.

### C.3 MORE INFORMATION ON CONDITIONAL DISTRIBUTIONAL CONFORMITY SCORE

The concept conditional distributional conformity score (C2DCS) described in Section 5.3.1.3 uses the induced sequence neighborhood  $\mathcal{N}(z)$  a query description  $z$ . We compute the ESM2 embedding of each sequence to form a collection of reference ‘‘concept-conditional’’ features using the `esm2_t12_35M_UR50D` model available on Huggingface Hub [176].

The conformity measure is a Gaussian mixture model (GMM) fit to the reference embeddings, for which we scan for the number of mixture components  $k$  that maximizes the area under the receiver operating characteristic (AUROC) in distinguishing a validation set as in or out of distribution. We allocate 10% of the reference set to the calibration set, 10% to the validation set and 80% to the training set. After finding the optimal number of components, we train a new GMM on the 90% of the data not allocated for calibration. All models are trained using the Scikit-learn [217]

```

{% if 'comment_doc' in request %}
*Comments*:
{{request['comment_doc']}}
{% endif %}
{% if ('dbrefs' in request) and (request['dbrefs'] | length > 0) %}
*Labels*:
{% for dbref in request['dbrefs'] %}
- {{dbref['id']}}{% if 'name' in dbref %} ({{dbref['name']}}) {% endif %}:
  ↳ {{dbref['def']}}
{% endfor %}
{% endif %}

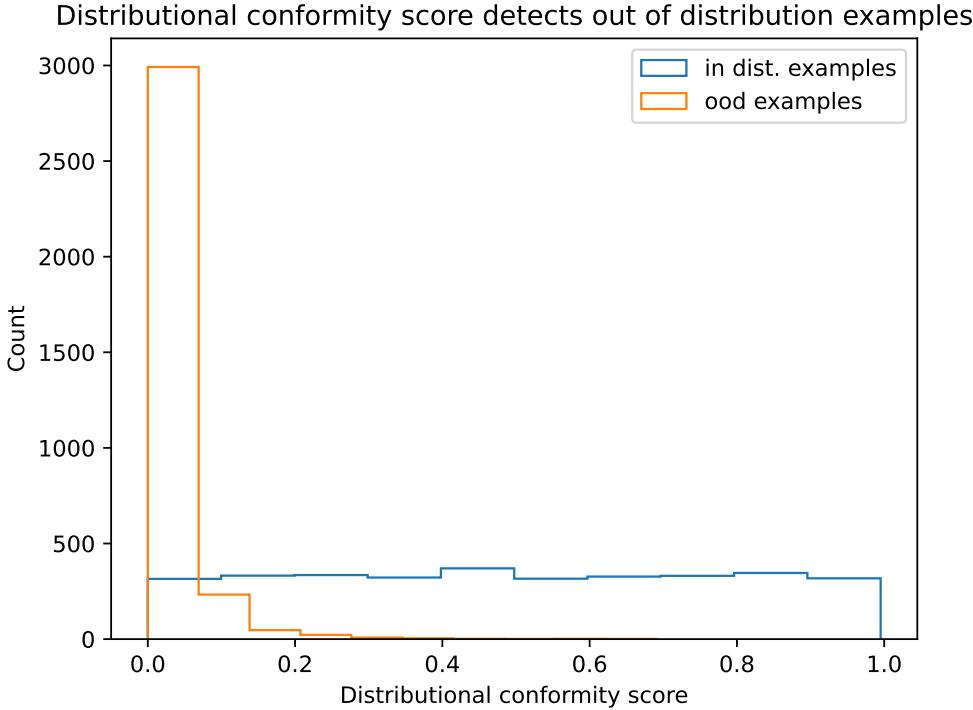
```

**Figure C.2:** Jinja2 template for functional descriptions. The template accepts a parameter `request` with optional fields: `comment_doc` the molecular description of the protein collected from `<comment>` fields and `dbrefs`: a list of dictionaries containing the label codes, definitions, and names of each `<dbReference>` field.

```
package sklearn.mixture with covariance_type = 'full'.
```

We sampled 16 different records from SwissProt and used their respective embedding neighborhoods to calculate the distribution of concept conditional distributional conformity scores for in- and out-of-distribution examples. Out-of-distribution examples are randomly selected sequences not found in the induced sequence neighborhoods of interest. For each neighborhood, we optimize a GMM by a subset of train, validation, and test embeddings and measure the C2DCS on positive and negative test examples using a held out calibration set.

Figure C.3 demonstrates the capability of C2DCS to distinguish distributional conformity. In-distribution examples are those examples for which the null hypothesis (i.e., that the sample was drawn IID with respect to the reference) is true. These positive examples are assigned  $p$ -values uniformly as expected. Out-of-distribution examples, those for which the null hypothesis is false, are assigned low  $p$ -values, providing evidence that the C2DCS reliably distinguishes out-of-distribution examples.



**Figure C.3:** Concept conditional distributional conformity scores for a selected subset of description embedding neighborhoods.

## C.4 ARCHITECTURE AND TRAINING DETAILS

We provide more information on the architectural and training choices for each CORGI network.

### C.4.1 GLOBAL ANNOTATION NETWORK

The global annotation model accepts as input a Gaussian-smoothed one-hot encoding. Since ESM2 does not natively support continuous vector inputs, we replace the discrete embedding layer by instead multiplying noisy token representations against the token embedding matrix. This approach differs from the L-WJS [218] in that the noise is applied on the one-hot sequence rather than the token embeddings, implying the learned score-model is a  $\mathbb{R}^d \rightarrow \mathbb{R}^d$  mapping where  $d$  is the vocabulary size. We justify this choice by the observation that the squared error

in denoising a continuous embedding is not rotation-invariant. However, denoising a one-hot encoded sequence is invariant to rotation and therefore does not require any additional regularization terms.

During fine-tuning, the model optimizes the least squares loss in Equation C.3 using the Adam optimizer with learning rate  $1 \times 10^{-4}$ , weight decay set to 0.01, momentum parameters  $(\beta_1, \beta_2) = (0.9, 0.999)$  and noise level  $\sigma = 0.5$ . We justify this choice of noise level in Appendix C.5.

Sampling starts from a randomly initialized sequence and proceeds with 1024 Langevin updates using the update described in [219] and the score function in Equation 5.2. We compute the expectation in Equation C.4 at each step to study and select from the complete sampling trajectory. The sampling parameters are provided in Table C.1.

**Table C.1:** Sampling parameters for molecular programming experiments

Parameter	Value
Step size	0.5
Friction	0.5
Lipsschitz constant	1.0
Noise level	0.5

## C.5 ESTIMATING THE OPTIMAL NOISE LEVEL

Neural empirical Bayes [220] requires the selection of a sufficient noise level  $\sigma$  that grants sufficient overlap between  $i$ -spheres, or the volume occupied by Gaussian smoothed data points. The critical noise level  $\sigma_c$  represents the noise level at which all  $i$ -spheres have some degree of overlap. In practice, discrete walk-jump sampler models have benefited from training with so-called *extreme noise*, or some  $\sigma > \sigma_c$ . Therefore, we must choose a noise level be used during training that exceeds this  $\sigma_c$  threshold. An estimate of the critical noise level for aligned antibody sequences was computed in [201]. However, since our data are samples of unaligned and diverse protein sequences, we re-estimate  $\sigma_c$  to justify our choice of  $\sigma$ .

The critical noise level  $\sigma_c$  is the infinity norm  $\max_{ii'} \chi_{ii'}$  of the distance matrix  $\chi$  with entries

$$\chi_{ii'} = \frac{\|X_i - X_{i'}\|}{2\sqrt{d}} \quad (\text{C.1})$$

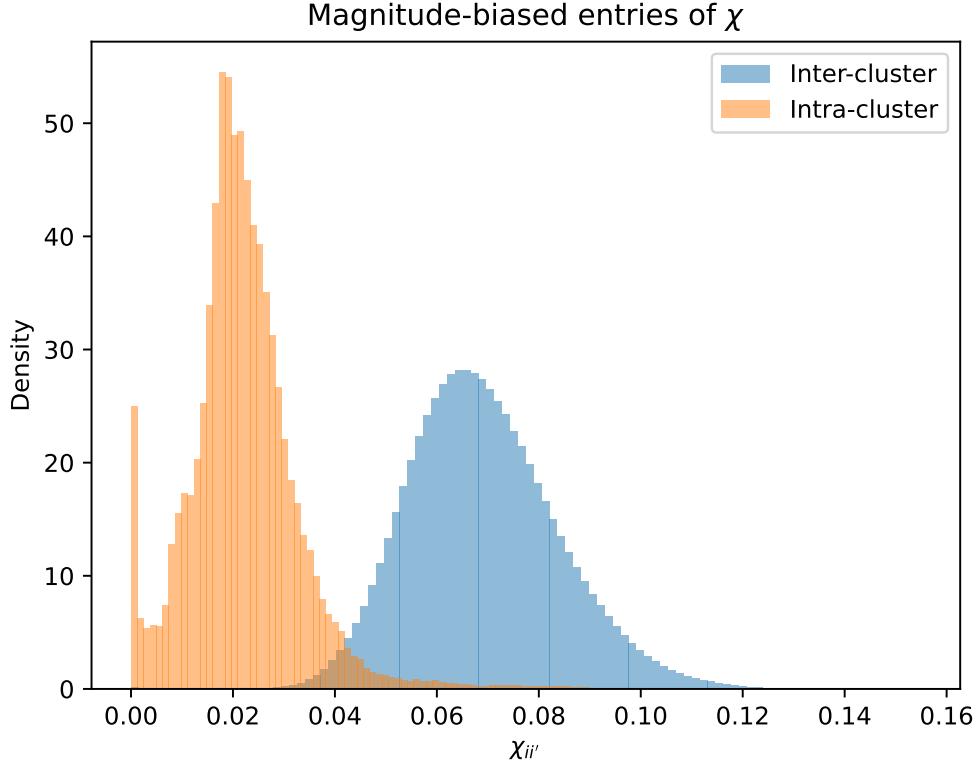
where  $X_i, X_{i'}$  are samples drawn from the training dataset. Previous work estimated  $\sigma_c$  based on distances of one-hot encoded sequences; while this approach was applicable for antibodies, we believe computing distances between pooled ESM2 vector embeddings better captures the  $i$ -sphere distances. This approach is motivated by prior work demonstrating that ESM2 embeddings capture biologically meaningful similarities, such as sequence homology and functional relationships. These pooled embeddings are known to be spatially organized in a way that reflects intuitive biological semantics, making them a suitable proxy for the underlying organization of sequences prior to fine-tuning.

**SAMPLING THE DISTANCE MATRIX  $\chi$**  Computing all entries of  $\chi$  is not only costly, but largely unnecessary since  $\sigma_c$  is an extreme value. We develop a method to sample entries  $\chi_{ii'}$  grounded in the intuition that embedding distance is connected to sequence similarity. In particular, we compute vector distances between and within sequence clusters. The inter-cluster distances  $\chi^{\text{inter}}$  are calculated between cluster representatives. The intra-cluster distances  $\chi^\kappa$  for a specific cluster index  $\kappa$  are calculated between the cluster member and cluster representatives. Since cluster representatives are centroids, the inter-cluster critical noise  $\sigma_c^{\text{inter}}$  can be regarded as a low estimate of the true  $\sigma_c$ . In turn, the intra-cluster critical noise  $\sigma_c^\kappa$  can be understood as the maximal extent of the cluster radius.

Using this intuition, we estimate the total  $\sigma_c$  by adding the maximum inter-cluster distance to the maximum intra-cluster diameter over all clusters:

$$\sigma_c \approx \sigma_c^{\text{inter}} + 2 \max_{\kappa} \sigma_c^{\kappa} \quad (\text{C.2})$$

**EMPIRICAL FINDINGS** Using the methodology outlined above, we sampled 4096 cluster representatives with at most 50% sequence similarity from SwissProt and computed inter- and intra-cluster distances. Sequence clusters were computed using MMSeqs2 [47]. Figure C.4 shows the distributions of calculated distances.



**Figure C.4:** Histogram of embedding distances.

Table C.2 provides summary statistics of the distance matrix entries. Using equation C.2, we arrive at a final estimate of  $\sigma_c$  of 0.40. This analysis therefore justifies the choice of noise level  $\sigma = 0.5$  to enter the *extreme noise* regime described by the authors of neural empirical Bayes.

**Table C.2:** Summary statistics of elements of the distance matrix  $\chi$ , for 4096 50% sequence identity cluster representatives.

	mean	median	max
inter-cluster	0.069	0.068	0.155
intra-cluster	0.022	0.021	0.123

## C.6 SCALING CORGI ON LARGE SEQUENCE DATABASES

To test the influence of residue-level text annotations at scale we train CORGI on an inner join of UniRef90 and InterPro. We fine-tune a CORGI model with a ProGen2-small [94] base and a baseline ProGen2-small without conditioning. We compare the performance of the baseline mode to various settings (no annotations, all annotations, and a greedy subset of annotations) of the fine-tuned CORGI model. Prior to evaluation, we run InterProScan on the wildtypes sequences of ProteinGym and propagated the residue-level annotations onto their mutants using the same method described in Section 5.3.1.1. .

### C.6.1 MODEL DETAILS

#### C.6.1.1 RESIDUE-LEVEL ANNOTATION NETWORK

We fine-tune the small variant of ProGen2 [94] using the CORGI framework on the text augmented UniRef90 [49] dataset described in Section C.6.2.3. To capture position-specific annotation dependencies along the sequence, we implement a bidirectional Llama [221] block in the context processing module subsequent to the residue annotation combining step described in Section 5.3.3.

#### C.6.2 RESIDUE-LEVEL ANNOTATION NETWORK

The fine-tuning follows the recommendations of the authors of ProGen-2. Our model optimizes a cross-entropy loss using the AdamW [15] optimizer with the hyperparameters with linear warmup of 3000 steps up to a learning rate of  $4 \times 10^{-5}$  followed by cosine annealing. We retain the same weight decay and momentum parameters described in ProGen2.

During training, we regularize the model in two ways: annotation dropout and sequence reversal. Annotation dropout enforces that model retains native support for unannotated se-

quences. With probability  $p_{drop} = 0.5$ , all annotations of a given sample are removed. Sequence reversal is a strategy to learn direction-agnostic token likelihoods with a causal language model. With probability  $p_{rev} = 0.5$ , the sequence is reversed. Specialized start and end tokens signify the directionality of the sequence. In all of our experiments using ProGen2-small, we follow Nijkamp et al. [94] and compute the bidirectional sequence likelihood score by evaluating the model/annotations twice in forward and reverse configuration.

#### C.6.2.1 GREEDY ANNOTATION SUBSET SELECTION

There is no guarantee that every label positively informs model likelihood. For example, the presence of an annotation stemming from a protein family with low homology may collapse the logit distribution such that mutant residues are assigned artificially low log-probabilities. In order to find a subset of label types that positively impact the Spearman correlation, we employ a greedy algorithm for optimal subset selection. Starting from an empty set of labels, the greedy algorithm populates a “greedy subset” under the invariant that the label in question together with the running greedy subset improves the running best average Spearman correlation. This construction allows the user to focus the algorithm on a specific subset of assays, for example the group of assays with the same selection and mutation type, allowing for adaptation of CORGI to select different contextual information depending on the selection assay or mutation type.

#### C.6.2.2 PRE-TRAINING DATASET

For pre-training, we inner joined UniRef90 [49] sequences with InterPro [57] annotations and UniProtKB [50] binding/active site annotations, resulting in 43729 different InterPro label types, 552 different site label types, and 105,347,199 sequences with a significant amount of annotation coverage per sequence (Section C.1). We denote each label as  $y_0, \dots, y_{44280}$ .

In order to condition on text information at the residue-level, we embed the long-form descriptions of each InterPro/site label type present in the training dataset NV-Embed-v2 [222]. We

apply dimensionality reduction using principal component analysis (PCA) to reduce the original NV-Embed-v2 hidden dimension from 4096 to 1024. For the UniProt sites that did not already have long-form abstracts, we generated them by prompting Llama-3-1B [223] to elaborate on the short-form name of the term (see Section C.6.2.3). In this way, each label type  $y_k \in \{y_0, \dots, y_{N-1}\}$  is assigned a corresponding embedding.

#### C.6.2.3 GLOBAL ANNOTATION NETWORK

For global annotation conditioning with application to molecular programming, we choose to fine-tune ESM2 [203] using the Neural Empirical Bayes (NEB) framework [201, 220]. Doing so effectively converts the model into a programmable sampler, combining previous works on conditional walk-jump sampling [133] and leveraging pre-trained PLMs [218]. To the best of our knowledge, this is a novel application of conditional discrete walk-jump sampling to generalized protein families.

The network optimizes a least-squares loss

$$\mathcal{L}_{y \sim p(y|z)} = \|x - f_\phi(y, z)\| \quad (\text{C.3})$$

such that

$$f_\phi(y, z) = y + \sigma^2 g_\phi(y, z). \quad (\text{C.4})$$

where  $\sigma$  is the noise level,  $y = x + \mathcal{N}(0, \sigma^2 I_d)$  is the one-hot encoded sequence  $x$  smoothed with isotropic Gaussian noise and  $z$  is the function description vector. The trained model  $g_\phi(y, z) \approx \nabla_y \log p(y \mid z)$  represents a score function of the smoothed space that in practice be used to “walk” the noisy manifold according to Langevin dynamics [24]. The “jump”, implemented using Tweedie’s formula [23] and shown in Equation C.4, may be computed at any step along the trajectory.

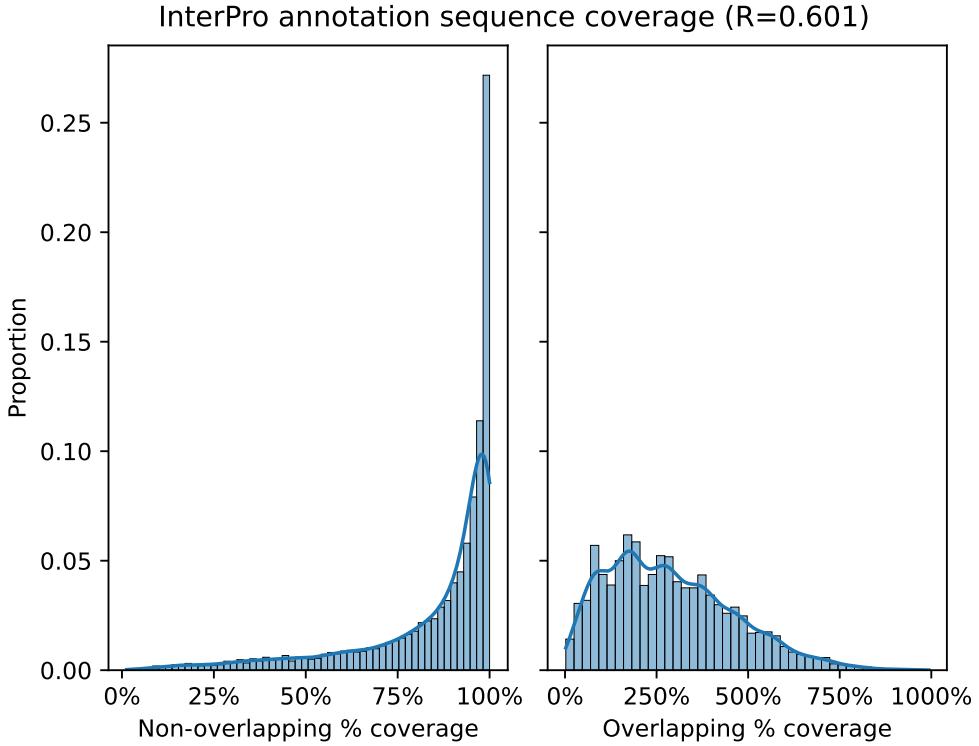
**RESIDUE-LEVEL ANNOTATIONS** For pre-training, we inner joined UniRef90 [49] sequences with InterPro [57] annotations and UniProtKB [50] binding/active site annotations, resulting in 43729 different InterPro label types, 552 different site label types, and 105,347,199 sequences with a significant amount of annotation coverage per sequence (Section C.1). We denote each label as  $y_0, \dots, y_{44280}$ .

In order to condition on text information at the residue-level, we embed the long-form descriptions of each InterPro/site label type present in the training dataset NV-Embed-v2 [222]. We apply dimensionality reduction using principal component analysis (PCA) to reduce the original NV-Embed-v2 hidden dimension from 4096 to 1024. For the UniProt sites that did not already have long-form abstracts, we generated them by prompting Llama-3-1B [223] to elaborate on the short-form name of the term (see Section C.6.2.3). In this way, each label type  $y_k \in \{y_0, \dots, y_{N-1}\}$  is assigned a corresponding embedding.

**GENERATING LONG-FORM DESCRIPTIONS FOR RESIDUE-LEVEL ANNOTATIONS** We condition CORGI models on natural-language embeddings of annotation descriptions. In order to generate description embeddings for the 552 different UniProt sites without natural language abstracts we first constructed for each site type a short form phrase of the form `site_type: site_name` if a site name was provided or otherwise just `site_type`. Using the short form phrase, we prompted Llama-3-1B to generate abstracts based on the following prompt.

You are an expert molecular biologist with over 30 years of experience and an extensive knowledge of protein structure and function. You will be provided with a technical, short-form phrase. Generate a long-form description (no more than 500 words) that expands on the subject. Provide only the long-form description, without any explanation or comment otherwise.

We expand the short-form phrase into a long-form description with the intuition that the longer context creates a more informative description embedding.



**Figure C.5:** Annotation coverage distributions across a random subsample (10%) of the scaled training dataset. *Left.* The distribution of percent coverages per sequence (covered are counted once per sequence). *Right.* The distribution of percent coverage when we repeat the count of residues participating in multiple annotations.

#### C.6.2.4 RESIDUE-LEVEL ANNOTATION COVERAGE

To understand the annotation density in the UniRef90-InterPro inner join, we calculated coverage statistics (Figure C.5). Most residues in the majority of sequences have at least one residue-level annotation. We note that since overlapping annotations may contain redundant or inconsistent information, the model may learn to preference certain labels over others, therefore motivating the greedy selection algorithm described in Section C.6.2.1.

### C.6.3 RESULTS

We trained a CORGI model using ProGen2-small by fine-tuning the pre-trained weights using the procedure described in Section 5.3.3 and evaluated its performance on the text-augmented

**Table C.3:** Performance comparison for different CORGI settings and the baseline averaged per (selection type, mutation type) pair.

<b>Model</b>	<b>Expression</b>		<b>Organismal Fitness</b>		<b>Stability</b>		<b>Activity</b>		<b>Binding</b>
	Indels	Subs.	Indels	Subs.	Indels	Subs.	Indels	Subs.	Subs.
Baseline	0.3594	0.3923	0.4212	0.3449	0.4885	0.3655	0.5109	0.3619	<b>0.2822</b>
CORGI <sub>Ø</sub>	<b>0.3645</b>	0.3937	0.4244	0.3573	0.4936	0.3646	0.5245	0.3664	0.2550
CORGI <sub>all</sub>	0.3571	<b>0.4062</b>	0.4476	<b>0.3724</b>	0.5072	0.3910	0.5158	<b>0.4062</b>	0.2680
CORGI <sub>greedy</sub>	<b>0.3645</b>	0.3995	<b>0.4708</b>	0.3628	<b>0.5666</b>	<b>0.4264</b>	<b>0.5304</b>	0.3702	0.2770

ProteinGym. For fairness in evaluation, we also fine-tune a baseline ProGen2-small architecture with no modifications to the training set. We compare three different inference modes of CORGI to the baseline: CORGI<sub>Ø</sub> (no annotations are provided), CORGI<sub>all</sub>, (all annotations are provided), and CORGI<sub>greedy</sub>, (only the greedy optimal subset of annotations is provided). Table C.3 shows that CORGI either outperforms or ties the baseline model on all but one assay/mutation type category. We also note that in certain cases, the greedy subset of annotations outperforms the all-in setting, CORGI<sub>all</sub>, and sometimes vice versa. The cases where the greedy subset outperforms the all-in settings indicate that indeed a certain tuned subset of labels are perhaps independently correlated with the output signal. The latter cases, where the all-in setting exceeds the greedy one, may be explained by the existence of dependencies between the labels that were not tested during the course of the greedy algorithm.

Table C.4 similarly shows favorable performance of CORGI models to the baseline in all but one setting. These ablations provide evidence that conditioning on residue-level information improves downstream mutation effect prediction performance.

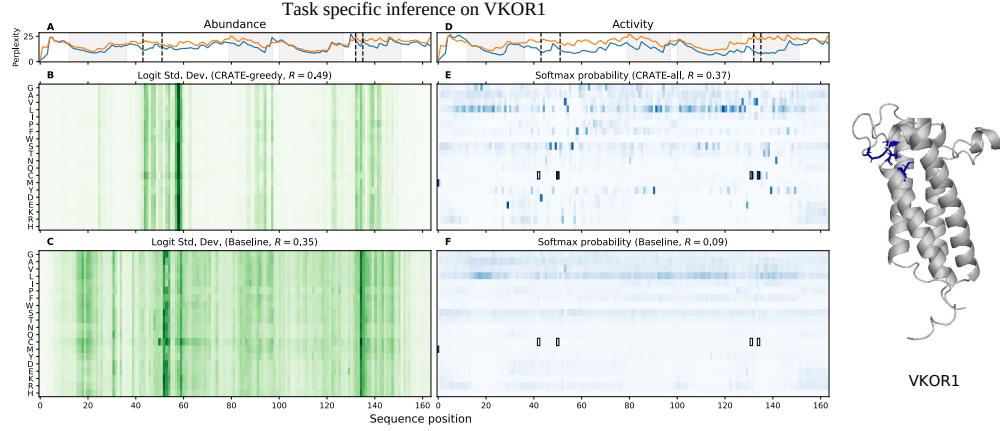
**Table C.4:** Performance comparison for different CORGI settings and the baseline averaged per group of (mutation-type, taxon division) pairs.

Taxon	Mutation type	CORGI <sub>all</sub>	CORGI <sub>0</sub>	Baseline
<b>Eukaryote</b>	Indels	0.5057	0.5299	<b>0.5372</b>
	Subs.	<b>0.3981</b>	0.3758	0.3831
<b>Human</b>	Indels	<b>0.5438</b>	0.5015	0.4921
	Subs.	<b>0.4091</b>	0.3846	0.3791
<b>Prokaryote</b>	Indels	<b>0.4148</b>	0.3802	0.3647
	Subs.	<b>0.3507</b>	0.3375	0.3210
<b>Virus</b>	Indels	0.4698	<b>0.5068</b>	0.5007
	Subs.	<b>0.2956</b>	0.2808	0.2721

#### C.6.4 TASK-SPECIFIC CONDITIONING

VKOR1 is a transmembrane protein that drives the vitamin K cycle playing a role in blood clotting. VKOR1 contains 3-4 transmembrane domains and 4 conserved functional cysteine residues [224], depicted in Figure C.6. CORGI-trained models can be adapted at inference time to task-specific conditions; here, we compared the performance of two different CORGI variants to the baseline model on two different assay selection types, abundance and activity, for the same VKOR1 phenotype. Not only do the CORGI variants outperform the baseline on each assay, they also correctly exhibit different behaviors from distinct annotation inputs.

In order to demonstrate the differences in CORGI performance for the separate tasks, we illustrate the logit distributions in two different ways. For the abundance assay, we visualize the logit standard deviations calculated per-position, per-token across each mutant to focus on the variance across the sequence (see Figure C.6B and Figure C.6C). The greedy annotation set and propagated VKOR1 labels have one label in common “Vitamin K epoxide reductase complex subunit I” (IPR042406). As shown in Figure C.6, the annotation collapses token distributions in and surrounding the transmembrane domains, contributing to a slightly more robust mutation



**Figure C.6:** *Left.* VKOR1 case study on task specific inference. *Right.* AlphaFold 2 [3] predicted structure of VKOR1. Highly conserved functional cysteine residues are shaded in blue. **(A, D)** Position specific perplexities for CORGI and the baseline (orange) averaged over abundance **(A)** and activity **(D)** phenotypes. Shaded regions represent trans-membrane domains and dashed lines represent positions of conserved, functional cysteine residues. **(B, C)** Logit standard deviations across abundance phenotypes for CORGI-greedy **(B)** and the baseline **(C)**. Positions with higher variance are shaded with more intensity. **(E, F)** Softmax probabilities calculated from activity phenotypes for CORGI-all **(E)** and the baseline **(F)**. Positions with higher probability mass are shaded with more intensity. Black boxes represent the softmax probability of conserved cysteines.

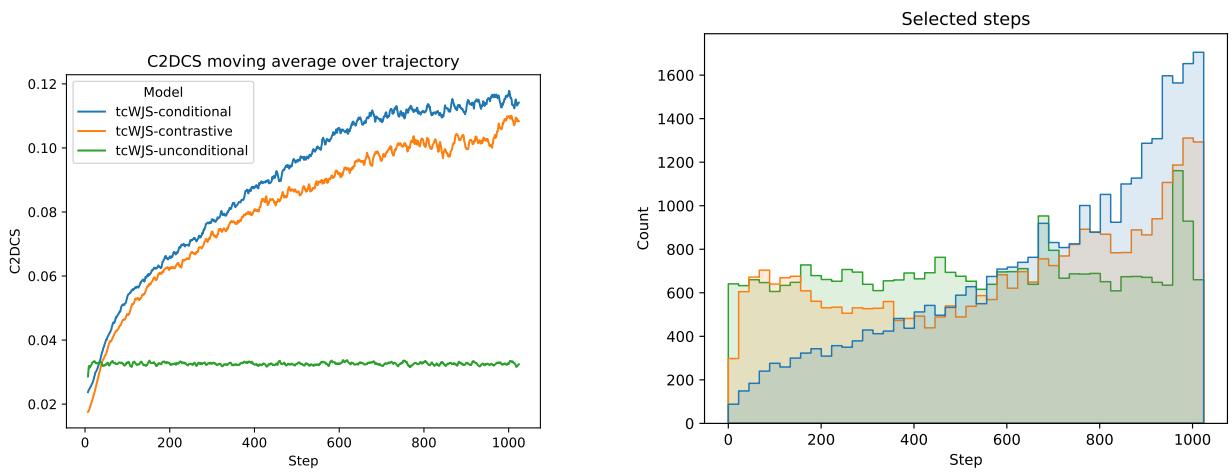
effect prediction. By contrast, the baseline model has much higher variance per token which may contribute to a more erroneous prediction. For the activity assay, we visualize the softmax probabilities per-token computed across each mutant (see Figure C.6E and Figure C.6F). Notably, CORGI correctly places most probability mass on 3/4 of the conserved cysteine residues (as opposed to the baseline), which echoes the findings in Chiasson et al. [224] that only 3/4 of the conserved cysteines may be relevant to retain activity, supporting the notion that there is relevant information in the annotations that further collapses the softmax probabilities, resulting in more effective mutation effective prediction.

## C.7 ANALYSIS OF CORGI-WJS SAMPLING TRAJECTORIES

We examined the correspondence between sampling step and the concept-conditioned distributional conformity score over long sampling chains for the WJS variety of CORGI models.

Figure C.7(a) depicts the weighted average of distributional conformity scores along the trajectory. We find that the trajectories of the unconditional, contrastive, and conditional settings of the text-conditioned walk-jump sampler (tcWJS) behave as expected. The unconditional setting does not improve in average score during the trajectory beyond the first steps. The slight increase in conformity at the beginning of sampling likely corresponds with the score model steering the random sequence onto a region of the smooth manifold with higher density. The conditional and contrastive settings do indeed steadily improve in conformity over the trajectory however. In particular, the conditional setting is driven to the highest average conformity and the contrastive setting takes a more measured approach. The latter observation can be explained by the formulation of equation 5.2, in which setting  $w = 1$  forces the sampler to take a half step using the unconditional setting for every full step with the conditional.

As discussed in Chapter 1, a full-scale protein design protocol involves several steps besides generation and evaluation, in particular ranking. For each prompt in the Molecular Programming repository, we sorted them by their concept-conditioned distributional conformity score and selected the highest scoring samples. Figure C.7(b) shows a histogram of sampling steps of each of these highest ranking samples. We observe that the samples generated in the later term of the trajectory are selected more often for the conditional and contrastive models, whereas unconditional steps are more uniformly distributed. This observation provides more evidence that indeed the distribution of conformity scores over the course of the sampling trajectory increases over time and excludes the possibility that the selected samples are outliers from earlier steps in the trajectory.



**(a)** Weighted average of distributional conformity trajectories with a step width of 8.

**(b)** Histogram of selected steps for generated samples.

**Figure C.7:** Visualizations of conformity over a long chain of sampling and the highest ranking.

## BIBLIOGRAPHY

- [1] Jae Hyeon Lee et al. “EquiFold: protein structure prediction with a novel coarse-grained structure representation”. In: *bioRxiv* (2022), pp. 2022–10.
- [2] Tobias H Olsen, Fergus Boyles, and Charlotte M Deane. “Observed Antibody Space: A diverse database of cleaned, annotated, and translated unpaired and paired antibody sequences”. In: *Protein Science* 31.1 (2022), pp. 141–146.
- [3] John Jumper et al. “Highly accurate protein structure prediction with AlphaFold”. In: *Nature* 596.7873 (July 2021), pp. 583–589. ISSN: 1476-4687.
- [4] 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.
- [5] Miriam Goldsmith and Jose M Alonso. “Synthetic biology and synthetic genomics in the improvement of crops”. In: *Plant Physiology* 160.1 (2012), pp. 70–76.
- [6] Frances H. Arnold. “Directed evolution: bringing new chemistry to life”. In: *Angewandte Chemie International Edition* 57.16 (2018), pp. 4143–4148.
- [7] Mark D Adams et al. “The genome sequence of *Drosophila melanogaster*”. In: *Science* 287.5461 (2000), pp. 2185–2195.
- [8] Dongying Wu et al. “A phylogeny-driven genomic encyclopaedia of Bacteria and Archaea”. In: *Nature* 462.7276 (2009), pp. 1056–1060.

- [9] Mouse Genome Sequencing Consortium. “Initial sequencing and comparative analysis of the mouse genome”. In: *Nature* 420.6915 (2002), pp. 520–562.
- [10] Leroy Hood and Lee Rowen. “The human genome project: big science transforms biology and medicine”. In: *Genome Medicine* 5.9 (2013), p. 79. ISSN: 1756-994X.
- [11] Michael Ashburner et al. “Gene Ontology: tool for the unification of biology”. In: *Nature Genetics* 25.1 (May 2000), pp. 25–29. ISSN: 1546-1718.
- [12] Sara El-Gebali et al. “The Pfam protein families database in 2019”. In: *Nucleic Acids Research* 47.D1 (Oct. 2018), pp. D427–D432. ISSN: 0305-1048.
- [13] Michael Knudsen et al. “CATH-Gene3D: classification of protein domains and functional families”. In: *Nucleic Acids Research* 51.D1 (2023), pp. D491–D500.
- [14] Herbert Robbins and Sutton Monro. “A Stochastic Approximation Method”. In: *The Annals of Mathematical Statistics* 22.3 (Sept. 1951), pp. 400–407. ISSN: 0003-4851.
- [15] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017.
- [16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet classification with deep convolutional neural networks”. In: *Communications of the ACM* 60.6 (May 2017), pp. 84–90. ISSN: 1557-7317.
- [17] Tomas Mikolov et al. *Distributed Representations of Words and Phrases and their Compositionality*. 2013.
- [18] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2016.
- [19] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020.
- [20] Saeed Saremi and Aapo Hyvärinen. “Neural Empirical Bayes”. In: *Journal of Machine Learning Research* 20 (2019), pp. 1–23.

- [21] Yang Song and Stefano Ermon. “Generative modeling by estimating gradients of the data distribution”. In: *Advances in Neural Information Processing Systems* 32 (2019).
- [22] Nathan C. Frey et al. “Lab-in-the-loop therapeutic antibody design with deep learning”. In: (Feb. 2025).
- [23] Herbert Robbins. “An Empirical Bayes Approach to Statistics”. In: *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability*. Vol. 1. University of California Press, 1956, pp. 157–163.
- [24] Max Welling and Yee W Teh. “Bayesian learning via stochastic gradient Langevin dynamics”. In: *Proceedings of the 28th international conference on machine learning (ICML-11)*. 2011, pp. 681–688.
- [25] Geoffrey E Hinton. “Training products of experts by minimizing contrastive divergence”. In: *Neural computation* 14.8 (2002), pp. 1771–1800.
- [26] Vladimir Vovk et al. “Criteria of efficiency for conformal prediction”. In: *CoRR* abs/1603.04416 (2016).
- [27] Glenn Shafer and Vladimir Vovk. “A Tutorial on Conformal Prediction”. In: *Journal of Machine Learning Research* 9.12 (2008), pp. 371–421.
- [28] Raphael Shu et al. “Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 05. 2020, pp. 8846–8853.
- [29] Fengyuan Dai et al. “Toward De Novo Protein Design from Natural Language”. In: (Aug. 2024).
- [30] Shengchao Liu et al. *A Text-guided Protein Design Framework*. 2025.
- [31] Nikša Praljak et al. “Natural Language Prompts Guide the Design of Novel Functional Protein Sequences”. In: (Nov. 2024).

- [32] Timothy P. Riley et al. “A generalized protein design ML model enables generation of functional de novo proteins”. In: (Mar. 2025).
- [33] Pascal Notin et al. “ProteinGym: Large-Scale Benchmarks for Protein Fitness Prediction and Design”. In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh et al. Vol. 36. Curran Associates, Inc., 2023, pp. 64331–64379.
- [34] Amos Bairoch and Rolf Apweiler. “The SWISS-PROT protein sequence database and its supplement TrEMBL in 2000”. In: *Nucleic Acids Research* 28.1 (Jan. 2000), pp. 45–48. ISSN: 0305-1048.
- [35] Julia Koehler Leman et al. “Sequence-structure-function relationships in the microbial protein universe”. In: *Nature Communications* 14.1 (Apr. 2023). ISSN: 2041-1723.
- [36] Walter M. Fitch. “Distinguishing Homologous from Analogous Proteins”. In: *Systematic Zoology* 19.2 (June 1970), p. 99. ISSN: 0039-7989.
- [37] Allan Larson. “Concepts in Character Macroevolution: Adaptation, Homology, and Evolvability”. In: *The Princeton Guide to Evolution*. Ed. by Jonathan B. Losos et al. Princeton, NJ: Princeton University Press, 2014, pp. 91–99.
- [38] Ronald Wetzel, Herbert LeVine, and Marvin E Goldberg. “Modification of the free energy of unfolding of ribonuclease A by alteration of charge on the protein surface”. In: *Proceedings of the National Academy of Sciences* 75.1 (1978), pp. 39–43.
- [39] Ronald Wetzel. “Protein engineering”. In: *Nature* 300.5894 (1982), pp. 643–646.
- [40] Michael Smith and Sarah Gillam. “Site-directed mutagenesis: a new approach to molecular biology”. In: *Nature* 291.5813 (1981), pp. 545–551.
- [41] James A Wells. “Additivity of mutational effects in proteins”. In: *Biochemistry* 29.37 (1990), pp. 8509–8517.

- [42] Frances H Arnold. “Design by directed evolution”. In: *Accounts of chemical research* 31.3 (1998), pp. 125–131.
- [43] Carol A. Rohl et al. “Protein Structure Prediction Using Rosetta”. In: *Numerical Computer Methods, Part D*. Vol. 383. Methods in Enzymology. Academic Press, 2004, pp. 66–93.
- [44] Hahnbeom Park et al. “Simultaneous Optimization of Biomolecular Energy Functions on Features from Small Molecules and Macromolecules”. In: *Journal of Chemical Theory and Computation* 12.12 (Dec. 2016), pp. 6201–6212.
- [45] Brian Kuhlman et al. “Design of a Novel Globular Protein Fold with Atomic-Level Accuracy”. In: *Science* 302.5649 (Nov. 2003), pp. 1364–1368. ISSN: 1095-9203.
- [46] Steven Henikoff and Jorja G. Henikoff. “Amino acid substitution matrices from protein blocks”. In: *Proceedings of the National Academy of Sciences* 89.22 (1992), pp. 10915–10919.
- [47] Martin Steinegger and Johannes Söding. “MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets”. In: *Nature Biotechnology* 35.11 (Oct. 2017), pp. 1026–1028. ISSN: 1546-1696.
- [48] Stephen F. Altschul et al. “Basic local alignment search tool”. In: *Journal of Molecular Biology* 215.3 (Oct. 1990), pp. 403–410. ISSN: 0022-2836.
- [49] Baris E. Suzek et al. “UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches”. In: *Bioinformatics* 31.6 (Nov. 2014), pp. 926–932. ISSN: 1367-4803.
- [50] Alex Bateman et al. “UniProt: the Universal Protein Knowledgebase in 2025”. In: *Nucleic Acids Research* 53.D1 (Nov. 2024), pp. D609–D617. ISSN: 1362-4962.
- [51] Human Microbiome Project Consortium. “Structure, function and diversity of the healthy human microbiome”. In: *Nature* 486.7402 (2012), pp. 207–214.

- [52] Doug Hyatt et al. “Prodigal: prokaryotic gene recognition and translation initiation site identification”. In: *BMC Bioinformatics* 11.1 (Mar. 2010). ISSN: 1471-2105.
- [53] Amos Bairoch and Rolf Apweiler. “The SWISS-PROT protein sequence database and its supplement TrEMBL in 2000”. In: *Nucleic acids research* 28.1 (2000), pp. 45–48.
- [54] W. Richard McCombie, John D. McPherson, and Elaine R. Mardis. “Next-Generation Sequencing Technologies”. In: *Cold Spring Harbor Perspectives in Medicine* 9.11 (Nov. 2018), a036798. ISSN: 2157-1422.
- [55] *The IUPAC Compendium of Chemical Terminology: The Gold Book*. International Union of Pure and Applied Chemistry (IUPAC), 2025.
- [56] Jaina Mistry et al. “Pfam: The protein families database in 2021”. In: *Nucleic Acids Research* 49.D1 (Oct. 2020), pp. D412–D419. ISSN: 0305-1048.
- [57] Matthias Blum et al. “InterPro: the protein sequence classification resource in 2025”. In: *Nucleic Acids Research* 53.D1 (Nov. 2024), pp. D444–D456. ISSN: 1362-4962.
- [58] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).
- [59] Pascal Vincent et al. “Extracting and Composing Robust Features with Denoising Autoencoders”. In: *Proceedings of the Twenty-fifth International Conference on Machine Learning (ICML ’08)*. ACM, 2008, pp. 1096–1103.
- [60] Geoffrey E. Hinton. “Training Products of Experts by Minimizing Contrastive Divergence”. In: *Neural Computation* 14.8 (Aug. 2002), pp. 1771–1800. ISSN: 1530-888X.
- [61] Yilun Du, Shuang Li, and Igor Mordatch. “Compositional visual generation with energy based models”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6637–6647.

- [62] Aapo Hyvärinen. “Estimation of non-normalized statistical models by score matching”. In: *Journal of Machine Learning Research* 6.Apr (2005), pp. 695–709.
- [63] Yang Song and Stefano Ermon. “Improved techniques for training score-based generative models”. In: *Advances in neural information processing systems* 33 (2020), pp. 12438–12448.
- [64] Jacob Austin et al. *Structured Denoising Diffusion Models in Discrete State-Spaces*. 2023.
- [65] Nate Gruver et al. “Protein Design with Guided Discrete Diffusion”. In: *arXiv preprint arXiv:2305.20009* (2023).
- [66] Prafulla Dhariwal and Alex Nichol. *Diffusion Models Beat GANs on Image Synthesis*. 2021.
- [67] Samuel Stanton et al. *Accelerating Bayesian Optimization for Biological Sequence Design with Denoising Autoencoders*. 2022.
- [68] Vladimir Gligorijević et al. “Function-guided protein design by deep manifold sampling”. In: *bioRxiv* (2021).
- [69] Jonathan Ho and Tim Salimans. *Classifier-Free Diffusion Guidance*. 2022.
- [70] Claude E. Shannon. “The Redundancy of English”. In: *Cybernetics: Transactions of the Seventh Conference*. New York: Josiah Macy, Jr. Foundation, 1951.
- [71] Margaret O. Dayhoff. *Atlas of Protein Sequence and Structure*. Vol. 5, Suppl. 3. Washington, D.C.: National Biomedical Research Foundation, 1978.
- [72] Rodney Staden. “Computer Methods to Locate Signals in Nucleic Acid Sequences”. In: *Nucleic Acids Research* 12.1 Pt 2 (1984), pp. 505–519.
- [73] Michael Gribskov, A. D. McLachlan, and David Eisenberg. “Profile Analysis: Detection of Distantly Related Proteins”. In: *Proceedings of the National Academy of Sciences of the United States of America* 84.13 (1987), pp. 4355–4358.
- [74] Lawrence R. Rabiner. “A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition”. In: *Proceedings of the IEEE* 77.2 (1989), pp. 257–286.

- [75] Kenneth W. Church. “A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text”. In: *Proceedings of the 2nd Conference on Applied Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, 1988, pp. 136–143.
- [76] Anders Krogh et al. “Hidden Markov Models in Computational Biology”. In: *Journal of Molecular Biology* 235.5 (Feb. 1994), pp. 1501–1531. ISSN: 0022-2836.
- [77] Sean R. Eddy. “Profile Hidden Markov Models”. In: *Bioinformatics* 14.9 (1998), pp. 755–763.
- [78] L Steven Johnson, Sean R Eddy, and Elon Portugaly. “Hidden Markov model speed heuristic and iterative HMM search procedure”. In: *BMC Bioinformatics* 11.1 (Aug. 2010). ISSN: 1471-2105.
- [79] Maxat Kulmanov, Mohammed Asif Khan, and Robert Hoehndorf. “DeepGO: predicting protein functions from sequence and interactions using a deep ontology-aware classifier”. In: *Bioinformatics* 34.4 (Oct. 2017). Ed. by Jonathan Wren, pp. 660–668. ISSN: 1367-4811.
- [80] Kevin K. Yang, Alex X. Lu, and Nicolo Fusi. “Convolutions are competitive with transformers for protein sequence pretraining”. In: *bioRxiv* (2022).
- [81] Tristan Bepler and Bonnie Berger. “Learning protein sequence embeddings using information from structure”. In: *International Conference on Learning Representations*. 2019.
- [82] Xueliang Liu. *Deep Recurrent Neural Network for Protein Function Prediction from Sequence*. 2017.
- [83] Ethan C. Alley et al. “Unified rational protein engineering with sequence-based deep representation learning”. In: *Nature Methods* 16.12 (Oct. 2019), pp. 1315–1322. ISSN: 1548-7105.
- [84] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).
- [85] Ahmed Elnaggar et al. *ProtTrans: Towards Cracking the Language of Life’s Code Through Self-Supervised Deep Learning and High Performance Computing*. 2021.

- [86] Alexander Rives et al. “Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences”. In: *bioRxiv* (2020).
- [87] Ali Madani et al. *ProGen: Language Modeling for Protein Generation*. 2020.
- [88] Roshan Rao et al. “Evaluating Protein Transfer Learning with TAPE”. In: *Advances in Neural Information Processing Systems* 32. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 9689–9701.
- [89] Tri Dao et al. *FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness*. 2022.
- [90] Jesse Vig et al. *BERTology Meets Biology: Interpreting Attention in Protein Language Models*. 2021.
- [91] Edith Natalia Villegas Garcia and Alessio Ansuini. *Interpreting and Steering Protein Language Models through Sparse Autoencoders*. 2025.
- [92] Etowah Adams et al. “From Mechanistic Interpretability to Mechanistic Biology: Training, Evaluating, and Interpreting Sparse Autoencoders on Protein Language Models”. In: (Feb. 2025).
- [93] Elana Simon and James Zou. “InterPLM: Discovering Interpretable Features in Protein Language Models via Sparse Autoencoders”. In: (Nov. 2024).
- [94] Erik Nijkamp et al. *ProGen2: Exploring the Boundaries of Protein Language Models*. 2022.
- [95] Mandar Joshi et al. *SpanBERT: Improving Pre-training by Representing and Predicting Spans*. 2020.
- [96] Sabrina J. Mielke et al. *Between words and characters: A Brief History of Open-Vocabulary Modeling and Tokenization in NLP*. 2021.
- [97] Kaj Bostrom and Greg Durrett. *Byte Pair Encoding is Suboptimal for Language Model Pre-training*. 2020.

- [98] Noelia Ferruz, Steffen Schmidt, and Birte Höcker. “ProtGPT2 is a deep unsupervised language model for protein design”. In: *Nature Communications* 13.1 (July 2022). ISSN: 2041-1723.
- [99] Burak Suyunu, Özdeniz Dolu, and Arzucan Özgür. *evoBPE: Evolutionary Protein Sequence Tokenization*. 2025.
- [100] Michel van Kempen et al. “Fast and accurate protein structure search with Foldseek”. In: *Nature Biotechnology* 42.2 (May 2023), pp. 243–246. ISSN: 1546-1696.
- [101] Mingchen Li et al. “ProSST: Protein Language Modeling with Quantized Structure and Disentangled Attention”. In: (Apr. 2024).
- [102] Geoffrey E Hinton and Terrence J Sejnowski. “Learning and relearning in Boltzmann machines”. In: *Parallel distributed processing: Explorations in the microstructure of cognition* 1.282-317 (1986), p. 2.
- [103] Yann LeCun et al. “A tutorial on energy-based learning”. In: *Predicting structured data* 1.0 (2006).
- [104] Pascal Vincent. “A connection between score matching and denoising autoencoders”. In: *Neural computation* 23.7 (2011), pp. 1661–1674.
- [105] Jascha Sohl-Dickstein et al. “Deep unsupervised learning using nonequilibrium thermodynamics”. In: *International Conference on Machine Learning*. PMLR. 2015, pp. 2256–2265.
- [106] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6840–6851.
- [107] Philip A Romero and Frances H Arnold. “Exploring protein fitness landscapes by directed evolution”. In: *Nature reviews Molecular cell biology* 10.12 (2009), pp. 866–876.
- [108] Jisun Kim et al. “Computational and artificial intelligence-based methods for antibody development”. In: *Trends in Pharmacological Sciences* (2023).

- [109] Wengong Jin et al. “Iterative refinement graph neural network for antibody sequence-structure co-design”. In: *arXiv preprint arXiv:2110.04624* (2021).
- [110] Xiangzhe Kong, Wenbing Huang, and Yang Liu. “Conditional antibody design as 3d equivariant graph translation”. In: *arXiv preprint arXiv:2208.06073* (2022).
- [111] Koichi Miyasawa. “An empirical Bayes estimator of the mean of a normal population”. In: *Bulletin of the International Statistical Institute* 38.4 (1961), pp. 181–188.
- [112] Annemarie Honegger and Andreas PluÈckthun. “Yet another numbering scheme for immunoglobulin variable domains: an automatic modeling and analysis tool”. In: *Journal of molecular biology* 309.3 (2001), pp. 657–670.
- [113] James Dunbar and Charlotte M Deane. “ANARCI: antigen receptor numbering and receptor classification”. In: *Bioinformatics* 32.2 (2016), pp. 298–300.
- [114] Roshan M Rao et al. “MSA transformer”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8844–8856.
- [115] Martin Heusel et al. “Gans trained by a two time-scale update rule converge to a local nash equilibrium”. In: *Advances in neural information processing systems* 30 (2017).
- [116] Kishore Papineni et al. “Bleu: a method for automatic evaluation of machine translation”. In: *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 2002, pp. 311–318.
- [117] Derek M Mason et al. “Optimization of therapeutic antibodies by predicting antigen specificity from antibody sequence via deep learning”. In: *Nature Biomedical Engineering* 5.6 (2021), pp. 600–612.
- [118] Peter JA Cock et al. “Biopython: freely available Python tools for computational molecular biology and bioinformatics”. In: *Bioinformatics* 25.11 (2009), pp. 1422–1423.

- [119] Diederik P Kingma et al. “Variational Diffusion Models”. In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer et al. 2021.
- [120] Richard W Shuai, Jeffrey A Ruffolo, and Jeffrey J Gray. “Generative language modeling for antibody design”. In: *bioRxiv* (2021).
- [121] Zeming Lin et al. “Evolutionary-scale prediction of atomic-level protein structure with a language model”. In: *Science* 379.6637 (2023), pp. 1123–1130.
- [122] Karolis Martinkus et al. “AbDiffuser: Full-Atom Generation of In-Vitro Functioning Antibodies”. In: *arXiv preprint arXiv:2308.05027* (2023).
- [123] 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.
- [124] Guillaume Alain and Yoshua Bengio. “What regularized auto-encoders learn from the data-generating distribution”. In: *Journal of Machine Learning Research* 15.1 (2014), pp. 3563–3593.
- [125] Saeed Saremi et al. “Deep energy estimator networks”. In: *arXiv preprint arXiv:1805.08306* (2018).
- [126] Emanuel Parzen. “On estimation of a probability density function and mode”. In: *The annals of mathematical statistics* 33.3 (1962), pp. 1065–1076.
- [127] Noelia Ferruz and Birte Höcker. “Controllable protein design with language models”. In: *Nature Machine Intelligence* 4.6 (2022), pp. 521–532.
- [128] Nataša Tagasovska et al. “A Pareto-optimal compositional energy-based model for sampling and optimization of protein sequences”. In: *arXiv preprint arXiv:2210.10838* (2022).
- [129] Louise Olsson et al. “AbLang: An antibody language model for completing antibody sequences”. In: *bioRxiv* (2022).

- [130] Richard W. Shuai et al. “Generative Language Modeling for Antibody Design”. In: *bioRxiv* (2021).
- [131] Sai Pooja Mahajan et al. “Exploiting Language Models for Protein Discovery with Latent Walk-Jump Sampling”. In: *Machine Learning for Structural Biology Workshop, NeurIPS 2023*. 2023.
- [132] Zarif Ikram, Dianbo Liu, and M Saifur Rahman. “Gradient-guided discrete walk-jump sampling for biological sequence generation”. In: *Transactions on Machine Learning Research* (2024). ISSN: 2835-8856.
- [133] Taylor Joren et al. “Guided Generation of B-cell Receptors with Conditional Walk-Jump Sampling”. In: *Learning Meaningful Representations of Life (LMRL) Workshop at ICLR 2025*. 2025.
- [134] 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.
- [135] Philip A. Romero and Frances H. Arnold. “Exploring protein fitness landscapes by directed evolution”. In: *Nature Reviews Molecular Cell Biology* 10.12 (2009), pp. 866–876.
- [136] Ethan C. Alley et al. “Unified rational protein engineering with sequence-based deep representation learning”. In: *Nature Methods* 16.12 (2019), pp. 1315–1322.
- [137] Ali Madani et al. “ProGen: Language Modeling for Protein Generation”. In: *bioRxiv* (2020).
- [138] Jung-Eun Shin et al. “Protein design and variant prediction using autoregressive generative models”. In: *Nature communications* 12.1 (2021), pp. 1–11.
- [139] Tristan Bepler and Bonnie Berger. “Learning protein sequence embeddings using information from structure”. In: *International Conference on Learning Representations*. 2019.
- [140] Vladimir Gligorijević et al. “Structure-based protein function prediction using graph convolutional networks”. In: *Nature Communications* 12.1 (May 2021). ISSN: 2041-1723.

- [141] Namrata Anand and Possu Huang. “Generative modeling for protein structures”. In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio et al. Curran Associates, Inc., 2018, pp. 7494–7505.
- [142] Raphael R. Eguchi et al. “IG-VAE: Generative Modeling of Immunoglobulin Proteins by Direct 3D Coordinate Generation”. In: *bioRxiv* (2020).
- [143] Sivaraman Balakrishnan et al. “Learning generative models for protein fold families”. In: *Proteins: Structure, Function, and Bioinformatics* 79.4 (2011), pp. 1061–1078.
- [144] Alex Hawkins-Hooker et al. “Generating functional protein variants with variational autoencoders”. In: *BioRxiv* (2020).
- [145] Kristian Davidsen et al. “Deep generative models for T cell receptor protein sequences”. In: *eLife* 8 (2019). Ed. by Detlef Weigel et al., e46935. ISSN: 2050-084X.
- [146] Joe G. Greener, Lewis Moffat, and David T. Jones. “Design of metalloproteins and novel protein folds using variational autoencoders”. In: *Scientific Reports* 8.1 (2018), p. 16189.
- [147] Pascal Vincent et al. “Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion.” In: *Journal of machine learning research* 11.12 (2010).
- [148] Yoshua Bengio et al. “Generalized Denoising Auto-Encoders as Generative Models”. In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*. NIPS’13. Lake Tahoe, Nevada: Curran Associates Inc., 2013, pp. 899–907.
- [149] Kyunghyun Cho. *Noisy Parallel Approximate Decoding for Conditional Recurrent Language Model*. 2016.
- [150] Jason Lee, Elman Mansimov, and Kyunghyun Cho. “Deterministic Non-Autoregressive Neural Sequence Modeling by Iterative Refinement”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018.

- [151] Jiatao Gu et al. “Non-Autoregressive Neural Machine Translation”. In: *International Conference on Learning Representations*. 2018.
- [152] Raphael Shu et al. “Latent-Variable Non-Autoregressive Neural Machine Translation with Deterministic Inference Using a Delta Posterior”. In: *AAAI* (2020).
- [153] Jason Lee et al. “On the discrepancy between density estimation and sequence generation”. In: *arXiv preprint arXiv:2002.07233* (2020).
- [154] Samy Bengio et al. “Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks”. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. NIPS’15. Montreal, Canada: MIT Press, 2015, pp. 1171–1179.
- [155] Marc’Aurelio Ranzato et al. *Sequence Level Training with Recurrent Neural Networks*. 2016.
- [156] Felix Hill, Kyunghyun Cho, and Anna Korhonen. “Learning Distributed Representations of Sentences from Unlabelled Data”. In: *CoRR* abs/1602.03483 (2016).
- [157] 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. ISSN: 0027-8424.
- [158] Jessica L. Gifford, Michael P. Walsh, and Hans J. Vogel. “Structures and metal-ion-binding properties of the Ca<sup>2+</sup>-binding helix–loop–helix EF-hand motifs”. In: *Biochemical Journal* 405.2 (June 2007), pp. 199–221. ISSN: 0264-6021.
- [159] Todor Dudev and Carmay Lim. “Competition among Metal Ions for Protein Binding Sites: Determinants of Metal Ion Selectivity in Proteins”. In: *Chemical Reviews* 114.1 (Jan. 2014), pp. 538–556.
- [160] Martin Reczko et al. “Prediction of hypervariable CDR-H3 loop structures in antibodies”. In: *Protein Engineering, Design and Selection* 8.4 (1995), pp. 389–395.

- [161] Simon P. Kelow, Jared Adolf-Bryfogle, and Roland L. Dunbrack. “Hiding in plain sight: structure and sequence analysis reveals the importance of the antibody DE loop for antibody-antigen binding”. In: *mAbs* 12.1 (2020). PMID: 33180672, p. 1840005.
- [162] Daisuke Kuroda et al. “Computer-aided antibody design”. In: *Protein engineering, design & selection* 25.10 (2012), pp. 507–522.
- [163] Mark L Chiu et al. “Antibody structure and function: the basis for engineering therapeutics”. In: *Antibodies* 8.4 (2019), p. 55.
- [164] James Dunbar and Charlotte M. Deane. “ANARCI: antigen receptor numbering and receptor classification”. In: *Bioinformatics* 32.2 (Sept. 2015), pp. 298–300. ISSN: 1367-4803.
- [165] Jonas Gehring et al. “Convolutional sequence to sequence learning”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1243–1252.
- [166] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. “Sequence to sequence learning with neural networks”. In: *Advances in neural information processing systems* 27 (2014).
- [167] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural machine translation by jointly learning to align and translate”. In: *arXiv preprint arXiv:1409.0473* (2014).
- [168] Jiatao Gu et al. “Non-autoregressive neural machine translation”. In: *arXiv preprint arXiv:1711.02281* (2017).
- [169] Jason Lee, Elman Mansimov, and Kyunghyun Cho. “Deterministic non-autoregressive neural sequence modeling by iterative refinement”. In: *arXiv preprint arXiv:1802.06901* (2018).
- [170] John Lafferty, Andrew McCallum, and Fernando CN Pereira. “Conditional random fields: Probabilistic models for segmenting and labeling sequence data”. In: (2001).
- [171] Ren Yi, Kyunghyun Cho, and Richard Bonneau. “NetTIME: Improving Multitask Transcription Factor Binding Site Prediction with Base-pair Resolution”. In: *bioRxiv* (2021).

- [172] Tomas Mikolov et al. “Recurrent neural network based language model”. In: *Interspeech*. Vol. 2. 3. Makuhari. 2010, pp. 1045–1048.
- [173] Arthur G Street and Stephen L Mayo. “Computational protein design”. In: *Structure* 7.5 (1999), R105–R109.
- [174] Derek N Woolfson. “A brief history of de novo protein design: minimal, rational, and computational”. In: *Journal of Molecular Biology* 433.20 (2021), p. 167160.
- [175] Jacob Glanville et al. “Precise determination of the diversity of a combinatorial antibody library gives insight into the human immunoglobulin repertoire”. In: *Proceedings of the National Academy of Sciences* 106.48 (2009), pp. 20216–20221.
- [176] Thomas Wolf et al. “Transformers: State-of-the-Art Natural Language Processing”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45.
- [177] Itai Gat et al. *Discrete Flow Matching*. 2024.
- [178] Andrew Campbell et al. *Generative Flows on Discrete State-Spaces: Enabling Multimodal Flows with Applications to Protein Co-Design*. 2024.
- [179] Zaixiang Zheng et al. *Structure-informed Language Models Are Protein Designers*. 2023.
- [180] Zeming Lin et al. “Evolutionary-scale prediction of atomic level protein structure with a language model”. In: (July 2022).
- [181] Joshua Meier et al. “Language models enable zero-shot prediction of the effects of mutations on protein function”. In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 29287–29303.
- [182] Ali Madani et al. “Large language models generate functional protein sequences across diverse families”. In: *Nature Biotechnology* 41.8 (Jan. 2023), pp. 1099–1106. ISSN: 1546-1696.

- [183] Alexander Rives et al. “Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences”. In: *Proceedings of the National Academy of Sciences* 118.15 (Apr. 2021). ISSN: 1091-6490.
- [184] Zhidian Zhang et al. “Protein language models learn evolutionary statistics of interacting sequence motifs”. In: *Proceedings of the National Academy of Sciences* 121.45 (Oct. 2024). ISSN: 1091-6490.
- [185] Ratul Chowdhury et al. “Single-sequence protein structure prediction using a language model and deep learning”. In: *Nature Biotechnology* 40.11 (Oct. 2022), pp. 1617–1623. ISSN: 1546-1696.
- [186] Mingjia Huo et al. “Multi-Modal Large Language Model Enables Protein Function Prediction”. In: (Aug. 2024).
- [187] Jin Su et al. “ProTrek: Navigating the Protein Universe through Tri-Modal Contrastive Learning”. In: (June 2024).
- [188] Thomas Hayes et al. “Simulating 500 million years of evolution with a language model”. In: (July 2024).
- [189] Hanjing Zhou et al. *ProtCLIP: Function-Informed Protein Multi-Modal Learning*. 2024.
- [190] Haonan Duan et al. “Boosting the Predictive Power of Protein Representations with a Corpus of Text Annotations”. In: (July 2024).
- [191] Ethan Perez et al. *Film: Visual Reasoning with a General Conditioning Layer*. 2017.
- [192] Ronghui You, Xiaodi Huang, and Shafeng Zhu. “DeepText2GO: Improving large-scale protein function prediction with deep semantic text representation”. In: *Methods* 145 (Aug. 2018), pp. 82–90. ISSN: 1046-2023.
- [193] Ningyu Zhang et al. *OntoProtein: Protein Pretraining With Gene Ontology Embedding*. 2022.

- [194] Alec Radford et al. *Learning Transferable Visual Models From Natural Language Supervision*. 2021.
- [195] Ross Taylor et al. *Galactica: A Large Language Model for Science*. 2022.
- [196] Kathy Y. Wei and Koosh. *MP4: AI Text2Protein Breakthrough Tackles the Molecule Programming Challenge*. Published on Medium by 310 AI. July 2024. url: <https://medium.com/310-ai/mpm4-ai-text2protein-breakthrough-tackles-the-molecule-programming-challenge-870045a8c1ad> (visited on 06/02/2025).
- [197] Andreea Gane et al. *ProtNLM: Model-based Natural Language Protein Annotation*. Preprint. Available at [https://storage.googleapis.com/brain-genomics-public/research/proteins/protnlm/uniprot\\_2022\\_04/protnlm\\_preprint\\_draft.pdf](https://storage.googleapis.com/brain-genomics-public/research/proteins/protnlm/uniprot_2022_04/protnlm_preprint_draft.pdf). Jan. 2022.
- [198] Zach Nussbaum et al. *Nomic Embed: Training a Reproducible Long Context Text Embedder*. 2025.
- [199] Glenn Shafer and Vladimir Vovk. “A Tutorial on Conformal Prediction”. In: *Journal of Machine Learning Research* 9.12 (2008), pp. 371–421.
- [200] Vladimir Vovk et al. *Criteria of efficiency for conformal prediction*. 2016.
- [201] Nathan C. Frey et al. *Protein Discovery with Discrete Walk-Jump Sampling*. 2024.
- [202] A. Bairoch. “The ENZYME database in 2000”. In: *Nucleic Acids Research* 28.1 (Jan. 2000), pp. 304–305. ISSN: 1362-4962.
- [203] Zeming Lin et al. “Evolutionary-scale prediction of atomic-level protein structure with a language model”. In: *Science* 379.6637 (Mar. 2023), pp. 1123–1130. ISSN: 1095-9203.
- [204] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013.
- [205] Nal Kalchbrenner et al. “Neural machine translation in linear time”. In: *arXiv preprint arXiv:1610.10099* (2016).

- [206] Ilya Loshchilov and Frank Hutter. “Decoupled weight decay regularization”. In: *arXiv preprint arXiv:1711.05101* (2017).
- [207] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.
- [208] Matthias Sachs, Benedict Leimkuhler, and Vincent Danos. “Langevin dynamics with variable coefficients and nonconservative forces: from stationary states to numerical methods”. In: *Entropy* 19.12 (2017), p. 647.
- [209] Saeed Saremi and Rupesh Kumar Srivastava. “Multimeasurement Generative Models”. In: *International Conference on Learning Representations*. 2022.
- [210] Yi-Chun Hsiao et al. “Restricted epitope specificity determined by variable region germline segment pairing in rodent antibody repertoires”. In: *MAbs*. Vol. 12. Taylor & Francis. 2020, p. 1722541.
- [211] Matthew IJ Raybould et al. “Five computational developability guidelines for therapeutic antibody profiling”. In: *Proceedings of the National Academy of Sciences* 116.10 (2019), pp. 4025–4030.
- [212] Yilun Du and Igor Mordatch. “Implicit generation and modeling with energy based models”. In: *Advances in Neural Information Processing Systems 32* (2019).
- [213] Chenlin Meng et al. *Concrete Score Matching: Generalized Score Matching for Discrete Data*. 2023.
- [214] Rolf Apweiler et al. “UniProt: the Universal Protein Knowledgebase”. In: *NUCLEIC ACIDS RES* 32 (2004), pp. 115–119.

- [215] Ashish Vaswani et al. “Attention is All You Need”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS’17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010.
- [216] Armin Ronacher. *Jinja2: A Template Engine for Python*. <https://jinja.palletsprojects.com/>. Version 3.x, accessed 2025-06-03. 2008.
- [217] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.
- [218] Sai Pooja Mahajan et al. “Exploiting Language Models for Protein Discovery with Latent Walk-Jump Sampling”. In: *Machine Learning in Structural Biology Workshop at NeurIPS*. 2023.
- [219] Matthias Sachs, Benedict Leimkuhler, and Vincent Danos. “Langevin Dynamics with Variable Coefficients and Nonconservative Forces: From Stationary States to Numerical Methods”. In: *Entropy* 19.12 (Nov. 2017), p. 647. ISSN: 1099-4300.
- [220] Saeed Saremi and Aapo Hyvarinen. *Neural Empirical Bayes*. 2020.
- [221] Hugo Touvron et al. *LLaMA: Open and Efficient Foundation Language Models*. 2023.
- [222] Chankyu Lee et al. *NV-Embed: Improved Techniques for Training LLMs as Generalist Embedding Models*. 2025.
- [223] MetaAI Llama Team. *The Llama 3 Herd of Models*. 2024.
- [224] Melissa A Chiasson et al. “Multiplexed measurement of variant abundance and activity reveals VKOR topology, active site and human variant impact”. In: *eLife* 9 (Sept. 2020). ISSN: 2050-084X.