
THE UNIVERSITY *of* EDINBURGH

# Towards Efficient and Accessible Protein Design with Machine Learning

*Leonardo V. Castorina*


Doctor of Philosophy

CDT Biomedical Artificial Intelligence

School of Informatics

The University of Edinburgh

2025

# Abstract

Proteins are the architects of life on Earth, driving virtually all biochemical processes, from cell signalling and metabolism to immune defence and industrial catalysts. This remarkable functional diversity arises from just twenty building blocks called amino acids arranged in various sequences and lengths. The combinatorial space of possible protein sequences vastly exceeds the number of atoms in the universe, yet nature has explored only a fraction of it.

Protein design aims to navigate this immense design landscape to create proteins with new functions and structures. The emergence of deep learning has significantly improved the speed and accuracy of protein design tools for *de novo* protein binders, enzymes, and neutralising antibodies.

However, the rapid proliferation of new models, each trained on different datasets, presents challenges in evaluating performance and selecting the most suitable tool for a given design task. To address this, we developed PDBench, a fold-balanced benchmark and software toolkit that systematically evaluates model performance across fold types and identifies prediction biases.

This benchmark guided the development of TIMED (Three-dimensional Inference Method for Efficient Design), a Convolutional Neural Network model for protein sequence design using voxel-based representations. Special flavours of TIMED account for biochemical constraints such as charge and polarity, allowing for more precise and tunable designs. To improve accessibility, we released TIMED-Design, a user-friendly web interface and command-line tool, democratising access to state-of-the-art protein sequence design models.

Beyond traditional sequence-based and atomic representations, we introduced a fragment-based representation that abstracts proteins into evolutionarily conserved functional fragments. This approach significantly reduces computational costs while preserving structural and functional signatures. We demonstrated that fragment-based representations improve clustering performance, accelerate functional searches, and serve as effective blueprints for guiding generative protein design.

By making these tools freely available, we aim to empower anyone, beyond researchers, to design the proteins of the future.

# Lay Abstract

Proteins are the building blocks of life, responsible for virtually all the chemical reactions in living organisms. For example, proteins, such as enzymes, speed up chemical reactions in the body and others, such as antibodies, are fundamental parts of the immune system. This wonderful diversity of functions arises from just twenty building blocks called amino acids, arranged in different combinations and quantities.

Protein design is an exciting field that aims to create new proteins with useful functional properties – before nature does. Advances in Artificial Intelligence (AI) have enabled fast and accurate tools for designing proteins. These AI models learn from protein structures using computer vision techniques, similar to those in self-driving cars, or analysing the sequence of amino acids as a "language" with language models like ChatGPT. These tools have led to breakthroughs, including new enzymes for chemical reactions and antibodies against diseases.

Despite these major advances, understanding AI protein design models and their limitations is challenging. Additionally, comparing these models fairly is difficult because they are trained on different datasets. To address this, we created PDBench, a benchmark tool that evaluates the performance of protein design models and identifies strengths and biases.

Using insights from this benchmark, we developed our own protein design model called TIMED (Three-dimensional Inference Method for Efficient Design), that uses 3D pixels to represent proteins. By analysing protein shapes in 3D, TIMED predicts the precise amino acid sequence needed to achieve the desired structure. This allows anyone to generate new shapes of a protein and create the amino acid sequence to work with it in living organisms. These 3D pixels can also incorporate design preferences, such as controlling charge, to guide the sequence design process. To make protein design accessible, we built TIMED-Design, a website that lets users interact with these powerful AI models without programming knowledge.

While TIMED uses 3D representations of proteins, we introduced a new approach. Instead of describing proteins as atoms in space or sequences of amino acids, we break them down into Lego-like fragments, each representing a small and functional part of a protein. These fragments can be assembled to create new proteins and also significantly speed up searching for functionally similar proteins in databases, while reducing storage size by over 90% compared to traditional methods.

By making these tools freely available, we aim to empower anyone, beyond researchers, to design the proteins of the future.

# Acknowledgements

To my family, my parents, my grandma, and my girlfriend, for their never-ending love and support throughout this journey. Without you, this would not have been possible.

To my supervisors, Dr. Kartic Subr and Dr. Christopher W. Wood, for their limitless knowledge, inspiring curiosity, and infinite patience even when research was tough.

To Prof. Lynne Regan and Prof. Teuta Pilizota, were it not for your support during my biochemistry undergraduate, I would have never discovered my love for protein design, let alone do a PhD.

To my friends, Sal and Olivier, with whom we were the three musketeers and future owners of a restaurant called Sale & Olio (SAL LEO OLI) in case the AI bubble bursts.

To my friends at NEC, for helping me discover the world of immunology, and for really pushing me to improve my code and writing. Honestly guys, I saw myself grow from being an amateur into an early-career researcher. A special thanks to the "TCR gang", with señior Filippo Grazioli, Dr. Pierre Machart, Dr. Anja Mösch, and Dr. Federico Errica. It's so fun to do science with you guys and I hope we will have many more occasions to walk around Heidelberg with a glass of Glühwein and Kinderpunsch, discussing anything from video games to the nature of reality.

To my friends at Microsoft Health Futures. A special thanks to my mentor Dr. H. Jabran Zahid, for helping me question *everything* to fully understand problems and his patience and dedication to helping me improve my writing. To Lorenzo Pisani, a software wizard, for his advice on writing clean code. To Dr. Sumit Basu, for his fresh perspective on writing creatively. And to Dr. T. Scott Saponas, because I'm sure he would be offended if I hadn't mentioned him by name.

To my friends at NEC OncoImmunity, learning about proteins and epitopes was fascinating (though not the loopy ones!). I cannot wait to see the results for all the proteins we designed with our majestic protein design pipeline (partially powered by TIMED). A special thanks to Dr. Matheus Ferraz, Dr. Ioannis Vardaxis, and Dr. Giulia Paiardi, for their dedication and amazing support throughout my journey.

To my friends, students and staff, at the University of Edinburgh, the Wells Wood Research Group and the TAG group, the Biomedical AI/Innovation CDT, and the Board Game Night team. You made this journey so fun.

To JKM and LPR for standing by me through the ups and downs of life.

And finally, to Gabriel, for inspiring me with his immense willpower. May you rest in peace.

# List of Publications

List of publications used in this thesis:

- **Castorina, L. V.**, Petrenas, R., Subr, K., and Wood, C. W. (2023). PDBench: Evaluating Computational Methods for Protein-Sequence Design. In *Bioinformatics*.
- **Castorina, L. V.**, Ünal, S. M., Subr, K., and Wood, C. W. (2024). TIMED-Design: Flexible and Accessible Protein Sequence Design with Convolutional Neural Networks. In *Protein Engineering, Design and Selection*.
- **Castorina, L. V.**, Wood, C. W., and Subr, K. (2025). From Atoms to Fragments: A Coarse Representation for Efficient and Functional Protein Design. In *bioRxiv*.

Other publications completed during the PhD but not part of this thesis:

- Li, B. M., **Castorina, L. V.**, Valdés Hernández, M. del C., Clancy, U., Wiseman, S. J., Sakka, E., Storkey, A. J., Jaime Garcia, D., Cheng, Y., Doubal, F., Thrippleton, M. T., Stringer, M., Wardlaw, J. M. (2022). Deep Attention Super-Resolution of Brain Magnetic Resonance Images Acquired under Clinical Protocols. In *Frontiers in Computational Neuroscience*.
- Grazioli, F., Machart, P., Mösch, A., Li, K., **Castorina, L. V.**, Pfeifer, N., Min, M. R. (2022). Attentive Variational Information Bottleneck for TCR–peptide Interaction Prediction. In *Bioinformatics*.
- **Castorina, L. V.**, Grazioli, F., Machart, P., Mösch, A., Errica, F. (2025). Assessing the Generalization Capabilities of TCR Binding Predictors via Peptide Distance Analysis. In *PLOS One*.
- Cotet, T.-S.*, Krawczuk, I.*, Stocco, F., Ferruz, N.*, Gitter, A.*, Kurumida, Y., de Almeida Machado, L., Paesani, F., Calia, C. N.*, Challacombe, C. A., Haas, N.*, Qamar, A., Correia, B. E., Pacesa, M.*, Nickel, L., Subr, K., **Castorina, L. V.***, Campbell, M. J., Ferragu, C., Kidger, P.*, Hallee, L., Wood, C. W., Stam, M. J., Kluonis, T., Ünal, S. M., Belot, E.*, Naka, A., Bustos, A., Torrubia, A., Chu, H., Adaptyv Competition Organizers[1] (2025). Crowdsourced Protein Design: Lessons From the Adaptyv EGFR Binder Competition. In *bioRxiv*.

[1] * Authors are listed in arbitrary order, grouped only by affiliation. Authors marked with an asterisk contributed with exceptional effort.

# Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Leonardo V. Castorina)*

In memory of

Gabriel Riera Alm
11 November 1998 - 10 March 2023

and all our “**T**innies **I**n the **M**eadows” in **ED**inburgh.

# Table of Contents


**1 Introduction** **3**

1.1 Thesis Statement . . . . . 5

1.2 Structure of the Thesis . . . . . 6

**2 Background** **8**

2.1 Fundamentals of Protein Structures . . . . . 8

2.1.1 The Torsion Angles of Proteins . . . . . 8

2.2 Fundamentals of Protein Folding . . . . . 11

2.2.1 Levinthal's Paradox and Protein Folding Pathways . . . . . 12

2.3 Fundamentals of Protein Design . . . . . 13

2.3.1 Fragment-based Design . . . . . 14

2.3.2 Convolutional Neural Networks (CNNs) . . . . . 15

2.3.3 Graph Neural Networks (GNNs) . . . . . 15

2.3.4 Large Language Models (LLMs) . . . . . 15

2.3.5 Generative Models . . . . . 16

2.3.6 Multi-Modal and Other Approaches . . . . . 16

**3 PDBench** **17**

3.1 Introduction . . . . . 17

3.2 Computational Protein Sequence Design . . . . . 18

3.3 Benchmarking Protein Sequence Design . . . . . 18

3.3.1 The PDBench Dataset . . . . . 19

3.3.2 The PDBench Software Toolkit . . . . . 20

3.4 Paper Overview and Contribution . . . . . 22

**4 TIMED** **26**

4.1 Introduction . . . . . 26

4.2 Methodological Foundations of TIMED . . . . . 27

4.2.1 Open-sourcing Voxels with Aposteriori . . . . . 27
4.2.2 TIMED Neural Network Architecture and Design Principles . 29
4.2.3 Voxelising Design Constraints . . . . . 29
4.2.4 Democratising Protein Sequence Design . . . . . 31
4.3 Performance Evaluation and Benchmarking . . . . . 33
4.3.1 Comparative Analysis with the State-of-the-Art . . . . . 33
4.3.2 Structure Validation with AlphaFold 2 . . . . . 34
4.3.3 Effect of Design Constraints on Model Performance . . . . . 34
4.3.4 Experimental Validation at the Adaptyv Bio Competition . . . 35
4.4 Paper Overview and Contribution . . . . . 36

**5 Fragments** **49**
5.1 Introduction . . . . . 49
5.2 Reimagining Protein Fragments . . . . . 50
5.3 Methodological Foundations of Fragment Techniques . . . . . 52
5.3.1 How to Abstract a Protein . . . . . 52
5.3.2 A Fragment Detection Framework . . . . . 53
5.3.3 Fragments Functional and Structural Evaluation . . . . . 55
5.4 Evaluating Fragments for Protein Functions and Design . . . . . 58
5.4.1 Fragments Capture Physicochemical and Structural Properties 58
5.4.2 Embedding Protein Function: Clustering with Fragment-Based Representations . . . . . 58
5.4.3 Fragments for Efficient Functional Searches . . . . . 59
5.4.4 Fragments as Blueprints for Generative Models . . . . . 59
5.5 Paper Overview and Contribution . . . . . 60

**6 Conclusion** **72**
6.1 Summary of Contributions . . . . . 72
6.2 Limitations and Future Work . . . . . 73

**A Supplementary Materials** **75**
A.1 PDBench . . . . . 75
A.2 TIMED . . . . . 84
A.2.1 TIMED Architecture . . . . . 84
A.2.2 Unbalanced Performance Comparison . . . . . 85
A.3 Fragments . . . . . 95

A.3.1 Sliding window algorithm . . . . . 95
A.3.2 Fragment Detection Metrics and Details . . . . . 95
A.3.3 Fragment Detection Accuracy . . . . . 98
A.3.4 Physico-chemical Properties in Fragments . . . . . 98
A.3.5 Fragment Coverage By Fold and Resolution . . . . . 99
A.3.6 H-bond Between Fragments Coverage . . . . . 100
A.3.7 H-bond Within Fragments Coverage . . . . . 101
A.3.8 Accessibility Coverage . . . . . 102
A.3.9 Charge Coverage . . . . . 103
A.3.10 Polarity Coverage . . . . . 104
A.3.11 Secondary Structure Coverage . . . . . 105
A.3.12 Distance Spearman Correlations . . . . . 106

**Bibliography** **123**

# Acronyms

**ARI** Adjusted Rand Index. 56, 58

**AUROC** Area Under the Receiver Operating Characteristic Curve. 56, 57, 59

**CASP** Critical Assessment of Structure Prediction. 12, 31

**CATH** Class, Architecture, Topology, Homologous Superfamily. 17, 20, 73

**CLI** Command-Line Interface. 5, 6, 27, 31–33

**CNN** Convolutional Neural Network. 4–7, 14, 15, 18, 22, 27, 36, 74

**EGF** Epidermal Growth Factor. 3, 7, 10, 35, 36

**EGFR** Epidermal Growth Factor Receptor. 3, 10, 35, 36

**fixbb** Fixed Backbone Design. 18, 22

**GAN** Generative Adversarial Network. 16

**GMM** Gaussian Mixture Model. 57

**GNN** Graph Neural Network. 4, 14, 15, 18, 22, 74

**GO** Gene Ontology. 56

**Kd** Dissociation Constant. 35, 36, 73

**LLM** Large Language Model. 4, 13–16, 74

**MAE** Mean Absolute Error. 34

**MHC** Major Histocompatibility Complex. 3

**mRNA** messenger RNA. 8

**MSA** Multiple Sequence Alignment. 12, 31

**NDCG** Normalized Discounted Cumulative Gain. 56, 57

**NMI** Normalized Mutual Information. 56

**PCoA** Principal Coordinate Analysis. 56

**PDB** Protein Data Bank. 3, 4, 9, 10, 12, 19, 32, 56, 57, 73, 74

**PFD** Protein Function Dataset. 56, 57

**RIF** Rotamer Interaction Field. 51

**RMSD** Root Mean Square Deviation. 31, 33, 34, 36, 56–59

**ROC** Receiver Operating Characteristic. 54, 58

**SMILES** Simplified Molecular Input Line Entry System. 52

**t-SNE** t-Distributed Stochastic Neighbor Embedding. 57

**TCR** T Cell Receptor. 3

**TERM** Tertiary Structural Motif. 51, 52, 74

**UI** User Interface. 5, 6, 26, 27, 30–33

**VAE** Variational AutoEncoder. 16, 61

# Chapter 1

# Introduction

Proteins are the architects of life on Earth, performing virtually all chemical reactions essential for biological function. Their diverse roles span the entire spectrum of living systems. Specialised proteins called enzymes catalyse chemical reactions from DNA replication to metabolic processes [4]. Transport proteins like haemoglobin deliver oxygen to tissues throughout the body [4]. Cell signalling is mediated by protein ligands and receptors. For example, Epidermal Growth Factor (EGF) binds Epidermal Growth Factor Receptor (EGFR) to regulate development, and it is also involved in a variety of cancers [4]. Similarly, the immune system relies on intricate protein-receptor interactions, such as those between T Cell Receptors (TCRs) and Major Histocompatibility Complexs (MHCs) that drive adaptive immune responses [4]. Proteins also play crucial structural roles, as shown by keratins which form hair, nails, and feathers [4].

This remarkable functional diversity emerges from just twenty amino acids arranged in different combinations and lengths. The potential design space is staggering: for an average protein of 200 units, there are more potential sequences than the number of available atoms in the universe [113]. Nature has explored only a small fraction of this design space through evolution, leaving an enormous landscape of possibilities largely unexplored [12]. In contrast, the number of available structures in the Protein Data Bank (PDB) is about 236K as of May 2025[1], which is an infinitesimal fraction of all proteins. Importantly, these structures are themselves models—refined against experimental data such as electron density, rather than direct observations. Therefore, any model trained on them learns to reproduce these refined approximations, inheriting their assumptions and biases rather than accessing absolute physical ground truth [106]. This bias is compounded by the limitations of the experimental techniques used to

[1] https://www.rcsb.org/stats/growth/growth-released-structures

determine protein structures: X-ray crystallography requires crystallisable proteins, Nuclear Magnetic Resonance (NMR) is limited to small proteins, and Cryo-Electron Microscopy is used for large proteins [106]. As a result, the PDB reflects only a structurally accessible subset of the proteome, and any evaluation of design or prediction models is necessarily constrained to this visible fraction. In addition, while proteins naturally explore a dynamic ensemble of conformations, both experimental and computational techniques typically aim to produce a single, static representative structure or (at best) a small ensemble. This simplification is practical for interpretation and modelling, but it obscures the intrinsic flexibility of proteins, particularly in disordered regions or allosteric transitions [60, 74]. As such, models trained on these datasets are implicitly biased towards a narrow snapshot of the protein's conformational landscape.

Protein design aims to explore this vast space to develop novel proteins with new functions and structures. Using physicochemical principles and physics-based methods, the field has achieved remarkable successes, including stable four-helix bundles[62], *de novo* protein folds like TOP7 [71], enzymes with new catalytic motifs for Kemp elimination [99], and vaccines that elicit immune responses [32].

Early successes in computational protein design heavily relied on physics-based methods, like Rosetta. These methods relied on fragment-assembly and calibrated energy functions (van der Waals, electrostatics, solvation) to search sequence/structure space via Monte Carlo and energy minimization, with no explicit "training" phase and a CPU-intensive cost per design [71]. By contrast, modern deep-learning pipelines shift most of the compute to a GPU-driven training phase, after which inference is orders of magnitude faster. [69]. These models learn implicit sequence-structure relationships from growing datasets, improving the performance further as more structures are deposited in the PDB [69]. Successes of deep learning methods include *de novo* protein binders [52], neutralising antibodies [102], and enzymes [88, 123].

The research presented here happened during the transformative era of protein design powered by deep learning. Therefore, it makes use of and builds upon the most important computational tools in the field such as protein folding models [1, 67, 97, 122], Large Language Models (LLMs) for folding and sequence ranking [48, 57, 97, 109], generative models for protein backbone generation [2, 116], and Convolutional Neural Network (CNN)[25, 93, 125] and Graph Neural Network (GNN)[36, 107] methods for protein sequence design. Rather than developing isolated tools, this work focuses on creating efficient and accessible tools that improve and democratise protein design methods.

The rapid proliferation of diverse methods, each trained on different datasets, has created a significant challenge for designers: determining the most appropriate method for their design task. This requires a more nuanced view of performance beyond simple accuracy. To address this gap, we developed PDBench, a fold-balanced benchmark set and software toolkit (Chapter 3), which calculates biologically meaningful metrics like prediction bias across fold types, revealing biases and allowing designers to make informed decisions based on their target protein.

As most tools remained closed-source and difficult to use, we decided to democratise these tools to allow anyone to use them. We led the reimplementation and retraining of all the CNN models for protein sequence design, developing our flagship model TIMED (Chapter 4). With TIMED-Charge and TIMED-Polar we showed that models could incorporate design constraints such as charge and polarity, allowing designers to specify desired properties during inference. We developed these models alongside TIMED-Design, a User Interface (UI) and Command-Line Interface (CLI) that allow researchers without computational expertise to use state-of-the-art protein design methods.

Finally, we addressed the growing computational demands of protein design models, which typically use to atom- or residue-level representations that scale exponentially with protein length [65]. We proposed a fundamentally different approach: representing proteins using evolutionarily conserved functional fragments (Chapter 5). We showed that fragments significantly reduce the memory requirements — by up to 99% — while preserving the functional fingerprints of proteins. We used fragment-based representations for functional clustering, fast database searches, and as blueprints to guide generative models.

Together, these contributions aim to democratise protein design, allowing researchers with diverse computational resources and expertise levels to discover and design novel proteins.

## 1.1 Thesis Statement

Protein design has been transformed by deep learning, yet key challenges remain: (1) a lack of standardized and biologically meaningful benchmarks to evaluate models, (2) limited control over sequence design constraints, (3) restricted accessibility to models, and (4) growing computational demands of protein design representations.

This thesis addresses these challenges by developing efficient, accessible, and interpretable computational approaches for protein design. Specifically, it argues that:

1. **Current protein design models lack rigorous and biologically meaningful evaluation.** We developed PDBench, a fold-balanced benchmark and software toolkit, to address this gap. PDBench provides biologically relevant metrics to systematically evaluate models, revealing biases across protein folds and improving comparability in sequence design research.

2. **Integrating biochemical constraints into sequence design improves model control and tunability.** We developed TIMED, a CNN for protein sequence design that uses voxel-based representations and incorporates constraints like charge and polarity, allowing designers to guide the sequence design with constraints.

3. **Despite advances in AI-driven protein design, accessibility remains a major barrier to adoption.** Many state-of-the-art models are closed-source or computationally demanding, limiting broader use. We reimplemented, trained, and publicly released all TIMED models, ensuring reproducibility and accessibility. To lower technical barriers, we developed TIMED-Design, a user-friendly UI and CLI, enabling researchers to use state-of-the-art design methods without requiring computational expertise.

4. **Current protein representations scale poorly with sequence length, limiting the efficiency and interpretability of large-scale protein design.** Models based on traditional atomic- and sequence- representations become exponentially more expensive as protein length increases, making them impractical for large proteins or complexes. We introduce fragments as coarse-grained representation that preserves functional information while significantly reducing computational costs. This allows faster functional clustering, efficient database searches, and scalable generative modelling.

Together, these contributions aim to improve and democratize protein design, allowing everyone to design novel proteins.

## 1.2 Structure of the Thesis

This thesis first discusses the background of protein design and then delves into the three main chapters:

- In Chapter 3, we present PDBench [24], a software toolkit and benchmark dataset to evaluate protein sequence design methods. The rapid advancement of deep

learning models for protein design has created a need for rigorous, biologically meaningful evaluation metrics beyond traditional sequence recovery. PDBench provides a curated fold-balanced dataset of 595 proteins alongside a model-agnostic evaluation framework to analyse model performances across different protein folds. Our analysis reveals significant fold-specific trends and the impact of backbone resolution on model performance. PDBench also played a key role in identifying early biases, informing the development of TIMED, our CNN-based method for protein design introduced in the next chapter.

- In Chapter 4, we present TIMED [25], a CNN-based method for protein sequence design that uses voxel-based representations of protein structures. We introduce TIMED-Charge and TIMED-Polar, novel flavours of TIMED that integrate biochemical constraints such as charge and polarity directly into the voxel space, allowing for constrained sequence design at inference time. To improve accessibility, we propose Aposteriori, an open-source voxelisation toolkit, and TIMED-Design, a user-friendly interface for researchers without computational expertise. We benchmark TIMED and its flavours against state-of-the-art sequence design models using PDBench and discuss its successful application in the Adaptyv Bio competition, where it ranked in the top 3% with the second lowest sequence identity among successful redesigns of the EGF.

- In Chapter 5, we present fragment-based representations as coarse and functional representations for protein structures. Traditional protein design methods often rely on computationally expensive representations that scale poorly with protein length. As an alternative, we propose decomposing protein structures into evolutionarily conserved fragments derived from structural motifs. Our fragment-based representation significantly reduces dimensionality by up to 99% while preserving functional information, allowing for orders-of-magnitude speed-ups in functional searches and better clustering performance compared to traditional methods. Additionally, we show that fragments can be effective blueprints by guiding generative models towards functionally plausible protein backbone structures.

Each chapter begins with an introduction that contextualises the work within the field and outlines its contributions. We then present the methodological foundations, results, and key findings before discussing their broader impact. Finally, each chapter concludes by presenting the original publication.

# Chapter 2

# Background

## 2.1 Fundamentals of Protein Structures

The "coding" regions of DNA encode the proteins of an organism. First, RNA Polymerase transcribes DNA into messenger RNA (mRNA). Then, the ribosomes translate the mRNA into proteins [4].

Proteins are linear polymers composed of twenty building blocks, known as amino acids, connected by peptide bonds (see Figure 2.1). Amino acids consist of a common backbone and one of twenty side chains with specific physicochemical properties. The side chains of amino acids are attached to the C$\alpha$ carbon, while the C$\beta$ is the first carbon atom of the side chain (with the exception of Glycine which lacks one) [4].

The 3D structure of a protein is encoded by its amino acid sequence. Side chains have diverse properties such as charge, polarity, size, and hydrophobicity, which affect how the protein folds. Linear sequences of peptides organise into secondary structures, primarily helical ($\alpha$-helix) or flat ($\beta$-sheet), through hydrogen bonds between the carbonyl oxygen and amide hydrogen atoms along the peptide backbone. These secondary structures assemble through side chain and backbone interactions to create the tertiary (3D) structure. In many proteins, tertiary structures associate to form functional quaternary complexes [4]. Figure 2.2 illustrates this structural hierarchy using Ubiquitin as an example.

### 2.1.1 The Torsion Angles of Proteins

Proteins usually fold exposing charged and polar amino acids to the solvent, and hydrophobic amino acids to the core, therefore minimising the system's free energy [101].

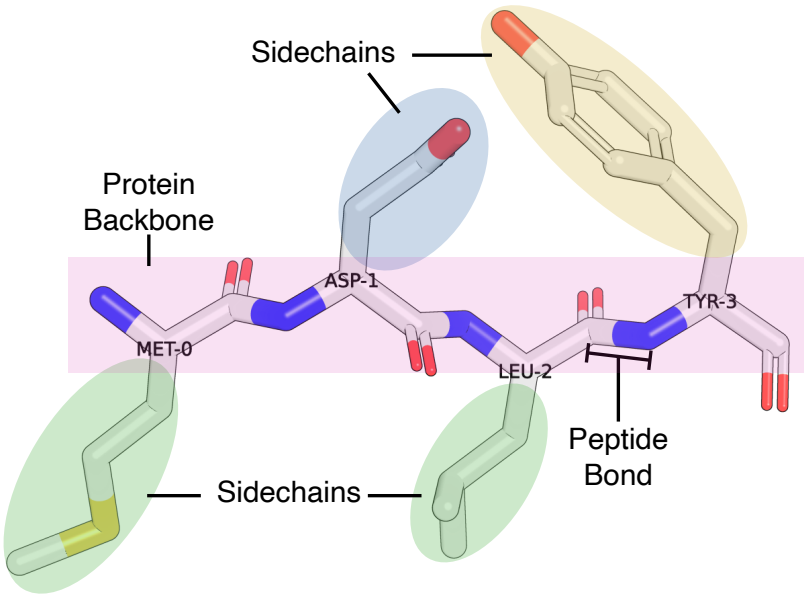



Figure 2.1: The Anatomy of a Protein. Peptide bonds connect a polypeptide of sequence Met-Asp-Leu-Tyr. The backbone is common to all amino acids, while the side chains are unique to each amino acid and contribute to the protein structure and function.

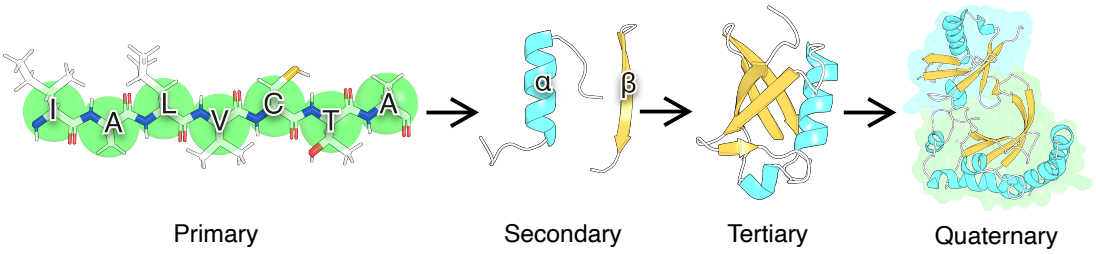



Figure 2.2: Hierarchical Structures of Proteins. Primary structure (linear amino acid sequence connected by peptide bonds), secondary structure (local folding into α-helices and β-sheets stabilised by hydrogen bonds), tertiary structure (complete 3D structure of Ubiquitin PDB code: 1UBQ), and quaternary structure (functional assembly of multiple polypeptides, shown by the yeast Vps23 UEV domain in complex with Ubiquitin). Each level of organization builds upon the previous level to create functional protein molecules.

These preferences influence the backbone torsion (or dihedral) angles. The $\phi$ angle describes the rotation around -N-C$\alpha$- bond, while the $\psi$ angle describes the rotation of the -C$\alpha$-C- bond, as shown in Figure 2.3 [101].

Although these bonds can rotate, providing some conformational flexibility, the steric hindrance ("bulkiness") from the side chains and backbone geometry restricts the range of energetically favourable angles [101]. A common way to analyse the

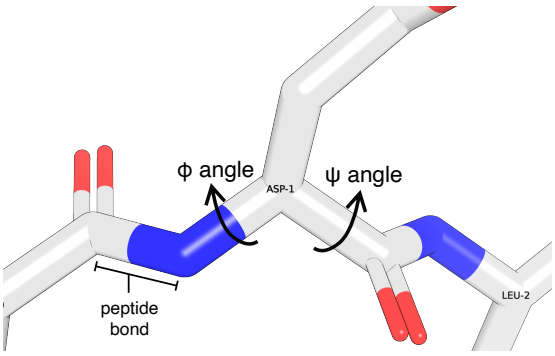
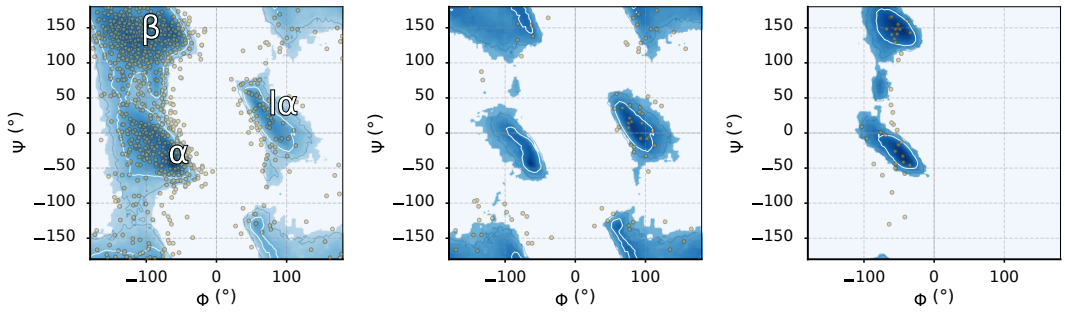



Figure 2.3: Torsion angles ϕ and ψ of a protein.

conformational space of a protein is to use Ramachandran plots, a density plot of the ψ angles on the y-axis and ϕ on the x-axis as shown in Figure 2.4 [94, 101]. Typically, α-helical structures occupy the middle-left quadrant ($\phi \approx -60°$ and $\psi \approx -45°$) and β structures in upper-left quadrant ($\phi \approx -120°$ and $\psi \approx 120°$) of a Ramachandran plot (Figure 2.4a). Left-handed helices ($l\alpha$), rare in proteins, occupy the middle right region of the plot. Glycine lacks a side chain on its Cα atom, which decreases the steric hindrance and allows more flexibility (Figure 2.4b). On the other hand, Proline cyclic side chain heavily constrains its backbone conformations, limiting the range of torsion angles (Figure 2.4c).




(a) All amino acids (b) Glycine (c) Proline

Figure 2.4: Comparison of Ramachandran plots for different residue types, showing the distribution of backbone torsion angles ϕ (x-axis) and ψ (y-axis). (a) All amino acid residues, showing clustering in regions corresponding to α-helices and β-sheets; (b) Glycine, showing a broader distribution due to reduced steric hindrance; and (c) Proline, showing a restricted distribution due to its cyclic side chain. Created using Ramachandran Plotter and the EGFR in complex with EGF (PDB code: 1IVO).

Understanding these conformational preferences is essential for protein design, as

they determine which sequences could fold into stable functional structures. Computational methods for protein design attempt to reconcile sequences and torsion angles data to produce novel proteins.

## 2.2 Fundamentals of Protein Folding

Protein folding begins as polypeptide chains come out of the ribosomes during translation. In the 1960s, Anfinsen [10] showed that proteins spontaneously fold into their shape, using only their primary sequence. Anfinsen [10] denatured a ribonuclease protein and showed that it could refold and retain catalytic activity, meaning that the sequence contained all the information required for folding.

Beyond covalent peptide bonds, weaker non-covalent interactions collectively stabilise the 3D structure and drive folding. There are four main types:

1. **Hydrogen Bonds** form when Hydrogens (H) bound to very electronegative atoms like Nitrogen (N) or Oxygen (O), interact with other electronegative atoms. In proteins, hydrogen bonds occur in secondary structures and between regions of the protein. In α-helices they stabilise the regular helical structure separated by four residues, while in β-sheets they connect nearby strands [4, 101].

2. **Electrostatic interactions** form between oppositely charged groups, for example the side chains of Glutamate or Aspartate (negative) and Lysine or Arginine (positive). These can be both attractive and repulsive and are very important for enzyme substrate interactions [4, 101].

3. **Van der Waals attractions** form between closely positioned atoms due to fluctuations in the electron distribution. Although these are weak forces, they are collectively significant when atoms are densely packed, for example in the protein core [4, 101].

4. **Hydrophobic effects** form as a result of hydrogen bonds forming and grouping polar molecules to minimise energy. When hydrophobic side chains cluster in the core, they are shielded from water, thus reducing the entropic penalty of water interacting with non-polar surfaces [4, 101].

Although individually weak, these forces along with backbone conformational constraints, collectively shape protein folds.

### 2.2.1 Levinthal's Paradox and Protein Folding Pathways

Despite the native structure of a protein being thermodynamically favourable, the *exact* way some proteins get to that state is still not fully understood. In 1968, Levinthal [73] illustrated this paradox with a simple thought experiment: A protein of 100 amino acids would need to sample all $10^{100}$ potential conformations (assuming just 10 possible $\phi$ and $\psi$ angle combinations per residue). Even if the protein sampled each conformation at an impossible rate of one picosecond per structure, the search would take longer than the age of the universe [73].

Yet, experimentally, proteins fold within milliseconds [42]. Therefore, protein cannot search the conformational space randomly, meaning that folding must proceed through preferred pathways conserved by evolution [73]. Folding is understood to proceed through an energy landscape with the shape of a "rugged funnel", with wells of local optima representing the intermediate states and the native state occupying the lowest energy state [39].

Predicting the protein structure from the amino acid sequence has been a long-standing challenge in the field. In 1994, the Critical Assessment of Structure Prediction (CASP) challenge was introduced to systematically evaluate progress in protein folding methods [87]. Researchers attempted to predict a set of structures based on their sequences before their release in the PDB [87]. Over multiple CASP iterations, template-based modelling remained dominant, while physics-based approaches struggled with computational feasibility [15]. Template-based methods rely on known structures as scaffolds, aligning sequences to homologous proteins, whereas physics-based approaches attempt to fold proteins from first principles using energy minimization and molecular dynamics simulations [15]. Deep learning began to transform the field in the late 2010s, with models like DeepMind's first AlphaFold iteration demonstrating significant improvements in CASP13 [5].

In 2021, Jumper et al. [67] released AlphaFold 2, a protein folding model that achieved atomic-level accuracies on the CASP14 protein folding challenge. AlphaFold uses sequence databases through Multiple Sequence Alignments (MSAs) and structure database searches, to create prior information about the protein structure [1, 67]. MSAs techniques use the evolutionary relationship between proteins, guided by the central principle that similar sequences tend to have similar three-dimensional folds [30]. This information is then processed by the EvoFormer, a transformer that attempts to reconcile structure and sequence information by learning the relationships of all the elements

involved [1, 67]. This is then used by the structure module which predicts the position of each amino acid [1, 67]. Shortly after, LLMs methods such as OmegaFold and ESMFold have been shown to be as effective at a fraction of the computational power [77, 122].

Although folding models do not solve Levinthal's paradox, they have paved the way for a new generation of protein design, allowing designers to rapidly obtain fairly accurate predictions of the *in vivo* structure.

## 2.3 Fundamentals of Protein Design

Richard Feynman famously said "What I cannot create, I do not understand", which captures the philosophy of protein design: understanding proteins, their interactions, and their functions by creating new ones [79]. At the core of protein design lies a paradox similar to Levinthal's: the number of possible sequences for a typical protein of 200 amino acids is $1.6 \times 10^{2601}$, a number larger than the estimated number of atoms in the observable universe, $10^{80}$ [113]. Even if only a small fraction of these sequences fold into stable structures, the design space remains astronomically vast and unexplored. Thus, several physically possible proteins are waiting to be discovered. Many of which could potentially performing completely new functions.

Navigating the immense design space cannot be done randomly and exhaustively. Protein design approaches that attempt to navigate this space can be broadly categorised into minimal, rational, and computational design.

- **Minimal design** relies on fundamental physicochemical properties of amino acids such as polarity and hydrophobicity to generate simple secondary structures like $\alpha$ helices or $\beta$ strands [120]. A classic example is $\alpha_1$, one of the first *de novo* designed proteins. It consists of a 13-amino-acid amphipathic helix, with a Leucine-rich hydrophobic face and a hydrophilic face containing Lysines and Glutamates. This pattern made the peptide self-associate into tetrameric or hexameric assemblies [61].

- **Rational design** uses structural and biochemical patterns from natural proteins to guide sequence selection, for example by incorporating functional motifs [120]. A notable example is the successful incorporation of a tetrahedral Zinc binding site into a designed four helix bundle [95]

[1] $20^{200}$

- **Computational design** uses computational models to evaluate structures and optimise sequences for given structural targets [120]. One of the earliest successes of this approach was TOP7, a *de novo* protein designed using fragment-based methods within the Rosetta software suite [71].

Computational tools for protein design started with physics-based and fragment-based methods and have evolved into modern deep learning methods. Pre-deep learning methods used structural motifs and energy minimisation to guide design. Whereas, post-deep learning methods are data-driven algorithms learning structural patterns from large databases. Nowadays, most designers use combinations of these methods to design proteins. Figure 2.5 illustrates the five key representations of proteins and provides a brief overview of the methods that use them below:

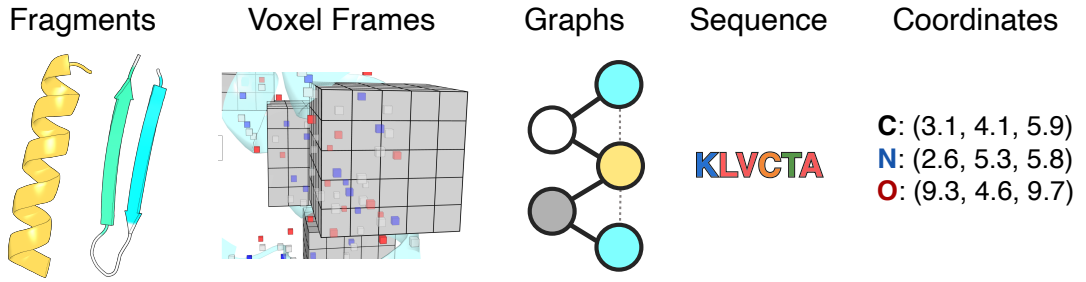



Figure 2.5: Representations of Proteins. Fragment-based methods use segments of conserved protein structures to simplify design complexity. Frames are voxel grids of atomic positions, typically used in CNNs. Graphs-based approaches, typically used in GNNs represent proteins as nodes (amino acids) and edges (peptide bonds or spatial proximity). Sequence-based methods, common in LLMs treat proteins as linear strings of amino acids. Finally, diffusion-based methods use atomic coordinates of protein backbones to generate realistic structures through iterative de-noising processes.

### 2.3.1 Fragment-based Design

Fragment-based Design methods, such as Rosetta, assemble protein structures using short sequence and structural fragment (3-9 amino acids long). This approach reduces the design complexity by using physically plausible backbone conformations and naturally occurring sequences making it computationally feasible [71]. Fragments are also the basis of our coarse representation for protein structure (Chapter 5). These methods traditionally rely on computationally expensive energy algorithms. Additionally, the

initial fragment libraries may introduce biases based on the input dataset, which may limit generalisation [71]

### 2.3.2 Convolutional Neural Networks (CNNs)

Generally, deep learning approaches can be more computationally lightweight and equally or more accurate than physics-based methods[24]. CNNs were among the first deep learning methods applied to protein design, building on classical neural networks methods like SPIN2 [89]. CNN models for fixed-backbone sequence design capture local 3D backbone environments using voxel representations, analogous to 2D image classification. Each amino acid position in the structure is converted to an overlapping voxel grid (a *frame*). The model then classifies each frame independently, producing a probability distribution over the twenty amino acids. Examples include ProDCoNN [125], DenseCPD and DenseNet [93], and TIMED [25], each with a different architecture to capture structural features. glsplcnn are limited by their local receptive fields, which may fail to capture long-range dependencies essential for global fold determination [125].

### 2.3.3 Graph Neural Networks (GNNs)

GNNs offer a complementary approach to CNNs by representing proteins as graphs, where amino acids are nodes connected by edges representing peptide bonds or spatial proximity. Unlike CNNs, GNNs are inherently rotation-invariant and can model both local and global sequence dependencies using message passing operations [35, 107]. Examples of GNN-based methods are ProteinSolver [107] which pioneered the approach, and ProteinMPNN which is one of the most widely used tools for fixed-backbone sequence design [35]. While GNNs model long-range dependencies better than glsplcnn, they are sensitive to graph construction choices, like node and edge features and cutoff distances [107].

### 2.3.4 Large Language Models (LLMs)

While CNNs and GNNs typically focus on the backbone, LLMs use sequence data which is more abundant. These models treat protein sequences as a form of “language”, and learn patterns by predicting the missing amino acids based on a “prompt” (partial sequence) or sampling noise. These models are powerful and flexible and were used

for sequence generation, as embeddings, for ranking sequences, and for designing novel proteins such as new enzymes [48, 88, 97]. Examples of LLMs used in protein design include ESM (and all of its variants) [77, 97], ProtGPT [48], ProtTrans [44], ZymCTRL [88], and OmegaFold [122]. LLMs may lack structural information, making it difficult to enforce spatial or biophysical constraints.

### 2.3.5 Generative Models

Beyond protein sequence design, several powerful models have been developed for backbone generation, simultaneous sequence and backbone design, and sequence optimisation tools. Early successes in this area included generative models such as Generative Adversarial Networks (GANs) [8, 9] and Variational AutoEncoders (VAEs) [43, 56] to generate backbones and/or sequences. More recently, diffusion models, which gradually de-noise random atomic coordinates into structured data, such as RFDiffusion [116] and EVODiff [2] have demonstrated abilities in generating natural-looking protein backbones. Generative models can hallucinate physically implausible structures or sequences, especially when trained on limited or biased datasets. Evaluating their outputs remains challenging due to the lack of standard metrics for structural novelty and functional correctness [119].

### 2.3.6 Multi-Modal and Other Approaches

Recently, more advanced tools were proposed that integrate multimodal protein data, such as ESM3 [57] and Chai-1 [28], for applications such as folding, embeddings and design. Specialised tools are also emerging, such as BindCraft [90] for protein binder design and ProtRL [105], which uses reinforcement learning to guide generation in language models towards desired properties. There are a plethora of deep learning tools for protein design. These approaches often require large amounts of data and computation, which may increase overall complexity and lower interpretability [28, 57, 90, 105]. Recent reviews [3, 38, 46, 49] cover many more methods that this list has and will inevitably miss. There has never been a better time for protein design.

# Chapter 3

# PDBench

## 3.1 Introduction

In this chapter, we focus on evaluating methods for computational sequence design (or inverse folding) – the process of designing amino acid sequences that fold into a given protein backbone.

Despite significant advancements in computational protein design, at the time we developed PDBench, there was no standardised benchmarking tool. Various models ranging from physics-based to deep learning approaches had been developed. However, it remained difficult to systematically evaluate their performance across diverse protein architectures.

This chapter introduces the PDBench [24], a benchmarking toolkit and fold-balanced dataset of 595 proteins carefully selected from the Class, Architecture, Topology, Homologous Superfamily (CATH) dataset [37] to improve model evaluation. PDBench serves three key purposes:

1. Curating a fold-balanced dataset that captures the diversity of protein structures.
2. Establishing biologically meaningful evaluation metrics beyond accuracy.
3. Standardizing visualization tools for comparing models across structural folds and experimental resolutions.

Unlike previous randomly sampled benchmarks, PDBench helps researchers identify fold-specific strengths and weaknesses in their models, guiding the development of more robust protein design tools.

In this chapter, we first review the landscape of protein design strategies, with particular focus on computational approaches. Then, we explore the PDBench dataset construction and evaluation metrics. Finally, we demonstrate its utility through comparative analysis of leading protein design algorithms.

## 3.2 Computational Protein Sequence Design

There is a wide variety of tools for computational protein sequence design. Physics-based methods, such as Rosetta's Fixed Backbone Design (fixbb) [78] and EvoEF2 [64], use energy functions to iteratively determine optimal amino acid side chains that satisfy the backbone constraints. While accurate, physics-based methods can be computationally demanding for end users. On the other hand, deep learning methods shift the computational burden to the training phase, allowing end users to run these models with relatively modest computational resources - even those available in modern smartphones.

By the time we developed PDBench, the primary deep learning models for protein sequence design were CNNs, such as ProDCoNN [125], DenseCPD and DenseNet [93], and TIMED [25], GNNs such as ProteinSolver [107].

The plethora of computational approaches available creates a significant challenge for designers: selecting the best tool for their protein design problem. At the time, model evaluation was restricted to basic metrics applied to randomly selected protein sets. Additionally, CNN-based models were inaccessible to designers as their code and weights were not publicly available.

To address the limitations of existing benchmarks and improve the model accessibility, we created and open-sourced PDBench and Aposteriori. PDBench is a dataset and software tool for benchmarking protein sequence design models, while Aposteriori is a library for 3D voxelisation of protein structures. Using these tools, we re-trained state-of-the-art CNN models on the same dataset, made them freely available, and systematically compared them against established methods like EvoEF2, Rosetta fixbb, and ProteinSolver.

## 3.3 Benchmarking Protein Sequence Design

Protein sequence design is typically used for re-designing the sequences of known protein backbones or designing sequences for computationally generated backbones. The

simplest metric for quantifying model performance is sequence recovery (or sequence accuracy) on known proteins. This is the percentage of correctly identified amino acids compared to the native sequence.

Before PDBench, most publications reported basic metrics such as sequence recovery, recall, and precision [93, 107, 125] using the TS500 benchmark, a set of 500 randomly selected proteins typically excluded from training sets [89].

The TS500 random selection process introduces significant biases. It overrepresents common architectures, such as α-helical orthogonal bundles, while underrepresenting or entirely omitting rare architectures (see Paper Figure 2). As a result, models evaluated on TS500 may perform well on frequently occurring folds but struggle with less common structural classes, leading to misleading generalization estimates. Additionally, designers may require more granular performance analyses, such as the sequence recovery across different folds and resolution levels.

To address these limitations, we developed PDBench, a fold-balanced and manually curated dataset along with a software toolkit for biologically relevant model evaluation. PDBench provides detailed breakdowns and visualisations of model performance across different structural folds and resolution ranges, making it a more informative benchmark. We retrained all the models using PISCES, a protein sequence culling server that provides non-redundant datasets of protein structures [114]. We used a dataset of 35K protein structures (40K+ chains), with resolutions up to 3.0 $\AA$ and used an 80-10-10 training split.

### 3.3.1 The PDBench Dataset

PDBench consists of 595 proteins carefully selected from the PISCES Culled PDB dataset[114], which uses sequence identity filters to avoid fold redundancies and ensure a broad coverage of folds and resolutions. PDBench includes approximately equal representation across 40 distinct protein architectures. These architectures span the four major fold classes: mainly-α, mainly-β, α–β, and special structures [24]. Additionally, we selected structures at resolutions ranging from 1 to 3 Å, to evaluate model robustness to backbone quality.

Crystallographic resolution, measured in angstroms (Å), defines the minimum distance at which two coordinates in an electron density map can be distinguished [118]. Therefore, lower values correspond to higher detail. Structures with resolutions better than 2.0Å are considered high quality, whereas those worse than 3Å often have higher

uncertainties, particularly in side chains and flexible regions [118].

The advent of AlphaFold introduced generated protein structures. Each residue is annotated with a confidence score called pLDDT (predicted Local Distance Difference Test), reflecting local structure reliability, and analogous to resolution [67]. Inevitably, models trained on either crystal structures or predicted structures will learn their respective biases, which can introduce artifacts in sequence-design tasks.

The CATH database classifies protein structures hierarchically into Class (C), Architecture (A), Topology (T), and Homologous superfamily (H). For example, the DNA helicase RuvA subunit (H), is classified as a Mainly α protein (C), that adopts an Orthogonal Bundle architecture (A), characteristic of helicases (T) [37].

CATH shares similarities with other structure classification datasets such as the Pfam [86] and SCOP dataset [29]. Pfam primarily uses sequence alignments to identify conserved functional motifs and domains, whereas SCOP uses structural similarities to organise proteins hierarchically [50]. SCOP is manually curated, while CATH combines structural similarity and manual curation [50].

To accompany the PDBench dataset, we created a software toolkit to automatically calculate and visualise metrics to quickly compare methods.

### 3.3.2 The PDBench Software Toolkit

PDBench includes a model-agnostic software package that evaluates protein design models across several biologically relevant metrics and visualises the results. It requires two input files:

- **Prediction Matrix**: a CSV file of shape 20xN, where each column represents a residue and each row corresponds to the predicted probability of one of the 20 standard amino acids (ordered alphabetically by 1-letter code).
- **Dataset Map**: a text file mapping each column of the prediction matrix to a specific residue in a protein structure, formatted as `<pdbID><chain> <number_of_residues>` (e.g., `1a41A 221`).

Example input files are provided in the GitHub repository for reference.

Previous benchmarks relied on per-residue accuracy, which can be biased toward over-predicting common amino acids. To counter this, we proposed the use of macro-averaged metrics, to account for the natural imbalance in amino acid distribution. For

instance, Macro-recall corrects for class imbalance by ensuring that model performance is not artificially inflated by overpredicting the most common amino acids:

$$\text{Macro-Recall} = \frac{1}{20}\sum_{aa}\text{Recall}(aa)$$

where *aa* represents each of the 20 standard amino acids, and:

$$\text{Recall}(aa) = \frac{\text{True Positives for } aa}{\text{Total actual occurrences of } aa}$$

For example, a trivial model that always predicts Leucine would achieve an average sequence recovery of 9.66% [1]. However, its Macro-Recall would be only 5%, equivalent to random guessing. A key limitation of sequence recovery is that protein designers are often not aiming for 100% recovery, but rather exploring variants with different properties. Nevertheless, before accurate structure prediction tools like AlphaFold became widely available, sequence-based metrics could still be useful for rapidly assessing model performance.

Similarly, we introduced the prediction bias metric to quantify amino acid overprediction. Prediction bias is calculated as the relative difference in amino acid composition between natural sequences and the designed sequences:

$$\text{Prediction Bias} = \sum_{aa}\frac{\left|\text{Freq}_{\text{designed}}(aa) - \text{Freq}_{\text{natural}}(aa)\right|}{\text{Freq}_{\text{natural}}(aa)}$$

where:

- $\text{Freq}_{\text{designed}}(aa)$ is the relative frequency of amino acid *aa* in the designed sequences.
- $\text{Freq}_{\text{natural}}(aa)$ is the relative frequency of amino acid *aa* in the native sequences.

We compute these metrics across various resolutions in our dataset to evaluate model performance at different backbone qualities and across various fold categories.

PDBench also provides visualization tools to aid in model analysis, including:

- **Confusion matrices** to identify amino acids that are frequently mispredicted.
- **Torsion angle frequency plots** to plot the distribution of $\phi$ and $\psi$ angles of correctly-predicted amino acids.

[1] Using the amino acid composition from UniProtKB and Swiss-Prot release April 2013, available at `https://web.expasy.org/protscale/pscale/A.A.Swiss-Prot.html`.

- **3D structural mapping** of sequence accuracy and entropy onto PDB structures.

For additional details and implementation, please refer to the PDBench repository.

## 3.4 Paper Overview and Contribution

Below we present the paper "**PDBench: Evaluating Computational Methods for Protein Sequence Design**" [24]. Using PDBench we evaluated two physics-based methods (EvoEF2, Rosetta fixbb) and four deep-learning methods (ProDCoNN, DenseCPD, DenseNet, ProteinSolver).

Deep-learning-based models generally outperformed physics-based models on mainly-β folds. Furthermore, for mainly-β folds, the sequence recovery correlated more strongly with resolution, suggesting that models may be more sensitive to backbone quality when designing mainly-β sequences.

We also observed that ProteinSolver was trained with structures containing the full amino acid side chains as input, leading to severe label leakage. When we removed the side chains from the benchmark structures, the performance dropped significantly. This implies that the model may only be used for sequence re-design when the sequence is known, and not for *de novo* settings. It is possible that retraining ProteinSolver without side chain leakage on the same dataset could significantly improve its performance, potentially approaching that of ProteinMPNN, given that both are GNN-based models.

We also used PDBench during the development of TIMED (Chapter 4) to identify a Glycine overprediction issue in Mainly α folds. We noticed that increasing the training data effectively addressed this issue. However, it is a common pattern for these deep learning methods to predict a poly-Glycine sequence for low-quality backbones [24].

PDBench was cited in several reviews [3, 38, 46, 49], design benchmarks [126], and novel models [83, 115]. For example, Wang et al. [115] developed a protein sequence design model, SPDesign, and used PDBench to benchmark it against other models.

Here, I would like to acknowledge the work of my co-authors Rokas Petrenas, Kartic Subr, and Christopher W. Wood. Together, we developed the benchmark concept and wrote the paper. Rokas selected the benchmark structures, developed the PDBench software toolkit, and benchmarked ProteinSolver, Rosetta, and EvoEF2. Rokas and I developed appropriate metrics, reimplemented the DenseCPD model, and identified early shortcomings of CNNs related to the Glycine trap. I created the training code for CNN models, reimplemented models, and retrained them on the same dataset.

*Bioinformatics*, 39(1), 2023, btad027
https://doi.org/10.1093/bioinformatics/btad027
Advance Access Publication Date: 13 January 2023
Applications Note



Structural bioinformatics

# PDBench: evaluating computational methods for protein-sequence design

**Leonardo V. Castorina[1,†], Rokas Petrenas[2,†], Kartic Subr[1] and Christopher W. Wood [2,*]**

[1]School of Informatics, University of Edinburgh, 10 Crichton Street, Newington, Edinburgh EH8 9AB, UK and [2]School of Biological Sciences, University of Edinburgh, Roger Land Building, Edinburgh EH9 3FF, UK

*To whom correspondence should be addressed.
†The authors wish it to be known that, in their opinion, the first two authors should be regarded as Joint First Authors.

Associate Editor: Lenore Cowen

Received on August 30, 2022; revised on November 14, 2022; editorial decision on December 15, 2022; accepted on January 12, 2023

## Abstract

**Summary:** Ever increasing amounts of protein structure data, combined with advances in machine learning, have led to the rapid proliferation of methods available for protein-sequence design. In order to utilize a design method effectively, it is important to understand the nuances of its performance and how it varies by design target. Here, we present PDBench, a set of proteins and a number of standard tests for assessing the performance of sequence-design methods. PDBench aims to maximize the structural diversity of the benchmark, compared with previous benchmarking sets, in order to provide useful biological insight into the behaviour of sequence-design methods, which is essential for evaluating their performance and practical utility. We believe that these tools are useful for guiding the development of novel sequence design algorithms and will enable users to choose a method that best suits their design target.

**Availability and implementation:** https://github.com/wells-wood-research/PDBench

**Contact:** chris.wood@ed.ac.uk

**Supplementary information:** Supplementary data are available at *Bioinformatics* online.

## 1 Introduction

The goal of protein design is to create novel amino acid sequences with useful properties and functions. An important part of this process is determining sequences that will fold to a target structure, and this can be thought of as the 'inverse protein folding problem' (Yue and Dill, 1992). To address this challenge, many successful approaches for designing proteins have been developed, but computational protein design (CPD) has quickly become the standard approach (Woolfson, 2021).

Current methods for benchmarking protein design methods focus on sequence recovery, where the backbones of natural proteins with known amino-acid sequences are passed as the input and the accuracy of the method is measured by identity between the predicted sequence and the true sequence (Qi and Zhang, 2020; Strokach *et al.*, 2020; Zhang *et al.*, 2020). However, accuracy values do not capture the real-world utility of a design method. Ultimately, *we must move beyond simplistic methods for evaluating design methodologies* and provide information to users that will help them to assess whether a specific method will be appropriate for their target application.

Here, we describe PDBench, a set of protein structures and associated tools for benchmarking the performance of CPD methods. PDBench generates a rich set of metrics to give a more holistic view of performance.

## 2 Materials and methods

Our benchmark set contains 595 protein structures spanning 40 protein architectures that are clustered into 4-fold classes: mainly-$\alpha$, mainly-$\beta$, $\alpha$–$\beta$ and special, as presented in the CATH database (Knudsen and Wiuf, 2010). Crystal structures with maximum resolution of 3 Å were chosen to cover the structural diversity present in the PDB (see 1). This ensures that the performance is evaluated on high- and low-quality inputs (see Supplementary Fig. S2) and the results are not biased towards the most common protein architectures.

Benchmarking tool: We have developed an open-source benchmarking library, written in Python (https://github.com/wells-wood-research/PDBench). The user supplies PDBench with a prediction matrix (in .csv format) and a dataset map (in .txt format), and it generates metrics for each model in a plot, as well as the option to generate comparison plots between different models to compare their performance. The software is not limited to the benchmarking set we have created, the user can specify any set of structures and

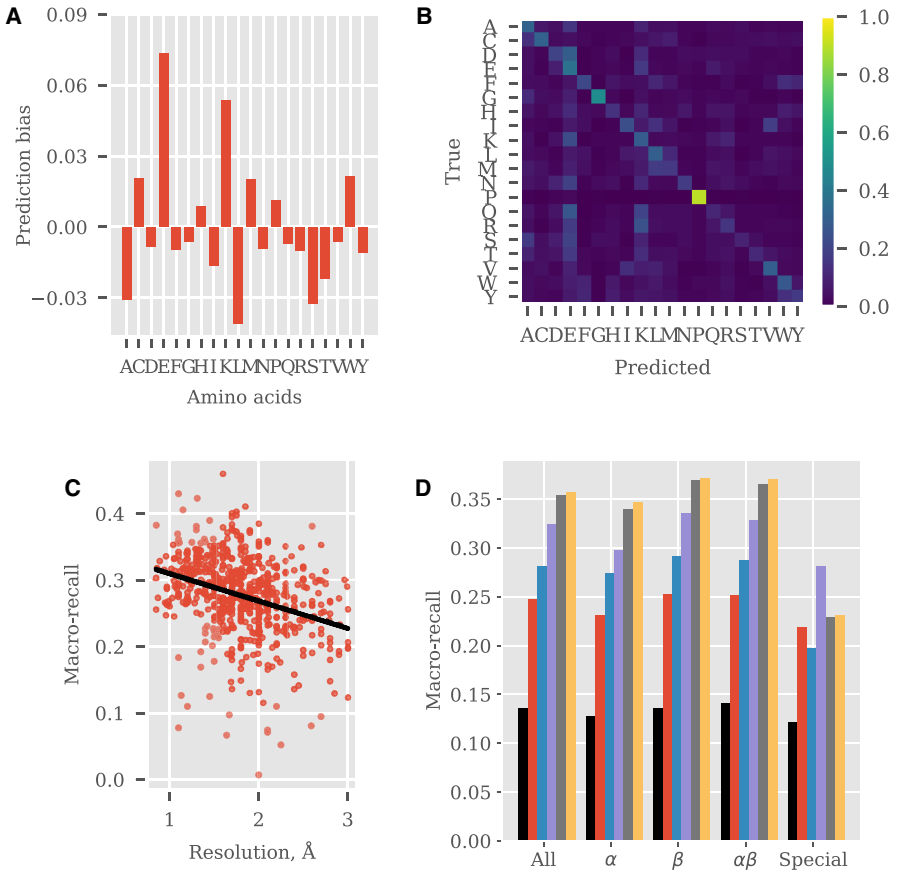



**Fig. 1.** Example plots produced by PDBench. (**A–C**) Selected performance plots for ProDCoNN. (**A**) Prediction bias of predictions relative to abundance of amino acids in the dataset. The model has a negative bias towards common residues such as leucine and alanine, indicating that there is not a bias for the most common class. (**B**) Correlation between performance and resolution of protein structures. (**C**) Prediction confusion matrix between predicted residue and real residue in the sequence. (**D**) Performance comparison plot between all models tested across different types of folds (bars in group from left to right): ProteinSolver (black), EvoEF2 (red), ProDCoNN (blue), Rosetta (purple), DenseCPD (gray) and DenseNet (yellow), see also Supplementary Figures S7 and S8 (A color version of this figure appears in the online version of this article)

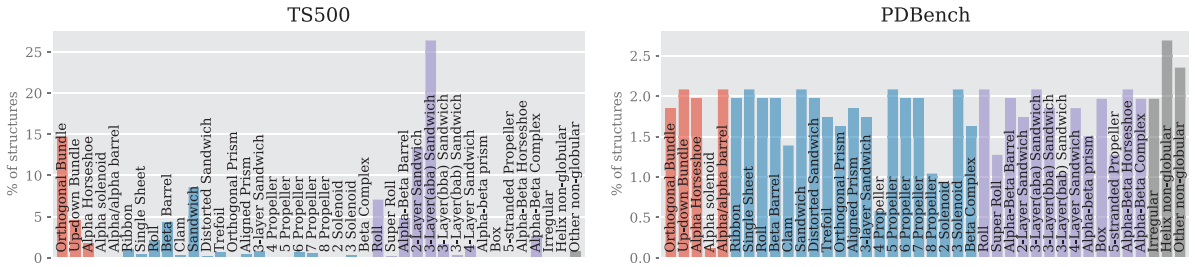



**Fig. 2.** Comparison between the TS500 benchmark (O'Connell *et al.*, 2018), commonly used in the literature, (left) and our fold-balanced benchmark PDBench (right). The PDBench benchmark comprises 40 protein architectures grouped into 4 categories: mainly-$\alpha$ (red)—70 chains, mainly-$\beta$ (blue)—282 chains, $\alpha$–$\beta$ (purple)—196 chains and special (yellow)—47 chains (A color version of this figure appears in the online version of this article)

the AMPAL library will be used to read the protein sequences and optionally replace non-canonical residues with standard amino acids (Wood *et al.*, 2017). The programme DSSP is used to assign the secondary structure for each residue (Joosten *et al.*, 2011). The CATH database (Knudsen and Wiuf, 2010) is used to assign protein architecture. Figure 1 shows a sample output for several models.

Metrics: We calculate four groups of metrics: (1) recall, precision, AUC, F1 score, Shannon's entropy and prediction bias *for each amino acid class*; (2) accuracy, macro-precision, macro-recall, similarity and top-3 accuracy *for each protein chain*; (3) accuracy, macro-precision, macro-recall, similarity and top-3 accuracy *for each secondary structure type* and (4) accuracy, macro-precision, macro-recall, similarity and top-3 accuracy *for each protein architecture*. As shown in Supplementary Figure S2 (right), the numbers of amino acids in proteins are heavily imbalanced, meaning that a model overpredicting the most common amino acid may obtain an accuracy higher than random. Macro-recall is an accuracy score resistant to class imbalance which allows a fairer comparison of models.

Prediction bias is a metric measuring the discrepancy between the occurrence of a residue and the number of times it is predicted (see Fig. 1A and Supplementary Fig. S5). To account for functional redundancy between amino acids, the relative frequency of substitution of amino acids in nature is used to calculate a similarity score (Henikoff and Henikoff, 1992). PDBench also outputs torsion angle

comparison plots between true and predicted residues, if structural models are provided, which is useful to further explore overprediction (see Fig. 1B and Supplementary Fig. S4).

Models evaluated: We tested two state-of-the-art physics-based methods: 'EvoEF2' (Huang *et al.*, 2019b) and 'Rosetta' (Das and Baker, 2008). We also tested deep-learning methods: 'ProDCoNN' (Zhang *et al.*, 2020) (CNN), 'DenseCPD' (Huang *et al.*, 2019a) (CNN), 'DenseNet' (Huang *et al.*, 2019a) (CNN) and 'ProteinSolver' (Strokach *et al.*, 2020) (GNN). Code for ProDCoNN, DenseCPD and DenseNet was not available, so we re-implemented them using Keras and filtering out the benchmark structures from the training set (Supplementary Fig. S6).

## 3 Discussion

As an example, we used PDBench to compare a range of published methods for sequence design (see Supplementary Fig. S3). We divided our benchmark set (595 structures) into four categories of protein folds, each with a balanced proportion of structures for each category as shown in (Fig. 2, right), unlike other benchmarks such as TS500, which is heavily unbalanced (Fig. 2, left). When considering accuracy metrics along with similarity to the target structures (Supplementary Fig. S3), there is a marked difference in the performance of all the design algorithms across the different fold classes. It is interesting that all of the deep-learning-based methods performed well when designing 'mainly $\beta$' structures, as these are challenging design targets (Huang *et al.*, 2016; Woolfson *et al.*, 2015). Furthermore, the accuracy of sequence recovery was more strongly correlated with resolution in the $\beta$-containing classes (Supplementary Fig. S3), suggesting that the sequence preferences in $\beta$ structure are closely linked to subtle details in the backbone conformation. The performance of the ProteinSolver was lower than expected, given the reported performance (Strokach *et al.*, 2020). We believe that this is due to leakage of information regarding side chains identities, if they are provided in the input model (Supplementary Section S4). As a result, while the method might be suitable for protein engineering, it is ill suited to *de novo* design where only backbone atoms are provided as an input.

While sequence recovery is an important metric in understanding the performance of a sequence design method, it is not sufficient to fully understand its properties. Furthermore, a static, single structure view of a protein is not representative of the behaviour of a protein in solution. The ultimate test is to produce design in the lab, but further computational analysis of the models can also generate useful information on designs (Goldenzweig *et al.*, 2016; Ludwiczak *et al.*, 2018; Ollikainen and Kortemme, 2013; Stam and Wood, 2021).

Our design-method agnostic benchmark and tools aim to shed light on the behaviour of CPD algorithms. We believe that this information will be of use to developers of CPD algorithms, especially when combined with modern methods of structure prediction (Chowdhury *et al.*, 2021; Jumper *et al.*, 2021; Rives *et al.*, 2019; Wu *et al.*, 2022). It also provides users of these methods crucial information regarding the appropriateness of the design method to their application.

## Funding

This work was supported by the Wellcome Trust-University of Edinburgh Institutional Strategic Support Fund [ISSF3]. C.W.W. is supported by an Engineering and Physical Sciences Research Council (EPSRC) Fellowship [EP/S003002/1]. L.V.C. is supported by the UK Research and Innovation (UKRI) Centre for Doctoral Training in Biomedical AI at the University of Edinburgh [EP/S02431X/1]. K.S. is supported by a Royal Society University Research Fellowship.

*Conflict of Interest*: none declared.

## Data availability

The data underlying this article are available in through the Protein Data Bank (PDB), which can be accessed here: http://www.wwpdb.org. Source code is available on GitHub: https://github.com/wells-wood-research/PDBench.

## References

Chowdhury,R. *et al.* (2021) Single-sequence protein structure prediction using language models from deep learning. *bioRxiv*.

Das,R. and Baker,D. (2008) Macromolecular modeling with rosetta. *Annu. Rev. Biochem.*, **77**, 363–382.

Goldenzweig,A. *et al.* (2016) Automated structure- and sequence-based design of proteins for high bacterial expression and stability. *Mol. Cell*, **63**, 337–346.

Henikoff,S. and Henikoff,J.G. (1992) Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. USA*, **89**, 10915–10919.

Huang,G. *et al.* (2019a) Convolutional networks with dense connectivity. *IEEE Trans. Pattern Anal. Mach. Intell.*, **44**(12), 8704–8716.

Huang,P.-S. *et al.* (2016) The coming of age of de novo protein design. *Nature*, **537**, 320–327.

Huang,X. *et al.* (2019b) EvoEF2: Accurate and fast energy function for computational protein design. *Bioinformatics*, **36**, 1135–1142.

Joosten,R.P. *et al.* (2011) A series of PDB related databases for everyday needs. *Nucleic Acids Res.*, **39**, D411–D419.

Jumper,J. *et al.* (2021) Highly accurate protein structure prediction with AlphaFold. *Nature*, **596**, 1–11.

Knudsen,M. and Wiuf,C. (2010) The CATH database. *Hum. Genomics*, **4**, 207–212.

Ludwiczak,J. *et al.* (2018) Combining Rosetta with molecular dynamics (MD): A benchmark of the MD-based ensemble protein design. *J. Struct. Biol.*, **203**, 54–61.

O'Connell,J. *et al.* (2018) Spin2: Predicting sequence profiles from protein structures using deep neural networks. *Proteins*, **86**, 629–633.

Ollikainen,N. and Kortemme,T. (2013) Computational protein design quantifies structural constraints on amino acid covariation. *PLoS Comput. Biol.*, **9**, e1003313.

Qi,Y. and Zhang,J.Z.H. (2020) DenseCPD: Improving the accuracy of neural-network-based computational protein sequence design with DenseNet. *J. Chem. Inf. Model.*, **60**, 1245–1252.

Rives,A. *et al.* (2019) Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *bioRxiv*.

Stam,M.J. and Wood,C.W. (2021) DE-STRESS: A user-friendly web application for the evaluation of protein designs. *Protein Eng. Des. Sel.*, **34**.

Strokach,A. *et al.* (2020) Fast and flexible protein design using deep graph neural networks. *Cell Syst.*, **11**, 402–411.e4.

Wood,C.W. *et al.* (2017) ISAMBARD: An open-source computational environment for biomolecular analysis, modelling and design. *Bioinformatics*, **33**, 3043–3050.

Woolfson,D.N. (2021) A brief history of de novo protein design: Minimal, rational, and computational. *J. Mol. Biol.*, **433**, 167160.

Woolfson,D.N. *et al.* (2015) De novo protein design: How do we expand into the universe of possible protein structures? *Curr. Opin. Struct. Biol.*, **33**, 16–26.

Wu,R. *et al.* (2022) High-resolution *de novo* structure prediction from primary sequence. *bioRxiv*.

Yue,K. and Dill,K.A. (1992) Inverse protein folding problem: Designing polymer sequences. *Proc. Natl. Acad. Sci. USA*, **89**, 4163–4167.

Zhang,Y. *et al.* (2020) ProDCoNN: Protein design using a convolutional neural network. *Proteins*, **88**, 819–829.

# Chapter 4

# TIMED

## 4.1 Introduction

Computational protein design has enabled breakthroughs in fields ranging from *de novo* enzyme design [13, 84, 88, 116, 123], binder design [20, 90], and therapeutics[21, 27, 45, 53, 102]. More recently, deep learning methods have shown great promise, achieving similar or better performance than traditional physics-based methods at a fraction of the computational cost.

Shortly after PDBench was developed, a major breakthrough in protein folding occurred with the release of fast and accurate protein folding methods, such as AlphaFold 2 [67], and later OmegaFold[122] and ESMFold[77]. These models enabled rapid *in silico* validation of protein designs, improving benchmarking.

Despite their potential, deep learning models faced significant practical limitations. Models like ProDCoNN and DenseCPD were closed-source and lacked protein voxelisation code. Additionally, their absence of user-friendly UIs further limited adoption.

Even when accessible, models like ProteinMPNN showed biases towards specific amino acids, such as Lysines and Glutamates, leading to undesirable sequence distributions[1]. Moreover, these models had no simple way to incorporate biochemical constraints such as charge or polarity.

To improve accessibility, some researchers turned to Google Colab notebooks for AlphaFold and ProteinMPNN. However, these workarounds required Google accounts and frequent interaction with the window to prevent disconnection. ProteinSolver initially provided a UI but was later discontinued.

In this chapter, we address these challenges with TIMED (Three-dimensional

[1] See GitHub Issue

Inference Method for Efficient Design), our family of CNNs for protein sequence design [25]. We present four key contributions:

1. Developing Aposteriori, an open-source protein voxelisation library.
2. Designing and validating a new CNN architecture.
3. Implementing a way to incorporate design constraints for voxel-based models.
4. Building TIMED-Design, an intuitive UI and Command-Line Interface (CLI) to democratise access to protein sequence design tools.

This chapter begins by describing the methodological foundations of TIMED, including the development of Aposteriori, the TIMED architecture, and our training approach. We then present benchmarking results comparing TIMED with existing sequence design methods and demonstrate the impact of incorporating design constraints. Finally, we discuss the UI and the implications of this work for the protein design field.

## 4.2 Methodological Foundations of TIMED

### 4.2.1 Open-sourcing Voxels with Aposteriori

CNN models for protein sequence design rely on transforming 3D protein structures into voxel grids called *frames*, centred around each amino acid. Each frame is independently processed by the CNN to predict a probability distribution across the twenty amino acids (see Paper Figure 1 a-b).

Voxelisation of proteins has been used in a range of structural biology tasks. HTMD [40] uses voxel grids to encode spatial environments for deep learning, while Metal3D [41] applies 3D CNNs to predict metal binding sites from voxelised protein pockets. Outside protein design, Gaussian-based voxelisation has long been standard in cryo-EM, as implemented in the Situs toolkit [98].

The Aposteriori library is tailored for sequence design: it builds residue-centred frames, supports design-specific features like charge and polarity, and integrates directly with our CNN pipelines. Zhang et al. [125] introduced several effective approaches for standardising voxelised inputs for neural networks in the ProDCoNN paper. First, they proposed using Gaussian distributions, rather than binary voxels, to represent atoms. This approach uses the Van der Waals' radius of the backbone atom being encoded (C, N, O) to determine the density based on the atom's position. For example, an atom with

x-coordinates of 1.5, 1.6, and 1.7 would be represented as $[0,1,0]$ in binary voxelisation. However, Gaussian voxelisation assigns densities instead: $[0.16,0.68,0.16]$ for $x = 1.5$, $[0.11,0.68,0.21]$ for $x = 1.6$, and $[0.08,0.65,0.27]$ for $x = 1.7$ (see Figure 4.1). While this significantly improves performance, it increases the computational costs by storing floating-point values rather than binary values.

Second, they standardised frame orientation by positioning the Cα at the centre and aligning the N-Cα-C plane to the XY plane.

Third, they added a virtual Cβ atom for all amino acids at the average position where the side chain would point.

However, these methods were not open-source. To address this gap, we created Aposteriori, an open-source Python library for protein structure voxelisation. Aposteriori creates 3D frames for each backbone atom (C, N, O, Cα, Cβ). Additionally, we introduced the ability to encode biochemical properties such as charge and polarity as separate channels at the Cα coordinates.

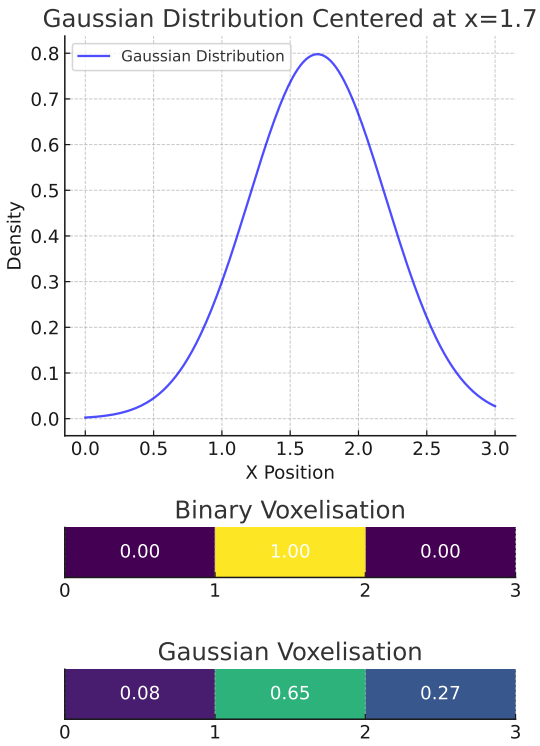



Figure 4.1: Comparison of binary voxelisation and Gaussian voxelisation. The top plot shows a Gaussian distribution centred at $x = 1.7$. The middle plot represents the binary voxelisation, where an atom is assigned discretely to a voxel. The bottom plot represents Gaussian voxelisation, where density is assigned based on a normal distribution.

### 4.2.2 TIMED Neural Network Architecture and Design Principles

We designed the TIMED neural architecture to be simple and computationally lightweight. The architecture consists of several components (see Appendix Figure A.1):

A series of 3D Convolutional Blocks, each followed by Exponential Linear Unit (ELU) activation functions and Batch Normalisation. These blocks process the spatial information in the voxel frames and extract relevant features for amino acid prediction.

We used Max Pooling layers to reduce the dimensionality while preserving important spatial information. Instead of standard Dropout, we used Spatial Dropout, which removes entire feature maps instead of individual neurons, helping to maintain spatial relationships [110].

Finally, we used Global Average Pooling instead of Fully Connected layers at the end of the network, reducing the number of parameters for faster inference while preserving the relationship between features and labels [76, 108].

We used Weights & Biases to tune the hyperparameters of the architecture [16].

#### 4.2.2.1 Training Strategy and Dataset Preparation

We trained TIMED and reimplemented versions of ProDCoNN, DenseCPD, DenseNet, and ProteinMPNN using the PISCES dataset described in the previous chapter. This approach avoided introducing bias our models toward common protein folds [114].

For Aposteriori voxelisation, we used frames of size 12 $\mathring{A}$ and 21 voxels per side. This resulted in a voxel corner-to-corner distance of approximately 1 $\mathring{A}$, preventing clashes attempting to encode two atoms in the same voxel.

To address the prediction bias toward abundant amino acids noted in Chapter 3, we implemented a balancing strategy using random undersampling during training. At each epoch, we presented the model with an equal number of frames for each amino acid type, matching the count to that of the least abundant amino acid (Cysteine). For more abundant amino acids, we randomly sampled frames while maintaining the total count fixed across all amino acid types. This approach significantly reduced prediction bias while allowing the model to see more instances of each amino acid during training.

### 4.2.3 Voxelising Design Constraints

Protein designers often approach problems with specific constraints, such as preferred chemical properties in specific regions like surfaces. Charge and polarity at solvent-exposed surfaces influence binding specificity and affinity, making them critical when

designing interfaces, for example in docking or antibody-antigen interactions [75, 91]. These constraints help create or maintain favourable surface energetics for protein-protein interactions, while also simplifying the design problem by reducing the search space. By encoding these properties directly into the voxel input, TIMED can be steered towards solutions that better match the desired surface chemistry.

To support this, we extended Aposteriori to allow the integration of biochemical constraints, directly into the voxel representation. We encode these as an additional channel at the Cα position of each frame. Using this approach, we developed two flavours of TIMED: TIMED-Charge and TIMED-Polar.

TIMED-Charge encodes the amino acid charge at pH 7. We use negative or positive Gaussian density values at the Cα position depending on residue charge, while neutral residues are left unmarked. This binary scheme simplifies the representation while preserving key electrostatic features. We reasoned that explicitly including charge information would help the model learn spatial charge distributions and electrostatic complementarity, which are critical for folding, solubility, and interactions. For example, positioning similarly-charged residues can stabilise surfaces by promoting favourable interactions with surrounding water molecules [54, 127]. Similarly, charge patches are commonly found at protein–protein interfaces, where they help drive binding through electrostatic complementarity [112]. By encoding this information in a separate input channel, TIMED-Charge can learn to use these features directly, without needing to implicitly learn them from backbone voxel representations.

Similarly, TIMED-Polar encodes polarity using data from Zimmerman et al. [129]. As with charge, we use positive and negative values to encode polar and non-polar residues, respectively. To focus on highly polar residues with strong solvation and electrostatic contributions, we classify a residue as polar if its Zimmerman score is $\geq 20$. This threshold captures charged and strongly polar side chains, such as D, E, H, K, and R, while excluding weakly polar residues like S, T, or Q that may not significantly affect surface energetics in isolation [92]. Similarly to TIMED-Charge, encoding polarity independently should allow the model to learn polarity-mediated effects directly, rather than relying on backbone context alone.

Both variants maintain the same underlying neural network architecture as the base TIMED model, with an additional input channel for the property. While these models require explicit property information that may not be available in *de novo* design settings, they are perfect tools for protein redesign and for refining designed sequences. Through the UI, users can also modify the charge or polarity at specific positions to

design or modify surface potentials to better fit with the design target.

We benchmarked all the models using the PDBench benchmark. As protein sequence design models output sequences given 3D shapes, we feed the predicted sequences to AlphaFold 2 to obtain the predicted 3D shape [67]. This validated whether the sequences would fold to that shape. To run AlphaFold 2, we used LocalFold, a local version of ColabFold [85]. We then use PyMol CEAligner[100] to align the original and the predicted structure and measure the Root Mean Square Deviation (RMSD), the root of the average distance between the backbones of crystal and predicted structure:

$$\text{RMSD} = \sqrt{\frac{\sum_{j=1}^{n} (y_i - \hat{y}_i)^2}{n}}$$

We normalised it by accounting for protein size using $\text{RMSD}_{100}$ proposed by Carugo and Pongor [22]:

$$\text{RMSD}_{100} = \frac{\text{RMSD}}{1 + \ln \sqrt{\frac{n}{100}}}$$

where $n$ is the number of amino acids.

CASP evaluations often rely on metrics such as GDT or TM-score to assess global fold similarity [5]. However, we chose to use RMSD as it is a commonly used metric by biologists and used in popular packages like PyMOL[100] and BioPython[31]. On the other hand, we found GDT and TM-Score to be relatively difficult to install.

At the time this research was done, AlphaFold 2 relied on an external MSA server. As AlphaFold gained popularity, obtaining the MSA for a protein required waiting in a queue, which was then followed by high GPU demands for folding. Therefore, to reduce the computational demands, we used 10% of the benchmark covering Mainly $\alpha$, Mainly $\beta$, Mainly, and $\alpha$-$\beta$ folds. We also chose to avoid the Special fold class, as AlphaFold predictions are usually of low confidence around these regions.

### 4.2.4 Democratising Protein Sequence Design

We believe that protein design should be accessible to everyone. Existing tools impose technical barriers, requiring users to create Python environments and manage dependencies before use. To address this challenge, we developed TIMED-Design, a UI and CLI software package, that provides access to state-of-the-art protein sequence design methods, even for researchers without computational expertise.

TIMED-Design also supports Monte Carlo sampling from the model's output distributions to generate diverse sequence variants. A temperature parameter controls how exploratory the sampling is: low temperatures favour confident, high-probability predictions (closer to argmax), while higher temperatures increase randomness by flattening the distribution. As shown in Paper Figure 5, higher temperatures lead to more diverse sequences but also higher RMSD values. This trade-off lets users tune between design fidelity and sequence diversity, depending on the goal.

#### 4.2.4.1 Web UI

TIMED-Design includes a responsive web-based interface built with Streamlit, accessible here. Users can select a protein structure from the PDB by entering a PDB code or upload their own structure file. The application automatically processes the structure using Aposteriori to generate voxelised frames and predicts sequences using the selected models (TIMED, TIMED-Charge, TIMED-Polar, ProDCoNN, DenseCPD, or DenseNet).

The interface displays prediction results through:

- Interactive 3D visualization of the protein structure
- Sequence metrics including charge, isoelectric point, and molecular weight
- Probability distribution visualisation for each position
- Performance plots including precision/recall and prediction bias

The UI is privacy-focused and files are regularly deleted between sessions. It also ships with a Docker container to allow local deployment.

For TIMED-Charge and TIMED-Polar, users can specify custom constraints for regions of the protein, enabling targeted redesign of functional sites or interfaces. The application also supports Monte Carlo sampling at different temperature factors to generate diverse sequence variants for experimental screening.

#### 4.2.4.2 CLI

In addition to the web application, TIMED-Design includes a CLI that enables batch processing and integration with existing computational workflows. The CLI offers the same functionality as the web application, making it suitable for high-throughput applications where many structures need to be processed.

By providing both a user-friendly web UI and a flexible CLI, we hope that TIMED-Design will bridge the gap between computational tool developers and experimentalists, and potentially accelerating the pace of protein design.

## 4.3 Performance Evaluation and Benchmarking

### 4.3.1 Comparative Analysis with the State-of-the-Art

To ensure a fair comparison, we evaluated TIMED using the PDBench benchmark described in the previous chapter, alongside physics-based methods (EVOEF2 and Rosetta) and deep learning methods (ProDCoNN, DenseCPD, DenseNet, and ProteinMPNN).

ProteinMPNN was released with CLI, contributing to its popularity since most other models were closed-source [35]. The original ProteinMPNN model was trained on 500K+ chains, including most of the structures in PDBench. To ensure a fair comparison, we retrained all the deep learning models using the same PISCES dataset, isolating architectural differences from training data advantages.

Figure 2a of the paper shows the Macro-Recall performance across models. TIMED's performance is comparable to DenseNet but slightly lower than that of DenseCPD. Notably, TIMED, DenseNet, and DenseCPD matched or outperformed ProteinMPNN across all fold types, while physics-based methods consistently showed lower Macro-Recall.

We found a significant correlation between Macro-Recall and resolution (Paper Figure 4). Mainly $\alpha$ folds showed the highest correlation with resolution, while $\alpha$-$\beta$ folds showed the lowest. For Mainly $\alpha$, ProteinMPNN and ProDCoNN were the most sensitive to resolution, while physics models and TIMED were the least sensitive. For Mainly $\beta$, TIMED and Rosetta were most sensitive, while EvoEF2 and TIMED-Charge were least. For $\alpha$-$\beta$ folds, all models showed low sensitivity to resolution, with TIMED and its flavours being the least sensitive. This information can guide designers in selecting the optimal model based on fold type and backbone quality.

Additionally, we found that model performance varied significantly with local packing density. Regions with higher packing density (typically protein cores) showed better prediction accuracy, lower RMSD, and lower prediction entropy. This correlation was particularly strong for Mainly $\alpha$ and $\alpha$-$\beta$ folds compared to Mainly $\beta$ folds.

Finally, as shown in Appendix Figure A.2, we found that models trained with balancing showed lower prediction bias than when trained without. For instance, when

trained with balancing TIMED and ProDCoNN showed a decrease in accuracy by more than 6% with similar Macro-Recall. However, the prediction bias decreased significantly, especially for Alanine, Glutamate and Leucine in Mainly α and for Leucine and Valine for Mainly β folds. After implementing balancing with random undersampling, the prediction bias for all amino acids approached 0%, meaning that the amino acid distribution of predicted sequences closely matched the natural distribution.

### 4.3.2 Structure Validation with AlphaFold 2

While sequence recovery metrics provide useful evaluations, the ultimate goal of protein sequence design is to produce sequences that fold into the desired structure. To evaluate this aspect, we used AlphaFold 2 [67] to predict the 3D structures of sequences generated by each model and compared them with the original crystal structures.

Figure 2b of the paper compares the $RMSD_{100}$ across models. EvoEF2 showed the largest range of RMSD values, especially for Mainly α folds. In contrast, all other models consistently achieved RMSD values below 1.5 $\AA$, especially for Mainly β folds and α-β. Interestingly, ProteinMPNN performed similarly to other models, even at a lower sequence Macro-Recall.

The strong structural recovery across all deep learning models indicates that they may generate sequences likely to fold to structures similar to the input backbone.

### 4.3.3 Effect of Design Constraints on Model Performance

To evaluate the impact of the charge and polarity design constraints, we compared the base TIMED model with its flavours, TIMED-Polar and TIMED-Charge.

As shown in Figure 3a of the paper, both TIMED-Polar and TIMED-Charge outperformed the base TIMED model in Macro-Recall across all fold types. TIMED-Charge achieves a significantly better performance across all metrics. TIMED-Polar generally outperformed TIMED in most metrics, except for RMSD of Mainly α folds, where TIMED achieves a smaller range of RMSD.

While most TIMED and TIMED-Charge designs had similar RMSD values (Paper Figure 3.4b and e), TIMED-Charge showed significantly better property preservation. It achieved lower Mean Absolute Error (MAE) values for charge (Paper Figure 3c) and isoelectric point (Paper Figure 3d). This suggests that when the properties—but not the amino acids—are known in advance, TIMED-Charge integrates these constraints more effectively into designed sequences.

### 4.3.4 Experimental Validation at the Adaptyv Bio Competition

Since its publication, we successfully used TIMED and its flavours in the Adaptyv Bio competition to design a binder against EGFR[2]. We opted for sequence redesign of the EGF and submitted 10 TIMED designs with less than 50% sequence identity to EGF. All had high expression, and one bound EGFR with a $\mu$M Dissociation Constant (Kd) (shown in Figure 4.2.

We began by analysing Round 1 designs using NetSolP [109] and DE-STRESS [104]. This revealed that lower Aggrescan3D aggregation propensity [72], higher NetSolP usability, and smaller design size were predictive of expression. These insights informed our Round 2 design strategy.

We redesigned a known EGFR binder, selecting EGF due to its small size, using the EGF–EGFR co-crystal structure (PDB: 1IVO) as the design backbone. Sequences were generated using vanilla TIMED, TIMED-Charge (including charge constraints), Co-TIMED (including EGFR context), DenseCPD, and DenseNet [93]. We sampled 20,000 sequences per model (100,000 total) using Monte Carlo sampling at T=0.1. NetSolP was then used to rank the sequences by usability.

We used two strategies to select designs: (1) selecting the top 200 NetSolP-ranked sequences and creating consensus sequences from the top 10 and top quartile; and (2) sampling 20 sequences from the top 1% of NetSolP scores. These sequences were folded with ColabFold [85] and assessed using DE-STRESS, competition metrics (ipTM, iPAE, ESM log-likelihood)[3], and BUDE force field interaction energies [82].

We selected the final 10 submissions based on a combination of these metrics. Eight came from strategy (1), while two were consensus sequences. All 10 designs were further validated using drMD simulations [103] to confirm structural stability and folding.

When analysing the results of the Adaptyv Bio competition, we observed a clear correlation between binding affinity (Kd) and both sequence and structural similarity to known proteins [33]. This suggests that designs closer to natural sequences or folds are more likely to bind effectively. While some of this may reflect human bias, i.e. contestants favouring known shapes, it also raises a broader question: are current methods truly performing *de novo* design, or are they effectively recalling and recombining patterns seen during training?

[2]See Adaptyv Bio Competition Round 2

[3]`https://github.com/adaptyvbio/competition_metrics`

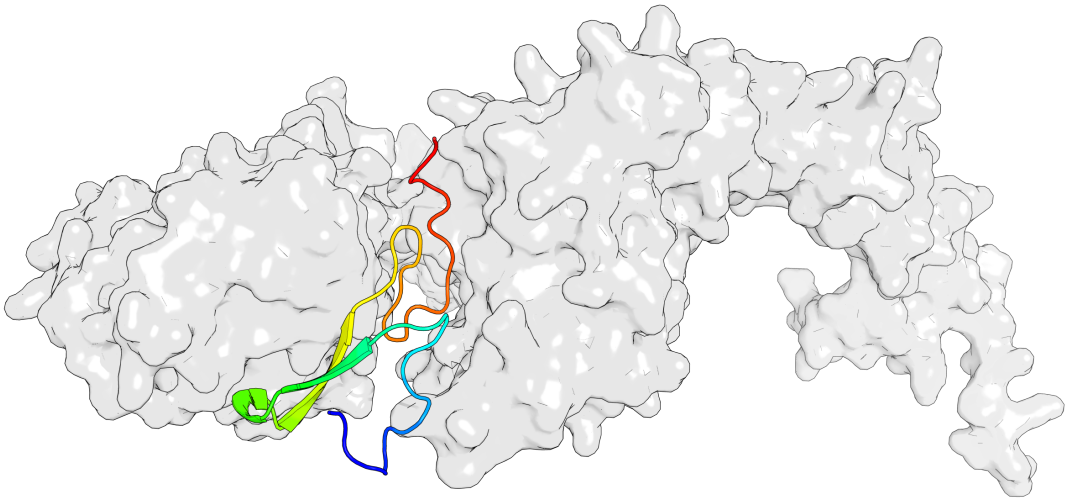


Figure 4.2: Designed EGF (Rainbow) by TIMED in complex EGFR (White). This particular design was submitted at the Adaptyv Bio binder competition and obtained a $\mu M$ Kd towards EGFR at less than 47% sequence identity.

## 4.4 Paper Overview and Contribution

Below we present the paper: "**TIMED-Design: flexible and accessible protein sequence design with convolutional neural networks**" [25]. Our work addressed key limitations in the field through four main contributions: an open-source voxelization library, a novel CNN architecture, integration of biochemical constraints, and accessible user interfaces.

Using the same training data, we showed that most CNNs models matched or outperformed ProteinMPNN, in both Macro-Recall and $RMSD_{100}$. We showed fold-specific relationships between Macro-Recall and backbone resolution, with Mainly α showing higher sensitivity to resolution than α-β folds. This implies that designing Mainly α proteins may require higher-quality backbones.

The strong correlations between performance (RMSD, Accuracy, Entropy) and Mean Packing Density suggests that areas around hydrophobic cores might be easier to design. This was particularly true for Mainly α and α-β folds.

TIMED-Charge demonstrated that incorporating biochemical constraints significantly improves property preservation without sacrificing structural accuracy. This is particularly valuable for applications requiring specific charge distributions, such as designing protein-protein interfaces or binding sites.

TIMED-Design democratises access to powerful protein design tools that were previously unavailable to researchers. By providing both web and command-line interfaces, we enable researchers across technical skill levels and without computational resources to use state-of-the-art sequence design methods. This gives researchers the opportunity to test different models and use ensembling techniques to improve the sequences.

The author contributions are outlined in the "Author Contributions Statement" section of the paper.

*Protein Engineering, Design and Selection*, 2024, **37**, gzae002
https://doi.org/10.1093/protein/gzae002
Advance access publication date 30 January 2024
**Original Article**

# TIMED-Design: flexible and accessible protein sequence design with convolutional neural networks

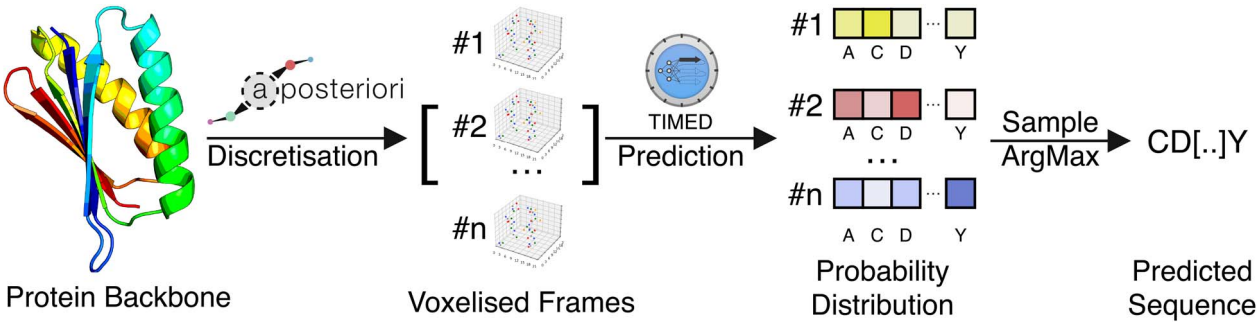
**Leonardo V. Castorina[1], Suleyman Mert Ünal[2], Kartic Subr[1] and Christopher W. Wood[2,*]**

[1]School of Informatics, University of Edinburgh, 10 Crichton Street, Edinburgh EH8 9AB United Kingdom
[2]School of Biological Sciences, University of Edinburgh, Roger Land Building, Edinburgh EH9 3FF, United Kingdom
*To whom correspondence should be addressed. Email: chris.wood@ed.ac.uk

## Abstract

Sequence design is a crucial step in the process of designing or engineering proteins. Traditionally, physics-based methods have been used to solve for optimal sequences, with the main disadvantages being that they are computationally intensive for the end user. Deep learning-based methods offer an attractive alternative, outperforming physics-based methods at a significantly lower computational cost. In this paper, we explore the application of Convolutional Neural Networks (CNNs) for sequence design. We describe the development and benchmarking of a range of networks, as well as reimplementations of previously described CNNs. We demonstrate the flexibility of representing proteins in a three-dimensional voxel grid by encoding additional design constraints into the input data. Finally, we describe TIMED-Design, a web application and command line tool for exploring and applying the models described in this paper. The user interface will be available at the URL: https://pragmaticproteindesign.bio.ed.ac.uk/timed. The source code for TIMED-Design is available at https://github.com/wells-wood-research/timed-design.

## Graphical Abstract



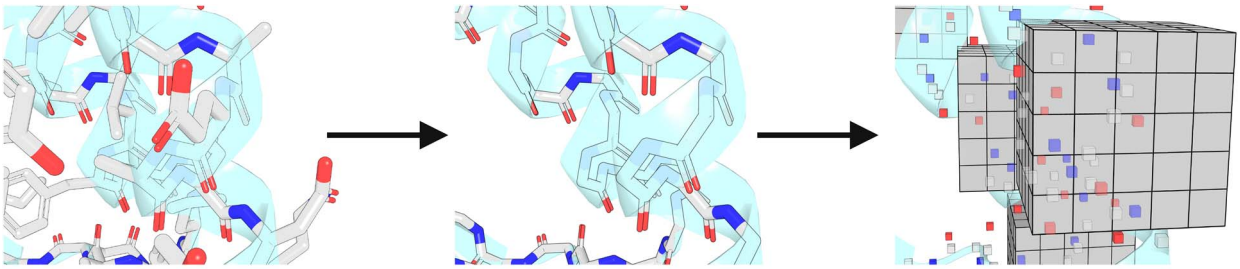

**Keywords:** protein sequence design, Convolutional Neural Networks (CNNs), user interface (UI), AlphaFold 2

## Introduction

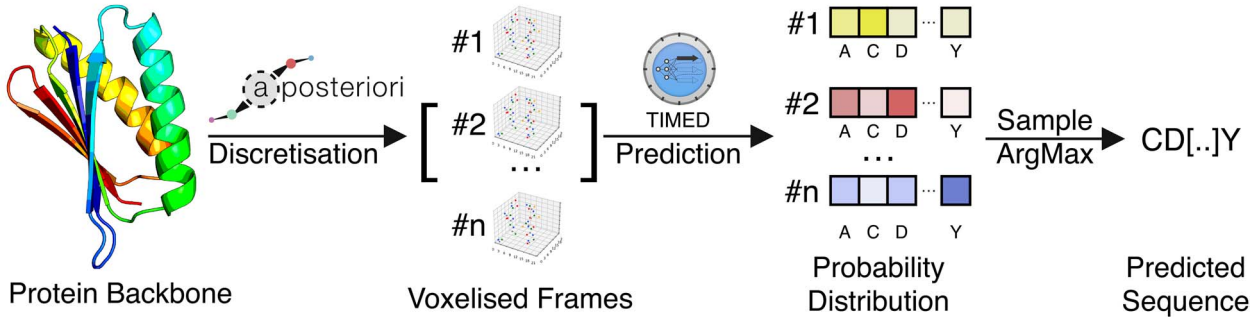
Protein design is a rapidly maturing field with an ever-increasing number of examples of designed proteins being produced (Woolfson, 2021). Excitingly, the field is moving beyond designing structures towards creating functional proteins (Pan & Kortemme, 2021), fulfilling a promise that has been repeated by protein designers for decades. Most protein design algorithms can be broken down into two phases: backbone design and sequence design. There are several approaches to backbone design, including fragment-based methods (Ferruz *et al.*, 2021, Krivacic *et al.*, 2022; Zhou *et al.*, 2020), parametric methods (Wood *et al.*, 2017; Yang *et al.*, 2021) and, more recently, deep learning (DL)-based methods (Watson *et al.*, 2023). Once backbone models have been generated, sequences must be selected that will fold to the target structure.

There are many approaches to sequence design too, including consensus design (Porebski & Buckle, 2016), Monte Carlo based sampling, such as the method employed by Rosetta (Leman *et al.*, 2020), and DL-based methods. DL-based methods offer several potential advantages over other methods: (1) while training DL models is computationally expensive, their application is usually much cheaper. This shifts the computational burden from the end user to the method developer, which improves accessibility of the method. (2) Given a rich enough data set, these methods can learn complex relationships that are present in the training data set without having to explicitly define these. For example, it is likely that the method will be biased by the training set to produce sequences that are more likely to be compatible with cellular environments. (3) As more training data become available, the performance of DL-based models increases without any change to the model architecture (Prapas *et al.*, 2021).

**Received:** August 1, 2023. **Revised:** December 12, 2023. **Accepted:** January 12, 2024



(a) Aposteriori Voxelisation Pipeline Zoomed on Frame




(b) Overview of the prediction pipeline using TIMED for a protein backbone of length n.

**Fig. 1.** (**a**) Voxelisation pipeline from structures to frames. Protein sidechain atoms are removed from the structures to leave just the backbone atoms. Then, the backbone atoms are discretised into voxels. Finally, we extract a fixed cube of space around each amino acid that we call a 'frame'. The frames contain atom voxels with the C$\alpha$ in its centre. (**b**) Aposteriori generates one frame for each amino acid in the backbone. For each frame, TIMED produces a probability distribution over the 20 amino acids. The output sequence can be obtained by either sampling from this probability distribution or by selecting the amino acid with the highest probability using the ArgMax function.

A range of neural network architectures have been explored for sequence design, including Convolutional Neural Networks (CNNs) (Anand *et al.*, 2022; Qi & Zhang, 2020; Zhang *et al.*, 2020), message-passing Graph Neural Networks (GNNs) (Dauparas *et al.*, 2022a) and Large Language Models (Ferruz *et al.*, 2022; Nijkamp *et al.*, 2022). CNNs have a range of useful properties that make them well suited to sequence design. CNNs are adept at learning spatial relationships and have been applied to many problems involving images (Deng *et al.*, 2009; Lin *et al.*, 2014). CNNs can be generalised beyond two dimensions, and can be applied to 3D data, where 3D voxels' replace 2D pixels, enabling them to be applied to protein structure data.

In order to perform sequence design with CNNs, the protein structure must be discretised into voxels (Fig. 1a), with each voxel containing information regarding its content, which is usually an identifier for an atom element or type. Regions around a particular residue are used as the input to the network, and a probability distribution for the identity of the amino acid is produced in the output. Training data sets can be generated using experimentally determined protein structures, with the aim of recovering the native sequence.

In this paper, we explore the application of CNNs to protein sequence design. We describe the development of TIMED, a CNN-based sequence design algorithm, as well as the reimplementation of a range of other CNN-based methods from the literature, that were not previously available. We explore the flexibility of the CNNs by encoding additional design constraints into the voxel data, enabling the designer to tune the properties of their designs. Finally, we described TIMED-Design, a web application (https://pragmaticprotein design.bio.ed.ac.uk/timed) that enables the use of all of the CNN-based methods described in this paper. The source code and models described in this paper are open source and available on GitHub (https://github.com/wells-wood-research/timed-design).

## Methods

### Dataset generation

To begin training CNN models, we generated voxelised structures of proteins. To facilitate this process, we have developed an open-source Python library called Aposteriori (https://github.com/wells-wood-research/aposteriori), which offers features such as multi-processing, compression, and various types of atom encodings. For every amino acid in the input structure(s), we define a cubic region of space around it, with an edge length equal to `-frame_edge_length` (default to 12 Å). This region is then mapped to discrete space with a specified number of voxels per edge, denoted by `-voxels-per-side`, (default to 21 voxels). To ensure consistency, we rotate the protein structure so that the C$\alpha$ atom of the input amino acid positioned at the center of the frame with the N–C$\alpha$–C plane lying on the XY plane of the voxel grid. We create a frame for each residue in the protein sequence. Side chain information is removed; the C, N, O, C$\alpha$ atom and a virtual C$\beta$ position are voxelised within the frame, and are represented by a Gaussian function whose size depends on the van der Waals radius of the atom, as described in Zhang *et al.* (2020). The library generates a `.hdf5` object, which can be manipulated as a Python dictionary. Additional technical details can be found in the code repository associated with this publication.

As shown in Fig. 1(b), CNN models are trained to classify frames, predicting probabilities for the twenty amino acid classes. Although frames have overlapping information, each frame is predicted independently of each other. Thus, for a protein of $n$ amino acids, the output is an array of shape ($n$, 20), containing the probability distributions for the identity of each amino acid. The probability distribution from CNN models allows us to explore the design space by weighted random sampling to generate new sequences. These sequences can then be folded using methods like AlphaFold (Jumper *et al.*, 2021) and screened using methods such as DE-STRESS (Stam & Wood, 2021).

## Training models with undersampling

The natural frequency of amino acid is not uniformly distributed, with some amino acids being more common than others. We use a random under sampling method to prevent CNN models from learning this frequency bias. Specifically, we cap the number of frames for each amino acid to match the count of cysteine, the least abundant amino acid in our training set. We perform random under sampling for the training and validation sets. At the beginning of each epoch, residues with a count higher than the minimum are randomly resampled from the discarded frames. This approach effectively increases the number of residues observed by the network while addressing the amino acid frequency bias.

The CNN models were built on the Keras framework (Abadi *et al.*, 2015). All of the models presented in this paper were trained using the same culled PDB data set from PISCES (cullpdb_pc90_res3.0_R1.0_d200702_chains40583) containing over 35K non-redundant protein structures (40K+ chains), with resolutions up to 3.0 Å (see Supplementary Section 3; Wang & Dunbrack, 2003).

## Models tested

We performed fixed-backbone sequence design with a range of methods, both physics- and DL-based:

### Existing methods:

- **EvoEF2 (Physics)**: uses several energy functions, including hydrogen bonding, electrostatic attractions and van der Waals interactions (Huang *et al.*, 2020).
- **Rosetta (Physics)**: uses Monte Carlo methods to optimise the sequence for a template structure (Ludwiczak *et al.*, 2018).
- **ProDCoNN (CNN)**: replicates the CNN architecture described in Zhang *et al.* (2020).
- **DenseNet (CNN)**: implements the DenseNet architecture for image classification proposed by Huang *et al.* (2019), but converted the 2D operations into 3D.
- **DenseCPD (CNN)**: implements Qi & Zhang (2020) proposed using a 3D DenseNet-inspired architecture for sequence design.
- **ProteinMPNN (GNN)**: a GNN method using a message passing architecture by Dauparas *et al.* (2022a).

### Novel models:

- **TIMED (CNN)**: stands for Three-dimensional Inference Method for Efficient Design. One notable feature of this neural network is the use of a Global Average Pooling layer instead of a Fully Connected (Dense) layer to preserve spatial information (Lin *et al.*, 2013). The model also uses Spatial Dropout rather than standard dropout to help enforce this relationship. The model also incorporates Spatial Dropout for enforcing spatial relationships. See Supplementary Fig. 1 for a diagrammatic overview of the architecture.
- **TIMED_Unbalanced (CNN)**: this model is similar to TIMED but is trained without the balancing operation at each epoch.
- **TIMED_Polar and TIMED_Charge (CNN)**: these models are built on the TIMED architecture and include an additional channel in the frame to specify the polarity or charge, respectively, in addition to the atomic channels. The polarity is based on a Zimmerman score, of less than 20 for non-polar, (-1) and polar (+1) otherwise. The charge is encoded as −1, 0 or +1 depending on the charge of the amino acid (Zimmerman *et al.*, 1968). Both features are encoded at the location of the C$\alpha$ atom of the backbone in an separate channel of the input data.

We used Keras to build the CNN models for ProDCoNN, DenseCPD, DenseNet, TIMED (Chollet, 2015). We trained the models for 50 epochs. Training was performed using a combination of the Cambridge Service for Data-Driven Discovery (CSD3) (NVIDIA Tesla P100 16GB GPU and 36 cores) and our internal servers (Intel Core i9-10980XE CPU @ 3.00GHz with 36 cores and NVIDIA Quadro RTX 8000 48GB GPU). We used Weights & Biases for experiment tracking and hyperparameter sweeps (Biewald, 2020).

## Validating using PDBench and AlphaFold

To compare our novel models with those of the literature, we used the PDBench toolkit with its benchmark set (Castorina *et al.*, 2023) and AlphaFold 2 (AF2). (Jumper *et al.*, 2021). The metrics were then broken down by fold type. The benchmark set consists of a fold-balanced set of 595 protein structures classified into three main fold types: Mainly $\alpha$, Mainly $\beta$, $\alpha$-$\beta$ folds. Performance metrics for each model were evaluated overall and separately for each fold type. We used AF2 to fold the sequences predicted by the models. To alleviate the computational demands of AlphaFold, we evaluated shape metrics on 10% of the PDBench structures. We used PyMOL structural alignment command 'cealign' to calculate RMSD by comparing the original (crystal) structure and the AF2-predicted structure (Schrödinger, 2015).

### Sequence Metrics

PDBench calculates sequence metrics such as Macro-Recall, and Mean Absolute Error (MAE) for charge and isoelectric point. Macro-Recall is an accuracy metric that accounts for the class imbalance of amino acids:

$$\text{Macro-Recall} = \sum_{\text{classes}} \frac{\text{recall of class}}{\text{number of classes}}$$

The amino acid composition in proteins varies significantly across different protein folds. Macro-Recall ensures that the maximum accuracy for each amino acid is capped at 1/20 (5%).

MAE measures the average difference in charge and isoelectric point between the original and predicted sequences:

$$\text{MAE} = \frac{\sum_{i=1}^{N} |y_i - \hat{y}_i|}{N}$$

where $y_i$ represents the charge or isoelectric point for the original sequence, and $\hat{y}_i$ represents the predicted value. $N$ is the number of sequences analyzed.

### Fold recovery

We used AF2 (Jumper *et al.*, 2021) to predict the 3D structure of the predicted sequence. We used root mean squared deviation (RMSD) to compare the distance between the C$\alpha$ of the original crystal and the predicted structure:

$$\mathrm{RMSD} = \sqrt{\frac{\sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2}{n}}$$

We normalise RMSD for the length of the protein ($n$) using $\mathrm{RMSD}_{100}$ proposed by Carugo & Pongor (2008):

$$\mathrm{RMSD}_{100} = \frac{\mathrm{RMSD}}{1 + \ln\sqrt{\frac{n}{100}}}$$

To reduce the computational burden of AF2, we ran it on a subset of 59 randomly selected monomeric protein structures from the PDBench set (approximately 10% of the benchmark) covering Mainly $\alpha$, Mainly $\beta$ and $\alpha$-$\beta$ folds.

We predicted the residue sequences using each of the models described in Section 2.3. Subsequently, we folded the predicted sequences using AF2 to obtain the predicted 3D structure. We then calculated the RMSD between the original and the predicted structure for each model and for each fold type. We excluded structures with of the 'special fold' as they are highly irregular.

We used a local version of ColabFold (Mirdita *et al.*, 2021) called LocalFold. We used the CEalign command in PyMOL to calculate RMSD because of its robustness to low sequence similarity (Schrödinger, 2015).

### Monte-Carlo-based sampling of amino acid probability distributions

To generate final sequences we took the most likely amino acid at each position (argmax), but we also explored sampling from the sequence probability distributions using a Monte-Carlo-based method. In this case, the temperature affects the chance of selecting low-probability amino acids. As the temperature increases, the sequences become more random. The temperature was applied to the output probability distributions using the following equation:

$$\frac{\exp\left(z_i/T\right)}{\sum_j \exp\left(z_j/T\right)}$$

where $T$ represents the temperature, $j$ is the number of classes and $z_i$ denotes the predicted probability distribution for class $i$. A temperature of $T = 0$ corresponds to selecting the class with the highest probability (argmax), $T = 1$ maintains the original distributions and $T > 1$ leads to a more uniform (higher entropy) prediction across all classes.

We sampled 20 sequences from the probability distribution of each model for 57 randomly sampled proteins at temperatures 0.2, 0.6 and 1. We then used all five models of AF2 to fold the predicted sequences. Subsequently, each predicted structure was relaxed using the AMBER force field (Salomon-Ferrer *et al.*, 2013). The process resulted in approximately 8K relaxed PDB files, excluding any failed structures due to out of compute errors. The calculations were performed using the CSD3 with NVIDIA A100 GPUs.

We analysed the following metrics: packing density of the original and predicted structures, RMSD, AlphaFold lDDT, Shannon entropy and accuracy. The packing density for the predicted structure was calculated using ISAMBARD as the number of non-hydrogen backbone atoms within a radius of 7 Å (Wood *et al.*, 2017). RMSD was calculated using PyMOL and the alignment function 'cealign' (Schrödinger, 2015). Shannon entropy was calculated using SciPy and NumPy based on the probability distribution output of the models (Virtanen *et al.*, 2020). The maximum entropy can be computed using the formula: $\log_2 N_{\mathrm{classes}}$. For models predicting 20 amino acid classes, the maximum entropy is 4.32.

The full list of structures analysed is as follows: *1k5cA, 1kapP, 1dmlA, 1bx7A, 1igqA, 1jh6A, 1k5nA, 1c3mA, 1gp0A, 1jkeA, 1muwA, 1c1yB, 1b2pA, 1i7wB, 1a92A, 1a41A, 1devB, 1cruA, 1l0sA, 1iz5A, 1jofA, 1gprA, 1luzA, 1lpbA, 1ewfA, 1b8kA, 1hf2A, 1jm1A, 1kcfA, 1j5uA, 1jdwA, 1gxmA, 1lktA, 1lslA, 1io0A, 1b70A, 1itvA, 1k4zA, 1dvoA, 1hxrA, 1hq0A, 1j3aA, 1b77A, 1g3pA, 1kkoA, 1chdA, 1i4uA, 1genA, 1i4jA, 1ejdA, 1gppA, 1dqgA, 1flgA, 1jovA, 1g61A, 1h32A, 1ds1A*.

### TIMED-Design: tooling and user interface

TIMED-Design is an open-source Python repository that provides a user interface (UI) and various tool kits for the usage, analysis, and visualization of protein sequence design models. It can be used with any CNN model that takes frames as input, such as TIMED, DenseNet, DenseCPD, ProDCoNN.

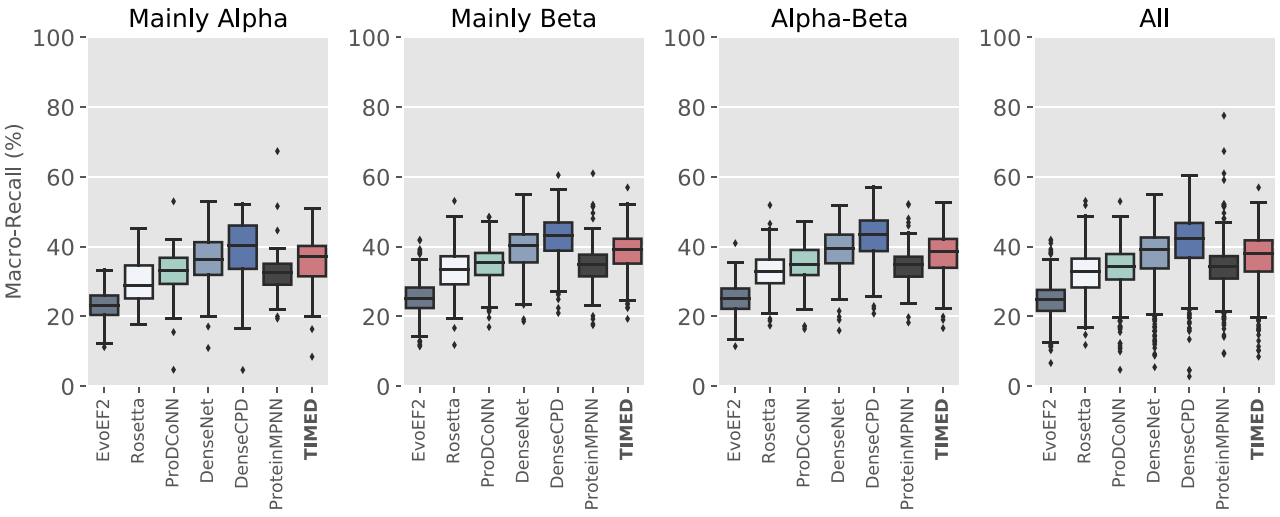
TIMED-Design includes a responsive UI created with Streamlit. The UI allows the selection of a PDB file from the Protein Data Bank or an option for uploading a file. Proteins are voxelised into frames using Aposteriori and predicted with the chosen model. The UI offers the following features:

- Metrics such as charge, isoelectric point, molecular weight and composition.
- Visualization of prediction probabilities distributions on the 3D structure of the protein.
- Performance plots, including precision/recall, prediction bias and sequence logo.
- Monte Carlo sampling at different temperature factors to generate novel sequences based on the probability distribution output of the CNN models.
- Confusion matrix between the original and the predicted amino acids.
- Prediction bias plot for the original and predicted sequence against the natural frequency of amino acids.

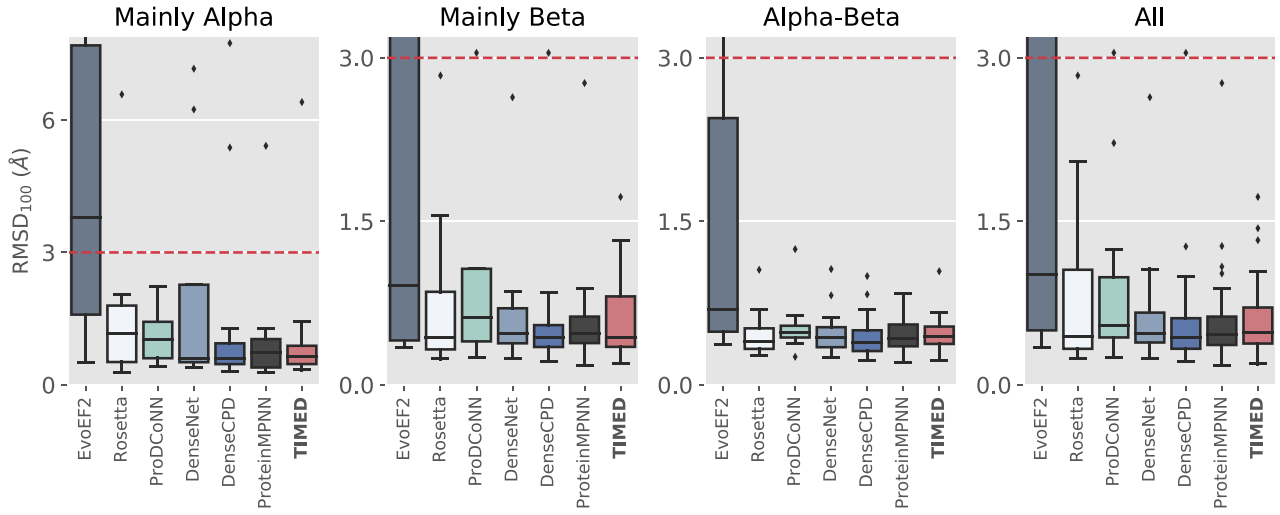
## Results

### CNN models and the state of the art

To assess the performance of the TIMED family of models, we compared them to other DL-based sequence design algorithms as well as some physics-based methods. For the CNN-based models, we took the most likely residue at each position to generate the final sequence. We compared performance using a range of accuracy metrics, as well evaluating recovery of the template fold by measuring the RSMD between an




(a) Macro-Recall Performance Comparison




(b) $RMSD_{100}$ Performance Comparison

**Fig. 2.** Performance metrics of physics and DL models. (**a**) Macro-Recall, sequence accuracy resistant to class imbalance of amino acids. (**b**) Fold recovery as determined by RMSD of the template structure to the AlphaFold model, normalized by the protein length. All plots are separated by fold type.

AlphaFold 2 model of the designed sequence with the template structure. On average, DL models outperform the physics-based methods (EvoEF2 and Rosetta) on accuracy and fold recovery metrics (Fig. 2). Overall, the performance of the CNN-based methods was higher than that of the GNN-based ProteinMPNN, although it is important to note that the reported performance is lower than the value reported in the original ProteinMPNN paper. In order to directly compare the architectures, all models in this paper were trained with the same training set, including ProteinMPNN, which led to a drop in performance. The DenseCPD architecture, which we reimplemented and made available, has the highest overall macro-recall, although fold recovery was similar across all the CNN- and GNN-based models.

## Balancing amino acids and prediction bias

We observed that balancing amino acids at training time (TIMED vs TIMED_Unbalanced) increases accuracy slightly though it has little effect on macro-recall. However, we wanted to determine whether balancing leads to biases in the selection of amino acids in the predicted sequences (Supplementary Fig. 11).

When TIMED and ProDCoNN were trained without balancing, while there was no change in macro-recall, the raw accuracy of sequence recovery on the benchmarking set increased by 6.32 and 6.90%, respectively. However, this also led to an increased prediction bias for the most common amino acids. The increased prediction bias is particularly prominent for alanine, glutamate and leucine in $\alpha$-helices (4, 10 and 6% bias in TIMED; 2, 15 and 10% in ProDCoNN, respectively) and, to a lesser extent, leucine and valine in $\beta$-sheets (3 and 5% in TIMED, and 4 and 7% in ProDCoNN).

When the amino acids were balanced through random undersampling, the prediction bias for all amino acids approached 0%, indicating that the predicted and true sequences have similar amino acid distributions.

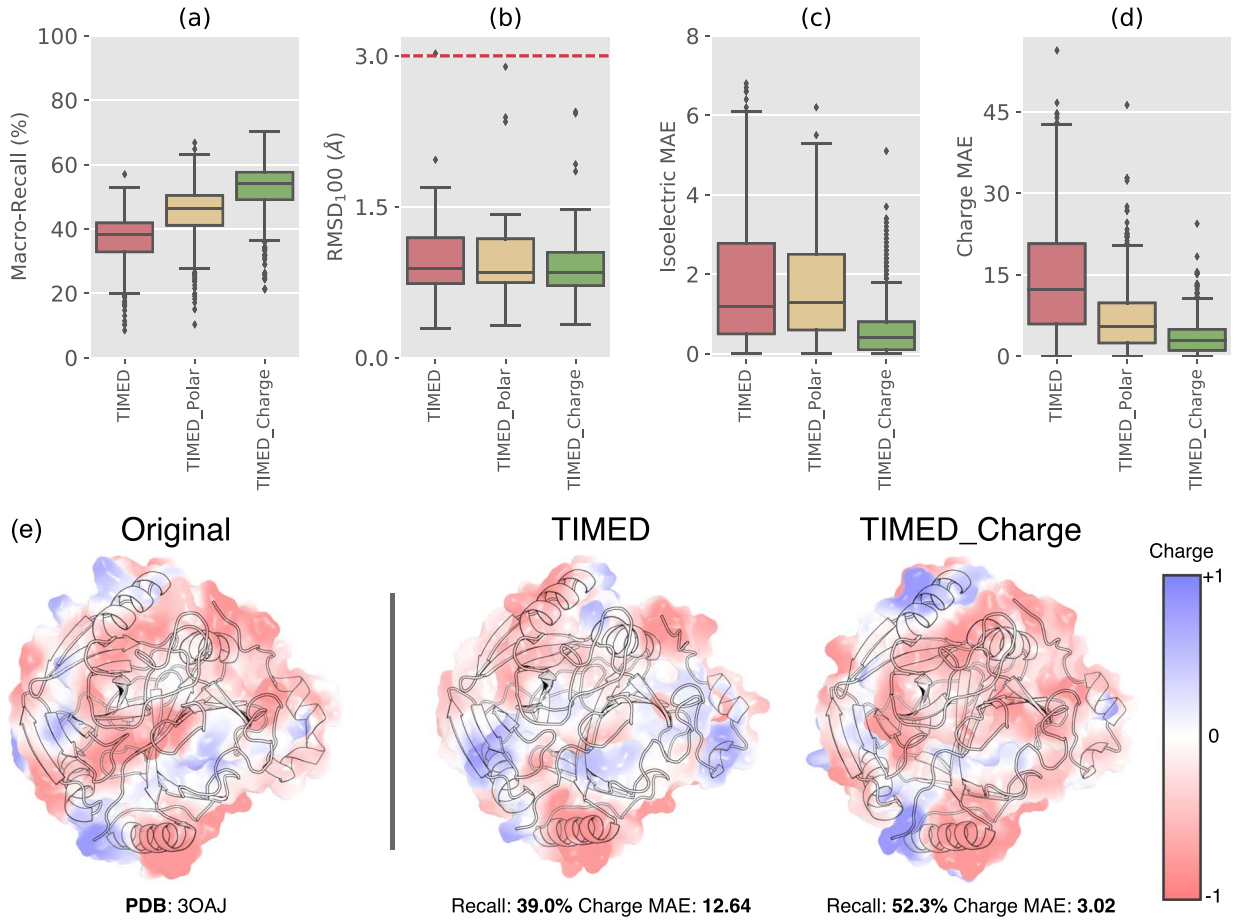



**Fig. 3.** Performance metrics of TIMED, our CNN model (red) and its variants TIMED_Polar (yellow) and TIMED_Charge (green). (**a**) Macro-Recall performance, sequence accuracy resistant to class imbalance of amino acids. (**b**) AlphaFold RMSD normalized by the protein length. (**c**) and (**d**) show the MAE for Isoelectric Point and Charge, respectively. See Supplementary Section 2 for these plots by fold and with other models. (**e**) Charge and Recall performance comparison of TIMED and TIMED_Charge model for PDB: 3OAJ. Both models correctly recover the target structure to under 1 Å RMSD; however, TIMED_Charge maintains charges and achieves higher sequence recall.

## Incorporating polarity and charge as design constraints

In typical protein design settings, designers often have specific constraints and requirements for the proteins they are designing, such as the incorporation of cofactor binding sites, preservation of active site residues or retention of charge compatibility. One powerful aspect of the voxel-based representation of proteins used by the CNN models presented here is that they can incorporate these constraints into the protein design process simply by adding channels that encode additional information to the input frame.

Here, we investigate the effect of incorporating polarity and charge as separate channels in the input frame. We compare the performance of TIMED with TIMED_Polar and TIMED_Charge. All models share the same underlying architecture. However, The TIMED_Polar and TIMED_Charge models receive additional polarity or charge information as input.

As Fig. 3 and Supplementary Figs 6–9 show, the TIMED_Polar and TIMED_Charge models outperform the TIMED model in terms of Macro-Recall. Notably, the TIMED_Charge model achieves significantly better performance across all metrics. The TIMED_Polar generally outperforms the TIMED model in all metrics, except in the AlphaFold RMSD of Mainly $\alpha$-helical folds where TIMED has a smaller range of RMSD values (Supplementary Fig. 7).

Comparing TIMED with TIMED_Charge, we observed that, while most designs exhibit similar RMSD values, the charge model better maintains overall charges, evident by the significantly lower charge MAE and higher recall achieved by the TIMED_Charge model. Figure 3(e) shows a protein with an equal RMSD of 1.05 Å for both models, while TIMED_Charge is better able than TIMED at effectively preserving areas of charges.

## Performance and dependence on resolution

We investigated the correlation between Macro-Recall Performance and Resolution per fold in Fig. 4. Among the DL models, the Mainly $\alpha$ folds exhibited the highest correlation between Macro-Recall and Resolution. In contrast, the physics models demonstrated a slightly stronger correlation between resolution and Macro-Recall in the $\beta$ folds compared to the $\alpha$-helical fold. The $\alpha$-$\beta$ folds generally exhibited low correlations.

Finally, we compared the overall correlation of performance between models, as shown in the Supplementary Fig. 12. The DL models were highly correlated among themselves. ProteinMPNN had the highest correlation with TIMED, while

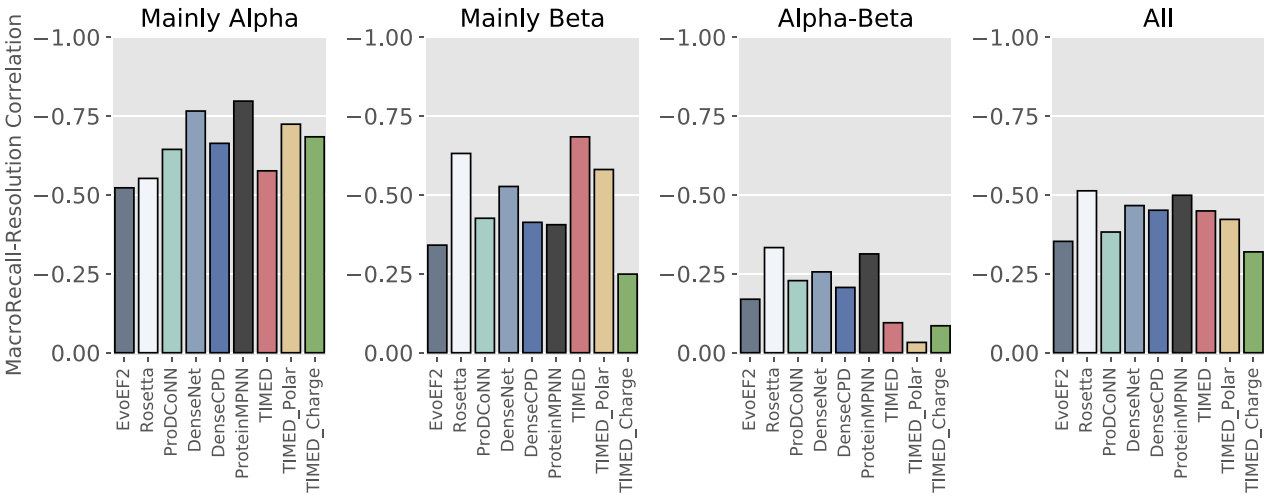



**Fig. 4.** Pearson correlation coefficients between Macro-Recall and resolution for models over different types of folds.

all CNN models had similar correlations between each other, indicating that the basis of predictions might be different between these models.

## Monte Carlo sampling for sequence generation

Once a probability distribution for amino acid identity has been generated for all amino acids in a protein, a sequence can be generated by drawing from these distributions. A naive approach would be to take the most probable amino acid at each position, but we could generate many more sequences by sampling from these distributions. We used TIMED-Design to perform weighted random sampling from the predicted probability distributions, with a temperature factor to increase sequence variability. Further to this, we hypothesised that the higher the performance of the model, the more robust sequence generation should be to higher temperatures.

To introduce diversity in the sampled sequences, we applied different temperature factors (0.2, 0.6 and (1) when sampling from the probability distributions then used AF2 to predict the structures of the sampled sequences. The following metrics were analyzed: Accuracy, Entropy, Mean Packing Density of the predicted structure, AlphaFold lDDT and RMSD using PyMol (Schrödinger, 2015). Entropy was calculated using the Shannon Entropy function of SciPy (Virtanen *et al.*, 2020). In Supplementary Table 1 and 2, we report the number of structures and average metrics.

### Correlation trends between sequence and fold recovery at different temperatures

First, in Fig. 5(a), we investigated the correlation between the accuracy of the TIMED model and the RMSD of the sampled structure by sampling sequences at different temperatures. We reasoned that, as we increase the temperature, the RMSD between the predicted structure of the designed sequence and the template structure would increase, but this behaviour might not be uniform across models. We calculated the Spearman correlation coefficient for accuracy and RMSD values and report the Spearman coefficients and corresponding $P$ values for each fold and temperature in Supplementary Table 2. We observed that higher accuracy is generally associated with lower RMSD, although the strength and significance of the correlation varies depending on the fold and temperature. All correlations had very significant $P$ values.

Next, we examined the correlation between the average prediction Shannon entropy of the output sequence probability distribution and RMSD, as illustrated in Fig. 5(b). Generally, higher entropy values were weakly correlated with higher RMSD, which makes sense as the Shannon entropy is roughly equivalent to the confidence of the amino acid identity at each position. The correlation is significant across all folds and is strongest in the mainly Mainly $\alpha$ fold and $\alpha$-$\beta$ and weaker in the Mainly $\beta$.

Finally, in Fig. 5(c), we investigated the correlation between the average prediction entropy and the AlphaFold lDDT. There was no clear correlation between these values.

### Correlation trends between performance and packing density

We also investigated the relationship between the Packing Density of the predicted structure and RMSD, Accuracy, and Prediction entropy at temperatures of 0.2, 0.6 and 1. We avoided temperatures higher than 1 as increasing the temperature further would results in random-like sequences and predicted structures with very high RMSD.

As shown in Fig. 5(d, e, f), there is generally a positive correlation between packing density and performance, indicating that regions of the proteins with higher packing density tend to be predicted with higher accuracy, lower RMSD, and lower entropy. The $P$ value for all of the correlations is very significant for all the performance metrics.

The correlation between Mean Packing Density and Accuracy (Fig. 5d) is significant and positive with a Spearman $r_s$ of 0.54. The correlation between Mean Packing Density and RMSD (Fig. 5e) is negative and significant with with a Spearman $r_s$ of −0.71. Finally, there is a weak correlation between Mean Packing Density and entropy (Fig. 5f) is negative and significant with a Spearman $r_s$ of −0.19. Interestingly, as shown in Supplementary Fig. 13, the Mean Packing Density and entropy correlation is much stronger for the Mainly $\alpha$ ($r_s$ −0.45) and $\alpha$-$\beta$ ($r_s$ −0.43) folds compared to $\beta$ ($r_s$ −0.08).

## TIMED-Design: a model-agnostic interface for protein sequence design models

All of the models we have described in this paper are publically available, including re-implementions of the other convolutional networks beyond our own. Furthermore, we

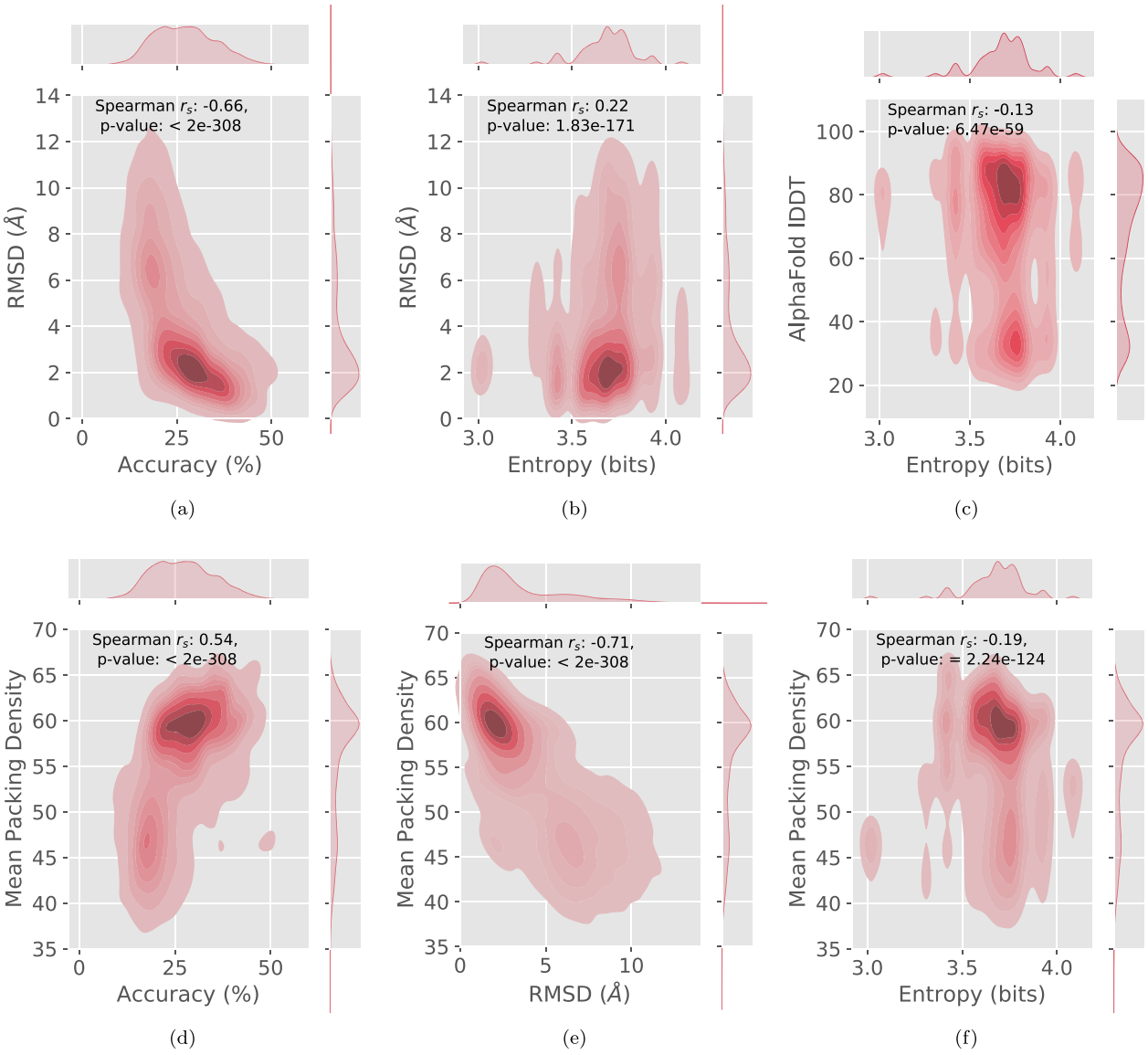



**Fig. 5.** RMSD performance against Accuracy (**a**) and Entropy (**b**), AlphaFold local distance difference test (lDDT) against Entropy (**c**), and Mean Packing Density of predicted structure against Accuracy (**d**), RMSD (**e**) and Entropy (**f**) for TIMED averaged across sampling temperatures (0.2, 0.6, 1). Sequences were sampled from the probability distribution of TIMED at different temperatures. The 3D shape of the predicted sequences was then computed through AlphaFold 2 and RMSD was calculated between the predicted shape and the original shape of each protein.

created TIMED-Design, an interface for designers to interact with these different sequence design models.

TIMED-Design is an open-source UI and CLI package built with Streamlit and Stmol (Nápoles-Duarte *et al.*, 2022). It currently features ProDCoNN, DenseCPD, DenseNet, TIMED, TIMED_Polar and TIMED_Charge.

## User interface

The UI allows user to select a backbone from a PDB code or upload a PDB file. The backbone is voxelised into frames by Aposteriori and the selected model is used to predict the most likely sequence (see Fig. 6a). We display the the most likely residues at each position with a sequence logo. Alternatively, an interactive plot of the predicted probabilities is also available, featuring the original ('ori') amino acid coloured in red (Fig. 6b).

The interface features several sequence metrics such as charge and isoelectric point. The probability distributions are displayed as a heatmap and as a sequence logo. Each position can be explored and visualised directly on the 3D shape of the protein.

Other plots include metrics plot (precision, recall and F1 score) per residue, prediction bias and a confusion matrix between the true residue in the protein chain and the predicted residue by the models. Other features include Monte Carlo temperature sampling where the user can specify the number of sequences to sample from the probability distribution.

In the case of TIMED_Polar and TIMED_Charge, the UI allows users to change or fix specific sites in the protein as polar/non-polar or positive/neutral/negative charge. This is helpful in the case of re-design of specific interfaces of the protein. The UI is available at the following URL: https://pragmaticproteindesign.bio.ed.ac.uk/timed

## Command line interface

The command line interface (CLI) features the same functionalities as the UI as well as additional analysis features

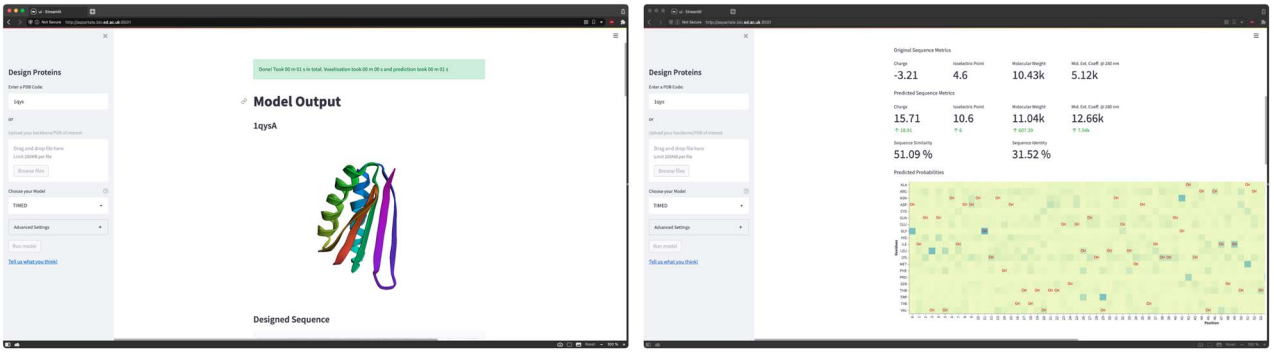



**Fig. 6.** Overview of the TIMED-Design user interface. (**a**) Once a backbone is selected, it is voxelised and predicted by the chosen model. (**b**) Prediction probabilities and designed sequence metrics.

involving further analysis and plots. All the scripts used to analyse or generate plots are present in a separate folder to allow customisation.

## Discussion

CNNs offer a flexible and performant architecture to encode spacial information for proteins. In this study, we introduce the TIMED models, which are state-of-the-art DL models for protein sequence design. We demonstrate that the expansion of the input voxel representation to include design-centric information such as polarity and charge, can drastically improve performance. Furthermore, while developing and benchmarking these methods, we made many observations about their performance that is important to consider when they are applied.

### Encoding additional information in the voxel input allows for broader applications for protein re-design

The polar and charge models implemented with the default TIMED architecture demonstrate superior performance and exhibit notable differences in property prediction compared to other models. While these models may not be suitable for truly *de novo* applications, they can be effectively used for protein redesign when both backbone and property information are available as input. Moreover, the UI allows users to selectively modify the property at specific positions, enabling targeted 're-painting' of specific regions of a protein with different properties, such as switching a binding region from positive to negative charge. The improved performance of these models suggests that further work on feature engineering for convolutional models is warranted, where more comprehensive information about the desired function of the protein could be incorporated within the voxel space, such as fixed amino acids in a catalytic site.

### Per-fold metrics for a granular performance overview

There is a notable difference in the performance of all the design models across the different fold classes for both sequence and fold recovery metrics (Fig. 2 and Supplementary Figs S2–S5). Interestingly, all the DL methods generally perform comparably well or better than physics methods when it comes to designing mainly $\beta$ structures, despite $\beta$ structures being historically challenging targets (Huang *et al.*, 2016; Woolfson *et al.*, 2015). Even more surprisingly, Mainly $\beta$ structures show a lower correlation between Macro-Recall and template structure resolution (Fig. 4) than Mainly $\alpha$ structures, so this cannot be explained by Mainly $\beta$ structures requiring higher quality templates.

Additionally, in Fig. 4, we see a strong correlation between Macro-Recall and Resolution, specifically for Mainly $\alpha$ and $\beta$ folds, and to a lesser extent for $\alpha$-$\beta$. For the Mainly $\alpha$, ProteinMPNN exhibits the strongest correlation, followed by DenseNet and TIMED_Polar.

### Balancing amino acids at training time reduces prediction bias

We explored the effect of balancing amino acid classes during training time. In some ways, unbalanced classes better reflect the biochemical availability of the individual amino acids, which could improve production of the proteins in living systems. However, this would mean that the biases of natural proteins would be reflected in the sequences produced by the design algorithms, when there is strong evidence that functional proteins exist in sequence spaces that are unexplored in nature (Weidmann *et al.*, 2019).

### AlphaFold as a tool for sequence design validation

We have discovered that accuracy, macro-recall and other statistical metrics can accurately estimate a model's performance only up to a certain point. For example, the EvoEF2 and ProDCoNN have similar performance as measured by sequence metrics. However, differences in RMSD in the AlphaFold2 predictions are at times significant, for example, in the Mainly $\alpha$ folds. Although AlphaFold comes at significant computational cost, alternative lighter-weight structure prediction algorithms, such as OmegaFold (Ruidong *et al.*, 2022) or ESMFold (Lin *et al.*, 2022), could be used in its place.

Additionally, most DL models perform similarly in terms of RMSD, usually under 3 Å and differences between models are usually less than 1 Å, which is the median RMSD from the original AlphaFold2 paper (Jumper *et al.*, 2021). Perhaps, after a certain level of performance, differences in accuracy metrics become less relevant. In the case of *de novo* design, for example, high accuracy might limit the utility of the design method, as the sequences produced will have lower variability. In real-world applications of protein, the increased diversity of lower accuracy models might be more desirable, especially when the experimental strategy involves high-throughput screening.

When comparing the performance of ProteinMPNN when trained with the culled PDB set (40K chains) and its performance using the full PDB (500K+ chains) we see a 33% increase in performance Dauparas *et al.* (2022a). It is evident therefore that the performance of most models is sensitive to the amount of training data. Selecting larger portions of training data would lead to higher sequence recovery. However, as all of these models achieve RMSD scores well below 3 Å, it is possible that training with smaller portions of the data set could be a way to obtain more 'creative' sequence designs, i.e. sequences with similar shapes but significantly different sequence similarity, although this requires further investigation.

### Monte Carlo sampling to produce sequences

The output of DL models for protein sequence design is a probability distribution for each amino acid of the template structure. In the case of the CNNs, the prediction of each position is independent of the next. This means that selecting the highest probable amino acid for each position may not be the best strategy for selecting sequences. An alternative approach is to sample from the probability distribution and generating many sequences.

In Fig. 5(b) and (c), we observed that the Shannon entropy of the output probability distributions correlates with RMSD and AlphaFold lDDT. This observation suggests that Shannon entropy could be used as a confidence score for sequence predictions in this case. Higher entropy values indicate more randomness in the distributions, indicating less confidence in the model's predictions. Calculating prediction entropy is significantly faster than running AlphaFold on the sequence, allowing high-entropy regions to be identified as candidates for further optimization. Futhermore, the Shannon entropy might be indicative of the quality of the backbone template and used as a basis for improvement of the template without computationally expensive simulation.

### Performance at different packing density varies with fold

We observe that performance, as measured by metrics such as RMSD and accuracy, varies at different packing densities (see Fig. 5 and Supplementary Fig. S13). Generally, higher packing density is correlated with better performance in the Mainly $\alpha$ and $\alpha$-$\beta$ folds. This correlation can be attributed to the fact that high-density regions often correspond to core of the protein, which are typically composed of hydrophobic residues (Banach *et al.*, 2020). Similar observations have been made in ProteinMPNN (Dauparas *et al.*, 2022a). Additionally, the reduced mobility of residues within hydrophobic cores, compared to solvent-exposed residues, contributes to a more well-defined backbone conformation. Interestingly, we find that the strength of these correlations varies across different folds. For instance, mainly $\beta$ folds exhibit a weaker correlations compared to mainly $\alpha$ and $\alpha$-$\beta$ folds in terms of accuracy and entropy. This highlights the influence of fold-specific characteristics on the relationship between packing density and performance metrics.

### TIMED-Design: UI and CLI

The TIMED-Design UI bridges the gap between designers and methods developers. The goal for the UI is to remove all the complexity involved in installing the models, the environment, and interpreting the predictions. To the best of our knowledge, TIMED-Design is the first model-agnostic UI for non-technical people to interact with state-of-the-art protein sequence design models. The CLI also offers users the ability to create scripts to further interact with these models programmatically. In future, we aim to incorporate TIMED-Design into DE-STRESS (Stam & Wood, 2021), our platform for evaluating protein designs, so that new designs can be generated, evaluated and shortlisted, all in one application.

## Conclusion

In this paper, we have demonstrated that CNNs are a powerful and flexible architecture for protein sequence design. We described the development and benchmarking of a range of high-performance sequence design algorithms, as well as the reimplementation of other CNNs from the literature. We have shown that voxelised representation of protein structure information is versatile and enables the incorporation of additional design considerations such as charge. Finally, we provided a public implementation of a few models and integrate them into a web-based design tool for insightful exploration and comparison.

## Acknowledgements

L.V.C. would like to thank Sumit Basu for helpful advice on improving the writing of the abstract of the paper.

## Author contributions statement

CWW and KS conceived research plan and supervised the work. CWW and LVC developed the TIMED architectures and the Aposteriori library. LVC developed the TIMED-Design UI. LCV and SMÜ trained the models presented in this paper and benchmarked their performance. CWW, LVC and KS prepared the manuscript. CWW acquired funding.

## Competing interests

No competing interest is declared.

## Funding

This work was supported by the Wellcome Trust-University of Edinburgh Institutional Strategic Support Fund [ISSF3]; Engineering and Physical Sciences Research Council (EPSRC) Fellowship (EP/S003002/1 to C.W.W.), Biotechnology and Biological Sciences Research Council Grant (BB/W013320/1); L.V.C. is supported by the UK Research and Innovation (UKRI) Centre for Doctoral Training in Biomedical AI at the University of Edinburgh (EP/S02431X/1 to L.V.C.); Royal Society University Research Fellowship (to K.S.). This work was performed using resources provided by the Cambridge Service for Data Driven Discovery (CSD3) operated by the University of Cambridge Research Computing Service (www.csd3.cam.ac.uk), provided by Dell EMC and Intel using Tier-2 funding from the Engineering and Physical Sciences Research Council (capital grant EP/T022159/1), and DiRAC funding from the Science and Technology Facilities Council (www.dirac.ac.uk).

## References

Abadi, M., Agarwal, A., Barham, P. *et al.* (2015) Software available from tensorflow.org.

Anand, N., Eguchi, R., Mathews, I.I. *et al.* (2022) *Nat. Commun.*, **13**, 746. https://doi.org/10.1038/s41467-022-28313-9.

Banach, M., Fabian, P., Stapor, K. *et al.* (2020) *Biomolecules*, **10**, 767. https://doi.org/10.3390/biom10050767.

Biewald, L. (2020) Software available from wandb.com.

Carugo, O. and Pongor, S. (2008) *Protein Sci.*, **10**, 1470–1473. https://doi.org/10.1110/ps.690101.

Castorina, L.V., Petrenas, R., Subr, K. *et al.* (2023) *Bioinformatics*, **39**, btad027. https://doi.org/10.1093/bioinformatics/btad027.
Chollet, F. *et al.* (2015) *Keras*, GitHub, 5, bcac37.
Dauparas, J., Anishchenko, I., Bennett, N. *et al.* (2022) *Science*, **378**, 49–56.
Deng, J., Dong, W., Socher, R., *et al.* Imagenet: a large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255. https://doi.org/10.1109/CVPR.2009.5206848.
Ferruz, N., Noske, J. and Höcker, B. (2021) *Bioinformatics*, **37**, 3182–3189. https://doi.org/10.1093/bioinformatics/btab253.
Ferruz, N., Schmidt, S. and Höcker, B. (2022) *Nat. Commun.*, **13**, 4348. https://doi.org/10.1038/s41467-022-32007-7.
Huang, G., Liu, Z., Pleiss, G. *et al.* (2022) *IEEE Trans. Pattern Anal. Mach. Intell.* **44**, 8704–8716. https://doi.org/10.1109/TPAMI.2019.2918284.
Huang, P.-S., Boyken, S.E. and Baker, D. (2016) *Nature*, **537**, 320–327. https://doi.org/10.1038/nature19946.
Huang, X., Pearce, R. and Zhang, Y. (2020) *Bioinformatics*, **36**, 1135–1142. https://doi.org/10.1093/bioinformatics/btz740.
Jumper, J., Evans, R., Pritzel, A. *et al.* (2021) *Nature*, **596**, 583–589. https://doi.org/10.1038/s41586-021-03819-2.
Krivacic, C., Kundert, K., Pan, X. *et al.* (2022) *Proc. Natl. Acad. Sci.*, **119**, e2115480119. https://doi.org/10.1073/pnas.2115480119.
Leman, J.K., Weitzner, B.D., Lewis, S.M. *et al.* (2020) *Nat. Methods*, **17**, 665–680. https://doi.org/10.1038/s41592-020-0848-2.
Lin, M., Chen, Q. and Yan, S. (2013) *Network in Network*, arXiv preprint arXiv:1312.4400.
Lin, T.-Y., Maire, M., Belongie, S., *et al.* (2014) Microsoft coco: common objects in context. In Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds), *Computer Vision – ECCV 2014*. Cham: Springer International Publishing, pp. 740–755. https://doi.org/10.1007/978-3-319-10602-1_48.
Lin, Z., Akin, H., Rao, R. *et al.* (2023) *Science* **379**, 1123–1130. https://doi.org/10.1126/science.ade2574.
Ludwiczak, J., Jarmula, A. and Dunin-Horkawicz, S. (2018) *J. Struct. Biol.*, **203**, 54–61. https://doi.org/10.1016/j.jsb.2018.02.004.
Mirdita, M., Ovchinnikov, S. and Steinegger, M. (2022) *Nature Methods*, **19**, 679–682. https://www.nature.com/articles/s41592-022-01488-1.
Nijkamp, E., Ruffolo, J., Weinstein, E.N. *et al.* (2023) *Cell Systems*, **14**, 968–978.e3. https://doi.org/10.1016/j.cels.2023.10.002.
Nápoles-Duarte, J.M., Biswas, A., Parker, M.I. *et al.* (2022) *Front. Mol. Biosci.*, **9**, 990846. https://doi.org/10.3389/fmolb.2022.990846.
Pan, X. and Kortemme, T. (2021) *J. Biol. Chem.*, **296**, PMID 33744284. https://www.jbc.org/article/S0021-9258(21)00336-7/abstract.
Porebski, B.T. and Buckle, A.M. (2016 ISSN 1741-0126) *Protein Eng. Des. Sel.*, **29**, 245–251. https://doi.org/10.1093/protein/gzw015.
Prapas, I., Derakhshan, B., Mahdiraji, A.R. *et al.* (2021) *Datenbank-Spektrum*, **21**, 203–212. https://doi.org/10.1007/s13222-021-00386-8.
Qi, Y. and Zhang, J.Z.H. (2020) *J. Chem. Inf. Model.*, **60**, 1245–1252. https://doi.org/10.1021/acs.jcim.0c00043.
Salomon-Ferrer, R., Case, D.A. and Walker, R.C. (2013) *Wiley Interdiscip. Rev.: Comput. Mol. Sci.*, **3**, 198–210. https://doi.org/10.1002/wcms.1121.
Schrödinger, L.L.C. (2015) *PyMOL The PyMOL Molecular Graphics System, Version 1.8*, Schrödinger, LLC.
Stam, M.J. and Wood, C.W. (2021) bioRxiv, 2021.04.28.441790.
Virtanen, P., Gommers, R., Oliphant, T.E. *et al.* (2020) *Nat. Methods*, **17**, 261–272. https://doi.org/10.1038/s41592-019-0686-2.
Wang, G. and Dunbrack, R.L. (2003) *Bioinformatics*, **19**, 1589–1591. https://doi.org/10.1093/bioinformatics/btg224.
Watson, J.L., Juergens, D., Bennett, N.R. *et al.* (2023) *Nature*, **620**, 1089–1100. https://doi.org/10.1038/s41586-023-06415-8.
Weidmann, L., Dijkstra, T., Kohlbacher, O. *et al.* (2019) *bioRxiv*, Cold Spring Harbor Laboratory, https://www.biorxiv.org/content/early/2021/06/27/706119.
Wood, C.W., Heal, J.W., Thomson, A.R. *et al.* (2017) *Bioinformatics*, **33**, 3043–3050. https://doi.org/10.1093/bioinformatics/btx352.
Woolfson, D.N. (2021) *J. Mol. Biol.*, **433**, 167160. https://doi.org/10.1016/j.jmb.2021.167160.
Woolfson, D.N., Bartlett, G.J., Burton, A.J. *et al.* (2015) *Curr. Opin. Struct. Biol.*, **33**, 16–26. https://doi.org/10.1016/j.sbi.2015.05.009.
Ruidong, W., Ding, F., Wang, R. *et al.* (2022) *bioRxiv*, Cold Spring Harbor Laboratory. https://doi.org/10.1101/2022.07.21.500999.
Yang, C., Sesterhenn, F., Bonet, J. *et al.* (2021) *Nat. Chem. Biol.*, **17**, 492–500. https://doi.org/10.1038/s41589-020-00699-x.
Zhang, Y., Chen, Y., Wang, C. *et al.* (2020) *Proteins*, **88**, 819–829. https://doi.org/10.1002/prot.25868.
Zhou, J., Panaitiu, A.E. and Grigoryan, G. (2020) *Proc. Natl. Acad. Sci.*, **117**, 1059–1068. https://doi.org/10.1073/pnas.1908723117.
Zimmerman, J.M., Eliezer, N. and Simha, R. (1968) *J. Theor. Biol.*, **21**, 170–201. https://doi.org/10.1016/0022-5193(68)90069-6.

# Chapter 5

# Fragments

## 5.1 Introduction

Over the past two decades, protein design has evolved from fragment-based methods to advanced deep learning approaches. This was driven by the growing availability of computational resources, curated protein datasets, and open-source machine learning frameworks, collectively pushing protein design to the next level.

Despite these advancements, significant challenges remain. Ingraham et al. [65] highlighted three major limitations of protein design models: (1) inability to jointly model the backbone structures and sequences, (2) with subquadratic scaling ($O(n^2)$) with protein length, and (3) integrating functional constraints without retraining. Beyond these, I would add that these models operate as "black boxes", lacking interpretability of the learned features.

This chapter reintroduces fragments as an efficient and interpretable alternative for protein design, demonstrating their ability to reduce dimensionality, accelerate functional searches, and improve generative modelling [26]. Using a curated set of 40 evolutionarily conserved functional fragments from Alva et al. [7], we show that fragment representations can:

1. Efficiently capture functional information while reducing dimensionality by up to 99%.

2. Improve the speed and accuracy of functional protein database searches.

3. Provide structural constraints that guide generative models to recover protein function.

4. Facilitate interpretability by mapping complex protein structures onto biologically meaningful and modular units.

In this final chapter, we first explore the historical context of fragment-based protein design, highlighting their early contributions and limitations. Then, we describe our methodological approach to representing proteins with fragments, covering detection algorithms and evaluation frameworks. Finally, we systematically evaluate fragment-based representations across three applications, functional clustering, database searches, and conditional protein generation, and discuss the implications of using fragments for the field.

## 5.2 Reimagining Protein Fragments

Fragment-based protein design emerged to reduce the astronomically large combinatorial space of protein structures during design. These methods use conserved structural elements as modular building blocks, similar to Lego pieces, combining them to create novel proteins with predictable properties [63].

Rosetta, developed in the mid-1990s by the Baker Lab, pioneered fragment-by-fragment design rather than atom-by-atom modelling, notably creating TOP7, the first computationally designed protein with a novel fold [71]. These fragments, typically 3-9 amino acids long, are selected from known structures based on sequence and structural similarity to short segments in known proteins. They are then assembled using Monte Carlo sampling and energy minimization to identify low-energy, native-like conformations [55].

The field then progressed from purely structural applications toward assembling fragments with defined functional roles. Rosetta's FunFoldDes implemented Motif Grafting, a technique that positions functional motifs within supporting scaffolding elements to support it [17]. A key application of this approach is vaccine design, where immunogenic peptides (i.e. epitopes) are grafted onto stable scaffolds to elicit targeted immune responses [32, 102].

Subsequently, several groups expanded fragment-based methods by incorporating functional fragments into their designs. SCHEMA [58] from the Frances H. Arnold Group, is a structure-guided recombination tool to create chimeric enzymes by swapping structurally compatible blocks from homologous proteins, generating variants with improved thermostability and catalytic activity.

The Grigoryan lab introduced Tertiary Structural Motifs (TERMs) and the dTERMen pipeline and computational designed a fluorescent protein [80, 128]. The Kuhlman lab introduced SEWING, a method for "stitching" fragments together, successfully producing stable helical proteins [66].

Fuzzle [47] extended this concept by cataloguing conserved subdomain-sized fragments shared across folds, many with ligand-binding roles, enabling their reuse in engineering novel binding sites. Similarly, Kolodny et al. [70] compiled sequence-based "THEMES" for functional protein analysis. More recently, Cao et al. [20] (Baker Lab) advanced fragment-based design by integrating Rotamer Interaction Field (RIF) Docking to assemble protein binders from structural motifs.

These protocols shifted the field from *de novo* atom-level modelling to the recombination of modular elements. Our approach builds on this by treating functional fragments not only as reusable parts, but as a coarse-grained, general-purpose representation for protein structure, search, and generation.

Current protein representations in state-of-the-art methods (Figure 2.5) face several critical limitations. Sequences and atomic coordinate representations require $O(n^2)$ complexity for attention-based models, making them computationally expensive for proteins longer than a few hundred amino acids. Sequence-based methods, while more efficient, may struggle to capture tertiary structural relationships [97]. Additionally, these representations create "black-box" models that learn functional features implicitly from data, with limited explainability behind specific design choices [65].

We propose fragments, not just as design tools, but as coarse-grained, functionally meaningful representations of proteins. This modular approach is based on the evolutionary origins of protein structures and functions, which are thought to have evolved through recombination, repetition, and accretion of small, functional peptides [6]. Representing proteins as evolutionarily conserved, functional fragments, significantly reduces the dimensionality (up to 99%) while preserving functional information. Fragment-based representations also significantly accelerate database searches, allow for functional clustering even at low sequence identity, and guide generative models such as RFDiffusion to produce backbones that successfully recover protein function. Finally, we provide a fast, open-source fragment detection and representation software implemented in Python and vectorised with NumPy, continuing the theme of accessible computational tools presented throughout this thesis.

## 5.3 Methodological Foundations of Fragment Techniques

Building on the historical context of fragment-based design, we developed a novel fragment detection framework that converts proteins into coarse-grained representations using 40 evolutionarily conserved structural fragments. This method not only identifies functional motifs within protein structures but also allows us to abstract proteins as either **Fragment Sets** or **Fragment Graphs**, each offering distinct advantages for computational analysis.

### 5.3.1 How to Abstract a Protein

Abstractions are powerful tools to reduce complexity while maintaining information. In other scientific domains, abstract representations such as Fourier transform convert signals from the time domain to the frequency domain, revealing periodic structures [18], molecular dynamics force fields allow the approximation of complex interaction energies while significantly reducing the computational costs [19, 23]. Additionally, in chemistry, compounds can be represented as text strings called Simplified Molecular Input Line Entry System (SMILES), used in software and deep learning applications [81, 117, 121].

Protein structures can be viewed at multiple hierarchical scales - ranging from individual atoms to secondary structures to complete structural domains, as discussed in Chapter 1. For example, secondary structures such as α-helices and β-sheets are used in ribbon diagrams to simplify visualisation [96]. These can be abstracted further to structural and functional motifs as TERM [80] or THEMES [70].

In deep learning applications, proteins are generally represented with atomic coordinates, sequences, graphs, or voxel frames (see Figure 2.5. As discussed earlier, however, these representations scale poorly with protein length, and require models to implicitly learn functional information from the data.

Here, we propose using fragments to abstract protein structures, independently of the amino acid sequence. Our representation uses 40 evolutionarily conserved structural fragments, identified by Alva et al. [7], as fundamental building blocks (see Figure 5.1). Each fragment is a recurring structural motif with distinct functions, such as DNA, RNA, Metal ion, GTP, and ATP binding. Using their data, we curated a set of 219 instances for the 40 fragments. These fragments are on average around 24 amino acids long, with a minimum length of 9 (Fragment 17) and maximum of 39 (Fragment 13).

We propose two complementary representations: **Fragment Sets** and **Fragment**

**Graphs** (see Paper Figure 1).

Fragment Sets provide a simplified and efficient representation by recording only the presence or absence of each fragment type, making them ideal for applications like database searches where speed is critical.

On the other hand, Fragment Graphs preserve structural context. Nodes represent fragments while edges represent peptide bonds and spatial proximity. This approach preserves the connectivity of how fragments are arranged in the three-dimensional structure, allowing for a more detailed functional analysis.

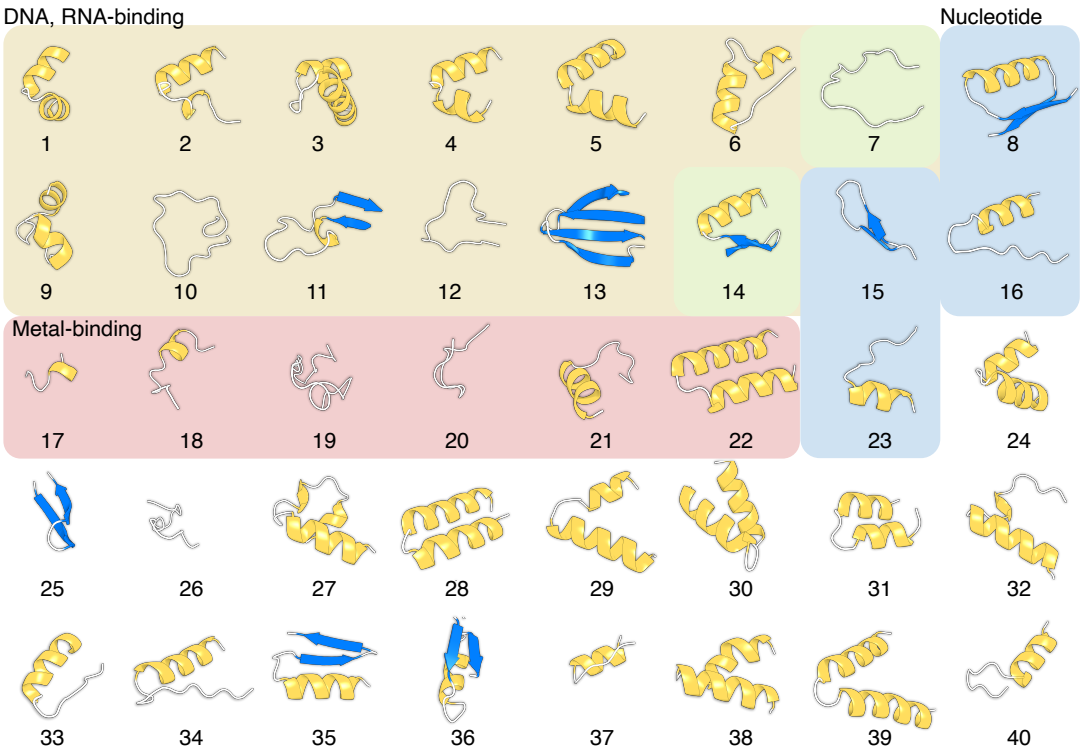



Figure 5.1: Overview of Evolutionarily Conserved Fragments from Alva et al. [7]. Fragments cover binding DNA, RNA, Nucleotides, Metal and other structural fragments. Fragments 7 and 14 are both DNA and Metal binding.

### 5.3.2 A Fragment Detection Framework

The detection algorithm uses a sliding window with varying lengths matching the sizes of our reference fragments, ranging from 9 to 37 residues. For each fragment number, the window length is fixed to match its length as defined in the reference library [7]. We slide the window along the target protein in 1-residue steps to ensure exhaustive coverage.

Fragment detection proceeds through a three-step process: First, fragments are detected in a given target structure. Then, the remaining unmatched regions are classified as unknown connectors or unknown fragments, depending on length. Finally, the classified structure is converted to a **Fragment Graph** or **Fragment Set** representation for further analysis.

For fragment detection, we implemented a sliding window algorithm (See Appendix Algorithm 1) that computes the distance between segments of the target protein and each reference fragment in our library.

The detection process compares each possible segment of the target protein by sliding a window of variable length corresponding to each fragment size. For each position, we extract the segment data and compute a distance against the fragments (as illustrated in Figure 5.2). We experimented with several metrics to calculate distance: sequence-based distances (sequence identity, BLOSUM distance) and angle-based metrics (RMS, RamRMSD, LogPr) which based entirely on the backbone torsion angles (that is, sequence-independent). We provide more details about how these metrics are calculated in the Appendix Section A.3.2. We avoided using the Hidden Markov Model cutoff metrics originally adopted by Alva et al. [7], as they are significantly slower and less scalable for dense sliding window operations across large datasets. In contrast, the distance metrics we used (e.g., RamRMSD, LogPr) are lightweight, fast to compute, and operate directly on structural features, making them better suited for high-throughput fragment detection.

Once the distances are calculated for all instances of the fragments, we normalised these values into probabilities within the range [0,1]. Regions with distances below the optimal threshold of 3.65% (determined through Receiver Operating Characteristic (ROC) analysis) are classified as matching fragments. When fragment matches overlap, we prioritise regions with lower distance scores and allow up to two amino acids of overlap between neighbouring fragments. The remaining unmatched regions are classified as unknown fragments if they are between 9-24 amino acids in length, or as unknown connectors if they are below 9 amino acids.

These classified protein regions are then converted into:

- **Fragment Sets** by recording the number of unique fragments in a structure, without connectivity information.
- **Fragment Graphs** by connecting fragment nodes with edges if they are bound by a peptide bond, or if they are spatially close ($<10\ \mathring{A}$). Edge features are one-hot

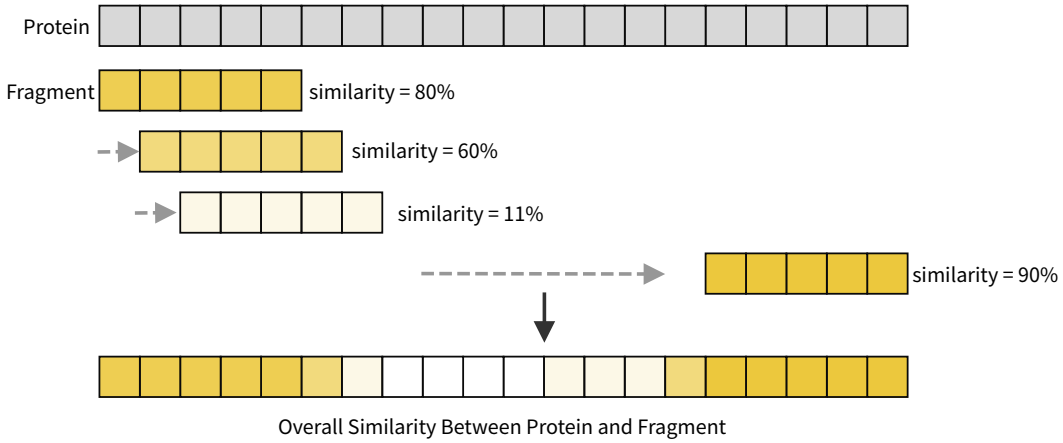



Figure 5.2: Sliding Window for Fragment Detection. A fragment is convolved across the protein structure, calculating the distance between each segment of the protein and the fragment. Regions below the threshold are classified as matching fragments, with priority given to lower-distance matches in cases of overlap. This process is repeated for all the instances of the fragments.

encoded to distinguish the type of connection.

We evaluated the strengths and weaknesses of each across different tasks.

Unlike the original analysis by Alva et al. [7], where each domain typically contained a single annotated fragment, our approach may identify multiple fragments within the same domain. This is expected as our method focuses on coarse-grained structural similarity rather than strict annotation, aiming to detect areas of structural similarities. Since we scan the entire structure for fragment-like regions, shorter fragments, such as Fragment 17 (which is just a small helix), can appear alongside other fragments. Rather than enforcing a one-fragment-per-domain rule, we prioritise coverage and reuse, consistent with the modular nature of protein evolution.

### 5.3.3 Fragments Functional and Structural Evaluation

To evaluate the effectiveness of our fragment-based representations we used two distinct datasets:

- **PDBench:** described in Chapter 3, to evaluate whether fragment-based representations preserve important structural and chemical properties across different folds.

- **Protein Function Dataset (PFD):** A custom dataset of 215 protein monomers spanning 12 binding functions for DNA, RNA, metal ions, GTP, ATP, and all combinations thereof. To ensure structural diversity, we enforced a sequence identity cutoff of $\leq$30% using the PDB Advanced Search[14] interface and filtered structures based on Gene Ontology (GO) codes [11]. We tried to select 10 structures per functional category wherever possible.

Then, we compare them against traditional sequence- and structure- based representations across three key applications:

1. **Functional Clustering** (Section 5.3.3.1): We assessed how well each representation captures functional relationships between proteins by clustering by functional categories and evaluating cluster quality using metrics such as Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), and Silhouette scores.

2. **Functional Searches** (Section 5.3.3.2): We evaluated the speed and accuracy of fragment-based database searches compared to traditional sequence- and structure-based methods, measuring both query time, memory requirements and retrieval quality through Normalized Discounted Cumulative Gain (NDCG) and Area Under the Receiver Operating Characteristic Curve (AUROC) scores.

3. **Conditional Generation** (Section 5.3.3.3): We used fragments as structural constraints to guide RFDiffusion in generating protein backbones with specific functional properties, evaluating whether the generated structures maintain the functional signatures of the original template proteins.

#### 5.3.3.1 Functional Clustering

We used clustering to evaluate how well fragment-based representations capture functional relationships between proteins. We specifically do this on the PFD dataset with low sequence similarity to simulate real-world scenarios of designed proteins with little similarity to known ones, or proteins from newly discovered organisms, for which no functional annotation exists.

First, we created pairwise distances between all the proteins in our dataset. We used traditional metrics such as RMSD (structure-based), BLOSUM62 (sequence-based), as well as fragment-based metrics BagOfNodes (Fragment Set difference) and GraphEditDistance (Fragment Graph difference). Then, we create embeddings using Principal

Coordinate Analysis (PCoA) (distance-preserving) and t-Distributed Stochastic Neighbor Embedding (t-SNE). Finally, we clustered proteins in a 10-dimensional embedding space using Gaussian Mixture Models (GMMs) and K-Means, and set the number of cluster to 12, corresponding with the number of unique functions in the PFD, and using several clustering metrics to assess the quality of the clustering. We also analysed the variance explained at different dimensions to compare the quality of the embedding space based on fragments and traditional representations.

#### 5.3.3.2 Functional Searches

During a design problem, retrieving functionally similar proteins can help identify different configurations that perform similar functions. At low sequence similarity however, traditional methods struggle to identify similarities for example because functional motifs are separated by large sequence or structural segments.

For each entry in the PFD, we evaluated retrieval performance using RMSD, BLOSUM, BagOfNodes, and GraphEditDistance (Section 5.3.3.1). We expected lower-distance entries to correspond to the same functions as the query.

Then, to assess the quality of the results, we used NDCG and AUROC.

#### 5.3.3.3 Conditional Generation

As mentioned in the introduction, current generative models struggle to conditionally generate functional proteins without retraining. Fragments, however, represent functional units of proteins. We therefore hypothesised that in backbone generation, fragments could be used as shortcuts to encode functional information, and the generative model could focus on generating a scaffold.

To do this, we took all the proteins in the PFD and masked non-fragments regions. Then, we used RFdiffusion to fill in the gaps between fragments, generating 5 backbones for each protein. We then used FoldSeek's[111] sequence-independent search, TM-Align[124], to identify the most similar structure in the PDB. We expected functional-looking backbones to retrieve proteins with similar functions among the top results.

## 5.4 Evaluating Fragments for Protein Functions and Design

### 5.4.1 Fragments Capture Physicochemical and Structural Properties

Converting to fragment-based representations requires accurate fragment detection algorithms. After systematically testing several metrics, we found that combining backbone-only metrics, specifically LogPr and RamRMSD, achieved the best performance with an F1 score of 0.85 (see Appendix Figure A.3). This combination also outperformed sequence metrics. Through comprehensive ROC analysis, we calculated that a distance threshold of 3.65% optimized the performance balance, maximizing true positive and minimising false positives.

Using the PDBench dataset, we analysed the physicochemical properties preserved by fragments regions compared to non-fragment areas. We found that fragments had higher proportions of intra-fragment hydrogen bonds, approximately 15% more in mainly β folds. On the other hand, fragments had lower inter-fragment hydrogen bonds, particularly in the mainly α folds with 47% less. Finally, we saw a slight decrease of surface accessibility in fragment regions (about 5%) and no significant difference in charge and polarity (see Appendix Section A.3.4).

### 5.4.2 Embedding Protein Function: Clustering with Fragment-Based Representations

In our analysis, using multiple clustering performance metrics, we found that fragment-based representations significantly improved protein separation according to their functional categories compared to traditional representations. Additionally, fragment-based embeddings retained significantly more information: BagOfNodes (95%) and GraphEditDistance (80%) within 20 dimensions, compared to BLOSUM (60%) and RMSD (40%). (See Paper Figure 2).

The GraphEditDistance metric achieved the best overall F1 score (0.1985) and ARI (0.0458). These values, while numerically modest due to the challenging nature of functional clustering at low sequence identity, represent substantial improvements over sequence-based and structure-based methods. BagOfNodes demonstrated exceptional cohesion with the highest Silhouette score (0.8227, on a scale from -1 to 1 where values

near 1 indicate excellent cluster separation) and nearly perfect distance preservation (Trustworthiness = 0.9991, Distance Correlation = 0.9998) (See Paper Table 1).

Interestingly, fragment-based distances showed strong correlation with sequence-based distances despite not directly using sequence information. GraphEditDistance achieved a Spearman correlation of 0.91 with BLOSUM distances and significant p-values. In contrast, RMSD showed minimal or slightly negative correlation with other metrics (see Appendix Figure A.18).

### 5.4.3 Fragments for Efficient Functional Searches

Fragment-based search methods showed a great trade-off between accuracy and speed. In terms of accuracy, fragment-based methods performed similarly to traditional approaches across most functional categories, with particularly strong performance in identifying complex multi-functional proteins. For DNA+ATP+GTP-binding proteins, fragment-based methods achieved AUROC values exceeding 0.8, outperforming both RMSD (0.56) and BLOSUM (0.75).

The most significant advantage of fragment-based methods was their computational efficiency. Our BagOfNodes implementation completed 100 queries in under 0.07 seconds, a speedup of approximately 68.7$\times$ (including initilisation costs) compared to RMSD-based searches (1717 seconds) and 1.64$\times$ compared to sequence-based BLOSUM searches (36.57 seconds). This significant increase in performance stemmed from the substantial dimensionality reduction achieved by fragment-based representations. Compared to backbone atom representation (RMSD), fragment-based representations reduced dimensionality by 99.1% for fragment graphs and 99.7% for fragment sets.

Even when including the one-time initialization cost of converting structures to fragment representations (approximately 25 seconds for all the 100 structures), fragment-based methods remained competitive for any application requiring multiple queries.

### 5.4.4 Fragments as Blueprints for Generative Models

To address the conditional generation of protein structure, we showed that functional fragments could effectively retain the functional signatures of the original protein and guide RFDiffusion to recover the target function, without requiring model retraining.

Metal-binding proteins achieved nearly perfect functional recovery. Single-function designs generally showed higher recovery rates compared to their multi-functional counterparts. Among dual-function proteins, DNA-binding combinations such as

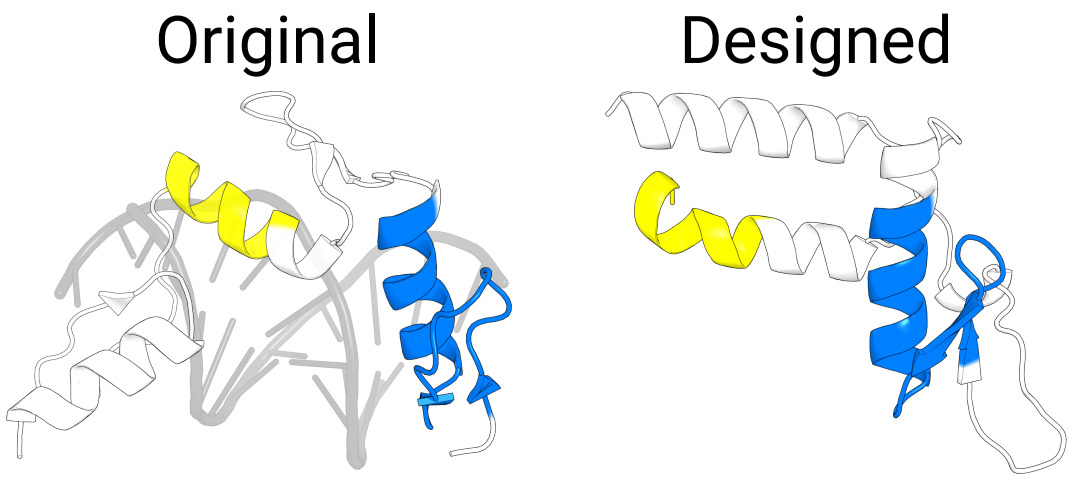



Figure 5.3: Comparison between an original Zinc finger protein (PDB:1A1F, left) and a design generated by fragment-based RFDiffusion (right). Fragments 14 (Blue - Zinc finger) and 17 (Yellow metal-binding) were fixed during generation. Despite differences in the overall structure (RMSD: 5.45 $\AA$), the generated protein maintains the functional conformation of the key fragments, demonstrating how generative models can produce diverse yet functionally consistent designs.

DNA+ATP, DNA+Metal, DNA+RNA, were slightly more successful with recovery rates around 80%. Interestingly, the triple-function combinations DNA+ATP+GTP and DNA+RNA+Metal achieved much higher recovery rates than all the dual function proteins (See Paper Figure 3).

Figure 5.3 shows an example of a Zinc finger DNA-binding protein. Our detection algorithm identified Fragment 14, associated with Zinc Fingers, and Fragment 17, associated with metal-binding. While the fragments were fixed, RFDiffusion generated new "connections", resulting in designs that differ significantly from the original (RMSD: 5.45 $\AA$). This highlights how current generative methods to sample functionally consistent yet structurally diverse proteins.

## 5.5 Paper Overview and Contribution

Below we present the paper: "**From Atoms to Fragments: A Coarse Representation for Efficient and Functional Protein Design**" [26]. We use evolutionarily conserved fragments identified by Alva et al. [7] to create coarse representations of protein structures. This significantly decreases computational complexity and improves efficiency

compared to traditional representations.

Current protein representations demand high-end computational resources, creating barriers to accessibility [65]. This trend is not sustainable and restricts protein design to those with sufficient computational resources. Our paper takes a fundamentally different approach by drastically reducing computational requirements through fragment-based representations. This approach allows protein function clustering, functional search, and blueprints for proteins generative models on modest hardware, democratising access to advanced protein design capabilities while maintaining functional accuracy.

We found that fragment-based embeddings are able to capture much more information at significantly less dimensions compared to traditional methods. This implies that models built on fragments should require significantly less number of dimensions to represent the same variance. We also noticed that fragment embeddings generally outperform other embeddings when used for functional clustering. The performance, however, is far from perfect, and this may be due to the low sequence similarity, the simplicity of the distances used, or the limited number of fragments in the library. Future work could explore learned embeddings based on fragments, for example, by using VAEs and using the embedding space to classify functions, or unsupervised learning methods [97]..

For functional search, fragments emerge as the fastest method compared to traditional methods. The beauty of this approach is that the distances used for search are really simple, light-weight, and easy to interpret. We believe that once the field identifies a good set of fragments, fragment-based searches may completely replace structural searches based on TM-Align for example, as they are notoriously slow and require a lot of compute [124].

Interestingly, we found that fragments can serve as blueprints for generative models to recover functional backbones. Much like in classical fragment-based design, fragments are shortcuts that are functionally or structurally meaningful, so this is no surprise [20, 71]. The great advantage of this however, is that we could strip a protein down to the functional component and interact with the fragment graph until we are satisfied with the design. Then we could use models to position fragments in space, respecting the constraints and use existing models for functional protein generation. Or replace these models all together with fragment representations instead of atomic ones, significantly decreasing the memory requirements and the model size.

Beyond protein design, fragment-based representations could also serve as visualisation or pedagogical tools, simplifying complex structures for improved interpretability.

There are fundamental limitations in this paper, for example, we use a limited set of fragments that do not fully represent the richness in protein function. Additionally, our fragment detection algorithm is based on very simple formulas and could be refined to include a more probabilistic approach to classification. For example, the classification could take into account the (prior) variation of observed torsion angles at specific positions of the fragments, and quantify the likelihood of observing that angle given the prior distribution. Nevertheless, we hope this to be a starting point towards showing that fragments can have applications beyond classical fragment-based design.

I would like to acknowledge my co-authors Christopher W. Wood and Kartic Subr. Chris and Kartic have guided me throughout the project. Together we came up with the idea for abstracting protein structures with fragments and wrote the paper. I implemented the library in Python and produced the figures.

*Journal Title Here,* 2022, pp. 1–9

**doi: DOI HERE**
Advance Access Publication Date: Day Month Year
Paper

PAPER

# From Atoms to Fragments: A Coarse Representation for Efficient and Functional Protein Design

Leonardo V. Castorina[1], Christopher W. Wood[2] and Kartic Subr[1]*

[1]School of Informatics, The University of Edinburgh, 10 Crichton Street, Newington, Edinburgh, EH8 9AB, UK and [2]School of Biological Sciences, The University of Edinburgh, Roger Land Building, Edinburgh , EH9 3FF, UK
*Corresponding author. k.subr@ed.ac.uk
FOR PUBLISHER ONLY Received on Date Month Year; revised on Date Month Year; accepted on Date Month Year

## Abstract

**Motivation:** Although deep learning has accelerated protein design, current protein representations such as sequences or full-atom structures scale non-linearly with protein length. We propose a sparse and interpretable representation for proteins, based on evolutionarily conserved fragments. Specifically, we use a curated set of 40 functional and evolutionarily conserved fragments as an alphabet to build Fragment Graphs and Fragment Sets. These fragment-based representations are both lightweight and functionally informative capturing up to 55% more variance using fewer than $\frac{1}{3}$ of the dimensions required by traditional methods.
**Results:** On a dataset of 215 functionally diverse proteins, our approach creates more coherent functional clusters than traditional sequence- and structure-based methods, even among proteins with $\leq 30\%$ sequence identity. Fragment-based searches of protein databases achieve accuracies comparable to traditional methods, while using 90% fewer tokens per protein. These searches execute $\sim$68.7$\times$ faster than RMSD-based structural methods and $\sim$1.64$\times$ faster than sequence-based methods, even including fragment pre-processing overhead. Additionally, we show that our representation effectively guides RFDiffusion for protein backbone generation with functional recovery rates higher than 40%. In summary, our fragment-based representation offers a scalable and interpretable alternative for the next generation of protein design tools for backbone design, sequence design, and functional similarity searches within protein structure databases.
**Availability:** Code will be available soon.

**Key words:** Protein Representation, Protein Design, Protein Search, Fragments

## Introduction

Designing functional proteins could transform medicine, biotechnology, and sustainability. From enzymes that catalyze reactions, to vaccines against target diseases, proteins are precise molecular tools. Yet, designing proteins remains a computationally intractable problem due to the combinatorial nature of the search space, making exhaustive search unfeasible.

Artificial Intelligence (AI) methods enabled *de novo* design of protein binders [14], antibodies [28], and enzymes [33, 23], using diverse representations: language models treat proteins as sequences [11, 26], diffusion models encode backbone geometry as vector frames [31, 1], and other approaches represent structure as voxel grids [8, 24] or graphs [10].

Despite enabling breakthroughs, these representations impose scale non-linearly with protein size, leading to high computational costs and complex models, highlighting the need for efficient and interpretable alternatives.

We propose **fragment-based representations** that model proteins as combinations of evolutionarily conserved structural fragments instead of sequences or full atomic coordinates (See Fig. 1). This approach stems from protein evolution theory, where structures evolved from recombination, repetition, and accretion of small, functional peptides [2]. We show that fragments significantly reduce dimensionality while preserving functional signatures, enabling faster, more interpretable protein search and design.

Proteins naturally lend themselves to abstraction at multiple scales, from secondary structures such as $\alpha$-helices and $\beta$-sheets [25] to tertiary structural motifs strongly associated with molecular functions such as $\beta$ hairpins [21]. Our fragment-based representation continues this trend, using fragments as functional building blocks rather than full atomic structures.

Previous studies have identified and leveraged recurring structural motifs for protein analysis and design. Alva et al. [3] identified conserved ancient fragments associated with binding functions. Frappier et al. [13] introduced recurring Tertiary Structural Motifs (TERMs) to design protein binders. Kolodny et al. [20] compiled several sequence-based "THEMES" for functional protein analysis. Fragment-based representations have also been used beyond proteins, such as fragSMILES [22]

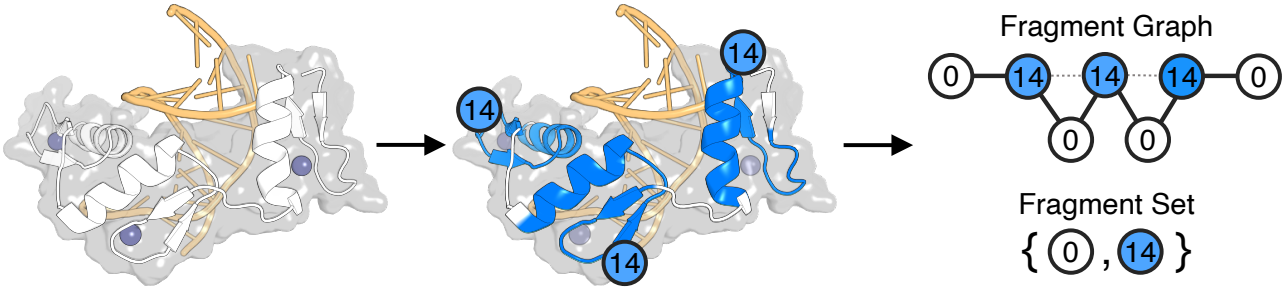



**Fig. 1.** Fragment-based protein representation of the ZIF268 Zinc Finger (PDB: 1AAY). Detected Fragments 14 (in blue), correspond to DNA- and metal- binding functions essential for zinc fingers. Unclassified regions are labeled as "unknown" (white). The structure is represented as a Fragment Graph, which preserves connectivity via peptide bonds (dark edges) and spatial proximity (dotted edges) or as a Fragment Set, with unique fragment types. DNA shown in yellow; zinc ions in purple.

for chemical representations and component-based approaches in computer vision [6].

Building on these ideas, our method explicitly encodes the backbone geometry rather than sequences or atomic coordinates, decomposing proteins into functional fragments. Using a library of 40 conserved fragments, we demonstrate three key applications of our fragment-based approach: (1) functional clustering to evaluate how well fragments capture protein function; (2) database searching to demonstrate effectiveness in retrieving functional proteins and computational efficiency; and (3) protein design using fragments as blueprints to guide RFdiffusion to generate backbones with functional signatures.

Additionally, we provide a fully vectorized Python package (MIT License) for fragment detection and representation, adaptable to any fragment library.

## Methods

We propose representing proteins abstractly as compositions of evolutionarily conserved fragments. Using 40 fragments identified by Alva et al. [3], we first explain fragment-based representations and then evaluate them via functional clustering, database searching, and protein design, comparing against traditional representations.

### Fragments as a Coarse Representation of Proteins

Given a protein structure from a PDB file, our representation decomposes it using evolutionarily conserved fragment motifs. We propose two representations for the protein structure, without sequence information, as a **Fragment Graph** or as a **Fragment Set**. In the former, nodes in the graph represent fragments and edges denote peptide bonds or spatial proximity. Fragment Sets, on the other hand, contain lists of unique fragments present, regardless of their arrangement (See Fig. 1).

To construct the reference library, we extracted all instances of the 40 fragments from Alva et al. [3] from PDB structures using AMPAL [32]. Then we filtered them to ensure matching length and sequence, yielding 219 verified instances (See Supp. Table 1).

### Building Fragment Representations

Fragment representation involves three steps: (1) detecting the fragments in the structure, (2) classifying unmatched regions, and (3) converting to a graph or a set (See Fig. 1).

Fragment detection identifies segments of the target protein matching library fragments below a distance threshold using a sliding window algorithm (see Supp. Algorithm 1). For this distance calculation, we evaluated several distance metrics:

- **Sequence-based** (sequence identity, BLOSUM distance) measure distance in amino acid sequence [17].
- **Angle-based** (RMS, RamRMSD, LogPr) are sequence-independent and measure distance in backbone torsion angles ($\phi$ and $\psi$) [19].

This yields a normalized **Fragment Distance Matrix** $D$, where each entry $D_{ik}$ denotes the distance between index $T_i$ of the target structure and fragment $F_k$ from the library. A segment is classified as matching a fragment if $D_{ik} < 3.65\%$ (determined via ROC analysis). If fragment matches overlap, we retain the one with lower distances, allowing up to two amino acids overlap between adjacent fragments.

Unmatched regions are classified as **unknown fragments** (9–24 residues) or **unknown connectors** ($< 9$ residues).

Finally, we represent classified regions as:

- **Fragment Sets:** presence or absence of fragment classes without considering connectivity information.
- **Fragment Graphs:** graphs with nodes for each fragments and edges for peptide bonds or spatial proximity (<10 Å). Edge features are one-hot encoded by types.

Fragment Sets are suited for speed-critical tasks like search, while Fragment Graphs preserve structural context, useful for clustering and design.

### Datasets for Validation

To validate whether fragments preserve structural and chemical properties, we used **PDBench** [7], a fold-balanced structures dataset. We quantified properties such as hydrogen bonding and solvent accessibility in fragment and non-fragment regions. For each property, we computed the correlation between its proportion in fragment regions and the overall fraction of the protein covered by fragments.

To evaluate functional clustering, database search, and design, we curated the **Protein Function Dataset (PFD)**, including 215 protein monomers across 12 functional binding classes (DNA, RNA, metal ions, GTP, ATP, and combinations). Structures were filtered using Gene Ontology (GO) codes [4] and $\leq 30\%$ sequence identity via the PDB Advanced Search [5]. Where possible, we selected 10 representative structures per functional category (detailed in Supp. Table 1).

### Fragments for Functional Clustering

We evaluated whether fragment-based representations better capture functional similarities between proteins compared to traditional structure- and sequence-based approaches. Specifically, we assessed whether proteins with similar functions form more coherent clusters under different representations.

For all protein pairs in the PFD, we computed pairwise distances using four metrics:

1. **RMSD (Root Mean Square Deviation)**: Structural distance using BioPython's CEAligner [29, 9].
2. **BLOSUM62**: Sequence distance using pairwise alignment scores based on amino acid substitution frequencies from BLOSUM[17].
3. **BagOfNodes**: Set distance based on the presence or absence of fragments (Fragment Sets).
4. **GraphEditDistance**: Graph distance accounting for both fragment identity and spatial arrangement (Fragment Graphs) [12].

Then, we projected each protein into a distance-preserving latent space of increasing dimensionality using Principal Coordinate Analysis (PCoA) and t-SNE, to explore how representation preserves functional relationships at various dimensions.

To quantify clustering performance, we applied Gaussian Mixture Models (GMM) and K-Means with 12 clusters, matching the known functional categories in PFD. To measure robustness and comprehensively evaluate clustering quality, we use Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), Silhouette and Trustworthiness scores, F1-score, and the correlation between embedding distances and the original pairwise distances.

### Fragments for Functional-based Searches

While the previous experiment evaluated the quality of the representation, here we assessed whether fragment-based representations support fast and meaningful protein database searches. Specifically, we measured retrieval speed, memory requirements, and functional relevance of the results.

We benchmarked fragment-based methods GraphEditDistance and BagOfNodes, against traditional methods RMSD and BLOSUM) querying 1, 10, and 100 proteins from a database of 100 structures using 35 cores [1]. For each method, we recorded initialization time, query time, and memory usage, measured as the number of data points per protein..

To assess retrieval quality, we queried each PFD protein against all others and ranked the results by distance. We then evaluated whether functionally similar proteins ranked higher using Normalized Discounted Cumulative Gain (NDCG) and Area Under the Receiver Operating Characteristic (AUROC) (see Supp. Section 13).

### From Fragments to Functional Proteins

Finally, we explored using fragments as blueprints for generating functional proteins by providing structural constraints to RFDiffusion. We hypothesized that if fragments capture functional information, then fragment-derived templates should yield structures similar to known proteins with the same function.

To test this, we used RFDiffusion[31], a state-of-the-art backbone generation model to complete partial templates derived from the 215 PFD proteins. For each structure, we detected fragments and removed all non-fragment regions, creating fragment-only templates. RFDiffusion then filled the missing regions, generating five candidate backbones per protein. We used each generated structure as a query in FoldSeek to retrieve the top 10 structurally similar proteins from the PDB using sequence-independent search [30].

To assess functional recovery rate, we calculated the the fraction of generated designs whose top 10 matches shared the exact Gene Ontology (GO) code(s) of the original protein. As a baseline, we performed the same evaluation with the original (unmodified) backbones.

## Results

We evaluated our fragment-based representation across four key areas: fragment detection accuracy, physicochemical properties of fragments, effectiveness in capturing functional patterns, and applications in protein search and design.

### Accurate Fragment Detection Using Combined Metrics

We validated our fragment detection algorithm using both sequence-based (BLOSUM and Sequence Identity) and angle-based (LogPr, RamRMSD, and RMSD) distance metrics. RMSD, via both PyMol and BioPython, was significantly slower and prone to silent failures during alignment.

As shown in Supp. Fig. 1, individual metrics achieved modest F1 scores (∼0.40). However, combining two complementary metrics, such as LogPr and RamRMSD, significantly improved performance to ∼0.85. Adding a third metric provided no significant improvement.

Using Receiver Operating Characteristic (ROC) analysis, we identified an optimal distance threshold of 3.65% for fragment classification, achieving an Area Under ROC (AUROC) of 87% (See Supp. Fig. 5).

### Fragment Regions Show Distinct Structural and Chemical Properties

We compared fragment and non-fragment regions across the PDBench benchmark to determine whether fragments capture distinct structural or chemical properties.

As shown in Supp. Fig. 6, fragments covered ∼40% of each protein on average with consistent standard deviations. There were some outliers like Alpha Solenoid or Alpha-Beta Horseshoe with lower coverage (∼20%) and the special folds had the highest standard deviation. Fragment coverage was consistent across resolutions (Supp. Fig. 7).

Fragment regions had a higher proportion of intra-fragment hydrogen bonds, especially in mainly $\beta$ folds with a ∼15% increase compared to non fragment regions. In contrast, inter-fragment hydrogen bonds were lower, particularly in the mainly $\alpha$ folds with a ∼ 47% decrease. Surface accessibility was also slightly reduced in fragment regions ( ∼5%) across most folds except special folds. Charge, polarity, and secondary structure distribution remained comparable between fragments and non-fragment regions (see Supp. Section 10).

[1] Intel(R) Core(TM) i9-10980XE CPU @ 3.00GHz

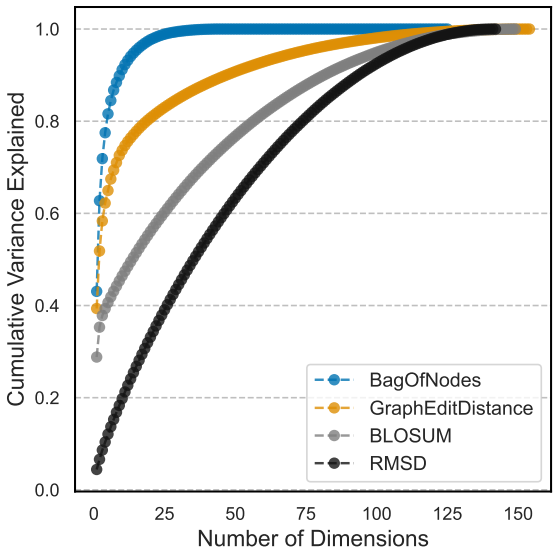



**Fig. 2.** Cumulative variance explained by different distance metrics after Principal Coordinate Analysis (PCoA) projection of the PFD containing 215 proteins across 12 functional categories.

## Fragment-Based Embeddings Efficiently Capture Functional Similarities

We evaluated how well fragment-based representations and traditional sequence- and shape-based methods capture functional similarities in reduced-dimensional embeddings. We compute a distance matrix for the PFD using RMSD (shape), BLOSUM (sequence), BagOfNodes (fragment sets), and GraphEditDistance (fragment graphs).

We projected the distances into distance-preserving latent-spaces using PCoA and calculated the cumulative explained variance across dimensions (Fig. 2). Fragment-based embeddings preserved significantly more cumulative variance than traditional metrics, with BagOfNodes and GraphEditDistance preserving over 95% and 80% within 20 dimensions, respectively. In contrast, BLOSUM and RMSD preserved less than 60% and 40%, respectively.

Despite being sequence-agnostic, fragment-based distances correlated well with sequence-based distances. GraphEditDistance achieved a Spearman correlation of 0.91 with BLOSUM, and BagOfNodes 0.57. RMSD, however, showed weak or slightly negative correlation with other metrics.

We evaluated clustering performance using GMMs and K-Means on PCoA, t-SNE, and UMAP embeddings. Table 1 summarizes the results for GMMs on PCoA embeddings. Overall, fragment-based representations demonstrated better clustering performance across most metrics.

Notably, BagOfNodes achieved the highest Silhouette score (0.8227), indicating well-separated clusters, along with the best Silhouette and Trustworthiness scores, and Distance Correlation. GraphEditDistance had the highest ARI (0.0458) and F1 score (0.1985), indicating the highest agreement with the true functional clusters after adjusting for chance. RMSD ranked highest in NMI score (0.3863), reflecting better mutual information between cluster assignments and true functions, and was second best for ARI and F1 Score. BLOSUM ranked second in Silhouette scores and Trustworthiness scores (See Supp. Table 3).

## Fragment-Based Search Combines Speed and Accuracy

We tested functional search retrieval quality and computational efficiency for fragment distance methods (GraphEditDistance and BagOfNodes) against traditional sequence (BLOSUM) and shape (RMSD) distance methods.

We selected each of the 215 annotated proteins from PFD as queries, computing the pairwise distance to all other proteins. We assessed retrieval quality using NDCG and AUROC to measure how highly functionally similar proteins ranked.

As shown in Supp. Figs. 3 and 4, all methods performed similarly overall, with scores across within one standard deviation. In terms of retrieval accuracy, fragment-based methods matched approaches for most functional categories, with particularly strong AUROC performance in identifying DNA+ATP+GTP-binding proteins (values >0.8 against 0.75 0.56 for RMSD and BLOSUM). RMSD showed an advantage in NDCG for specific functions, especially in DNA+GTP, RNA+GTP, and RNA+GTP+Metal binding searches.

Then, we benchmarked the computational efficiency of each method, measuring query times for 1, 10, and 100 queries against a database of 100 proteins using 35 cores (Table 2).

Fragment-based representations substantially reduce data dimensionality compared to traditional methods. Relative to backbone atom representation (RMSD), our fragment approach achieves dimensionality reduction of 99.1% for fragment graphs and 99.7% for fragment sets. Even compared to sequence representations, we observe significant compression: 94.4% reduction for fragment graphs and 98.3% for fragment nodes.

Fragment-based representations use significantly fewer datapoints compared to traditional methods. Relative to backbone atom-based representations (RMSD), fragment graphs and fragment sets reduce dimensionality by approximately 99.1% and 99.7%, respectively. Compared to sequence-based representations, the reductions are 94.4% and 98.3%, respectively.

Overall, sequence search with BLOSUM distance is the fastest method considering initialization time and search time. BagOfNodes is the fastest search method overall, completing 100 queries in under 0.07 s, while other methods required substantially longer – RMSD took about 1717 s, GraphEditDistance about 573 s, and BLOSUM took 36.57 s.

Although fragment-based methods have a higher initial cost which involves converting protein structures to fragment graphs (around 6 s compared to 25 s), this cost is quickly offset by the faster search times.

## Functional Design Recovery with Fragment-Based Diffusion

We evaluated the ability of fragment-based templates to guide the generation of functional proteins using RFDiffusion. For each of the 215 proteins, we generated a template backbone of fragments. We used RFDiffusion to fill in the gaps between the fragments and generate 5 different structures. Then, we use FoldSeek to search for the closest 10 backbones using sequence-independent TMAlign. For each design and for the original backbone, we define recovery rate as the fraction of backbones annotated with the function of the original backbone (See Supp. Fig. 20). We also calculate the relative recovery rate as the recovery rate of the design over the recovery rate of the original backbone (See Fig. 3 and boxplot in Supp. Fig. 21)

In general, there is a range of recovery rates, across different functional categories. Metal-binding proteins achieved

| Distance | Based On | ARI | NMI | Silhouette | Trustworthiness | Distance Corr. | F1 Score |
|---|---|---|---|---|---|---|---|
| RMSD | Shape | *0.0357* | **0.3863** | -0.0334 | 0.9598 | 0.8647 | *0.1957* |
| BLOSUM | Sequence | 0.0027 | 0.2933 | 0.0248 | *0.9923* | *0.9966* | 0.1640 |
| **GraphEditDistance (ours)** | Fragment Graph | **0.0458** | *0.3832* | *0.0766* | 0.9915 | 0.9829 | **0.1985** |
| **BagOfNodes (ours)** | Fragment Set | 0.0050 | 0.3455 | **0.8227** | **0.9991** | **0.9998** | 0.1660 |

**Table 1.** Clustering performance comparison of different distance metrics using Gaussian Mixture Models (GMM) on Principal Coordinate Analysis (PCoA) embeddings of the PFD (215 proteins across 12 functional categories).

| Distance | Init. Time (s) | 1 Query Time (s) | 10 Queries (s) | 100 Queries (s) | Memory Req. |
|---|---|---|---|---|---|
| RMSD | *5.6796* | 22.2474 | 155.3840 | 1717.0282 | 1744.36 ± 1306.90 |
| BLOSUM | **4.5215** | *0.4238* | *3.4366* | *36.5742* | 290.73 ± 217.82 |
| **GraphEditDistance (ours)** | 24.9708 | 5.6529 | 57.1725 | 573.0536 | *16.39* ± *13.57* |
| **BagOfNodes (ours)** | 24.9931 | **0.0012** | **0.0075** | **0.0686** | **4.87 ± 2.48** |

**Table 2.** Performance comparison of query methods across different protein representations. Initialization and query times (1, 10, and 100 queries) are measured on a database of 100 proteins using 35 cores. Memory requirement is reported as the average number of data points required to represent a protein: backbone atoms (RMSD), residues (BLOSUM), nodes (Fragment Graphs), or elements (Fragment Sets).

perfect recovery rates, while ATP- and GTP-binding proteins also showed consistently high recovery rates. Multi-functional proteins demonstrated more variable outcomes, with DNA+ATP+GTP-binding showing the widest range, varying from 0% to 300% relative recovery rate and also the lowest recovery rate for the control.

Single-function designs generally demonstrated higher recovery rates compared to their multi-functional counterparts. Among dual-function proteins, metal-binding combinations proved most successful, with DNA+Metal, GTP+Metal, and RNA+Metal showing particularly high recovery rates. Interestingly, some triple-function combinations achieved surprisingly high recovery rates, particularly for DNA+ATP+GTP, DNA+RNA+Metal, and RNA+GTP+Metal binding.

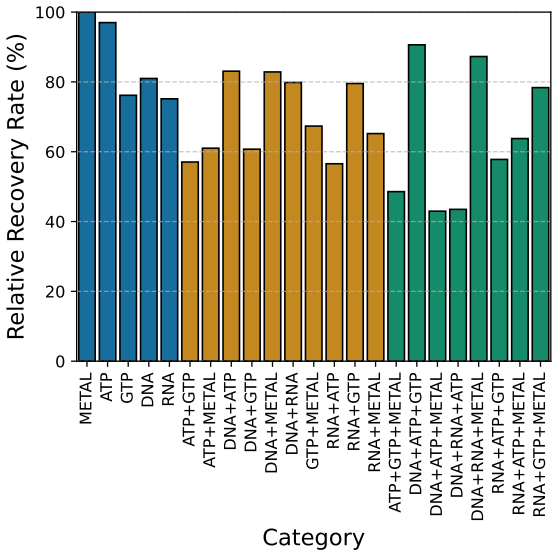



**Fig. 3.** Relative recovery rates for fragment-constrained backbone generation. The fragments from each protein in the Protein Function Dataset (PFD) were used to generate a template for RFDiffusion. The recovery rate is defined as the fraction of generated backbones whose top 10 structural matches in FoldSeek share the exact Gene Ontology (GO) function of the original protein. The relative recovery rate compares this to the recovery rate of the original protein backbone.

## Fragments and Functional Similarity

We identified two DNA-binding proteins with high sequence and shape distances but low fragment distance (Fig. 4). These proteins are the UvrABC system protein C, involved in DNA repair (PDB: 2NRR), and a viral DNA-dependent RNA polymerase (PDB: 6RIE). Despite their overall differences, they share fragments 17 (metal-binding), 23 (nucleotide-binding), and 35 (structural).

Fragment 17 is a small helix involved in metal binding, while fragment 23 is a helix-loop-sheet associated with nucleotide binding. Fragment 35, a sheet-loop-sheet motif, contributes to structural integrity. Notably, none of these fragments are explicitly classified as DNA-binding, yet their presence captures similarities in overall fold architecture. This is reflected in their low fragment distance scores compared to sequence (90% divergence) and shape (RMSD: 6.71 Å) distances.

Additionally, Graph Edit Distance (GED: 4%) accounts for the fragment neighborhood, considering factors such as the number of adjacent unknown fragments and peptide bonds.

# Discussion

In this study, we demonstrate that fragment-based representations effectively coarsen protein structures while preserving essential functional information. Using just 40 evolutionarily conserved fragments, our approach captures important structural properties while reducing the dimensionality by up to 99% compared to traditional methods. Our fast and vectorized fragment-detection algorithm allows fast conversion to fragments and achieves an F1 score of 0.85. Furthermore, we successfully use fragments to guide backbone towards preserving protein functional signatures with recovery rates between 40-100%. These results make fragment-based representations a promising alternative to traditional sequence- and structure-based approaches for protein analysis, search, and design.

## From Libraries to Detection: Building Fast and Robust Fragment Algorithms

We evaluated several metrics for fragment detection, focusing on shape and sequence. While individual metrics performed similarly, combining LogPr and RamRMSD nearly doubled the

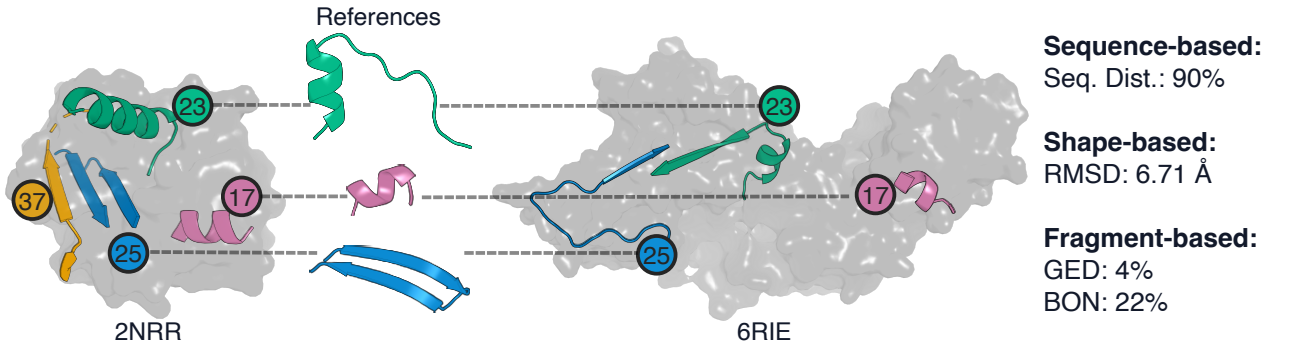



**Fig. 4.** Comparison of two structurally distinct DNA-binding proteins using sequence-, shape-, and fragment- based methods. The UvrABC system protein C, involved in DNA repair (PDB: 2NRR), is shown on the left, while a viral DNA-dependent RNA polymerase (PDB: 6RIE) is on the right. Despite significant differences in sequence and overall structure, both proteins share common functional fragments (17, 23, and 25, highlighted in color). Traditional sequence- and structure-based distances indicate high divergence between these proteins. In contrast, fragment-based metrics show a relatively low distance, suggesting a potentially shared functional role.

F1 score from 0.40 to 0.85.[19]. This highlights that torsion-angle alone can outperform sequence-based methods for robust fragment detection

While both metrics measure differences in backbone torsion angles $\phi$ and $\psi$, they process information differently. RamRMSD uses root-mean-square deviation, where squared differences are averaged, giving more weight to larger deviations. In contrast, LogPr applies a logarithmic transformation to normalized angle differences, emphasizing small deviations and converting them to a probability-like scale. This complementarity allows our algorithm to be sensitive to both, large and small deviations in torsion angles.

The fragment detection algorithm is written in Python and uses AMPAL [32] for parsing protein structures and NumPy's [16] vectorised convolutional operations. We deliberately avoided structural alignment methods based on incremental combinatorial extension (CE), which, despite potentially improving detection accuracy, proved computationally expensive and occasionally unstable during testing. [29, 9].

A key strength of our software is its flexibility. Users can easily swap our library with custom fragments by providing folders of PDB structures with the fragments of interest. This extends the software applications beyond the binding functions presented here, including enzyme design, antibody engineering, and *de novo* structural design. The software is written in Python and it is highly modular, meaning that users can expand it to integrate their own distance algorithms. Additionally, we use vectorized operations through NumPy, delivering fast performance while retaining the intuitive syntax that Python offers.

## Fragment-based Representations Capture Functional Information

Using the fold-balanced PDBench dataset, we found that fragment regions capture distinct structural and chemical properties. These regions contained higher proportions of intra-fragment hydrogen bond, particularly in mainly $\beta$ structures (+15%). This is consistent with $\beta$ folds forming hydrogen bonds between adjacent strands[27]. On the other hand, inter-fragment hydrogen bonds were significantly lower in fragment regions, with a 47% reduction in mainly $\alpha$ structures. This observation is consistent with the characteristic hydrogen bonding pattern of $\alpha$-helices, where hydrogen bonds stabilise the helical structure internally ($i$, $i+4$ pattern), reducing the potential for hydrogen bonds with adjacent fragments [27]. These results suggest that fragments may capture "self-contained" structural units. This is also supported by the reduced surface accessibility in fragment regions, such as the core of the protein, which is more likely to have folded regions, compared to surface exposed areas like loops [27].

Additionally, fragment-based representations outperform or match traditional methods in capturing functional similarities in embedding spaces. Both Fragment Graph (GraphEditDistance) and Set (BagOfNodes) metrics consistently achieved strong clustering scores, with BagOfNodes reaching a Silhouette score of 0.82 and GraphEditDistance showing the best overall performance for ARI (0.046) and F1 score (0.20). Fragment-based methods also preserved substantially more information at lower dimensions, achieving 95% and 80% cumulative variance compared to 60% and 40% for traditional sequence- and shape-based methods, respectively at 20 dimensions.

Notably, fragment-based representations capture these functional patterns without relying on the amino acid sequences. Instead they rely solely on backbone geometry. This effectiveness arises because fragments capture functional motifs regardless of their sequential arrangement, which is typical of other alignment-free analysis tools [34]. Sequence-alignment tools assume colinearity, meaning that they expect homologous residues to occur in the same order in both sequences [34]. Structural-alignment tools, such as combinatorial extension (CE), mitigate this by breaking the structure into smaller regions and reassemble them to complete the alignment [29]. However, these tools may struggle when there is little structural homology between proteins with the same function [15]. In contrast, fragment-based representations maintain performance by focusing on the presence of specific functional units. For example, Fragment Sets simply track the presence or absence of functional fragments, without their spatial precise arrangement.

## Practical Implications of Fragments for Searching Protein Databases

Protein database searches are essential tools for finding structurally or functionally similar proteins. Traditional sequence- and shape-based methods can miss important

relationships when functional motifs are arranged differently, for example when divided by other structural elements. Fragment-based representations overcome this limitation while delivering equal or better performance than traditional methods. Fragments require an initial processing cost to convert structures to graphs or sets. However, this one-time computation can be done for the entire dataset in advance and it is quickly offset by the search speedups. In our benchmarks, Fragment Sets searches using BagOfNodes execute at fractions of a second (0.07s for 100 queries), over 500× faster than sequence-based BLOSUM searches (36.57s). Similarly, Fragment Graphs searches with GraphEditDistance are ∼3× faster that structure-based searches with RMSD (573s vs. 1717s for 100 queries), but are slower than sequence searches.

The major advantage, however, is memory efficiency. Fragment representations reduce the memory requirements by 90-99%, compared to traditional representations. For large-scale applications involving millions of proteins, this reduction could enable searches on hardware that would otherwise be insufficient for atom- or residue-level comparisons. These efficiency gains, coupled with comparable functional retrieval accuracy (as measured by AUROC and NDCG scores), make fragments an attractive alternative for the next-generation of protein search tools.

### Fragment Constraints as Design Guides

The current generative tools for protein design are difficult to prompt for functional generation. For example, Ingraham et al. [18] highlight that there is currently no protein design system that can: (1) sample conditionally under diverse design constraints without retraining for new target functions, (2) with a sub-quadratic scaling computational efficiency, and (3) which integrates both sequence and structure modeling. For instance, RFDiffusion, a state-of-the-art diffusion model, lacks explicit mechanisms to enforce specific functional constraints in the generated structures.

Fragment-based constraints address this limitation by using evolutionarily conserved functional units to guide the generation process. Instead of retraining models with additional functional labels, our approach leverages evolutionarily conserved "building blocks" to steer generative models toward functionally relevant backbones.

We successfully generated functional-looking protein structures using fragments as RFDiffusion templates. Using our fragment-detection algorithm, we detected fragments in existing proteins and created partial backbones containing only these regions. We then used RFdiffusion to fill the connecting segments and used FoldSeek to retrieve the closest proteins available. On a dataset of various functional categories, we successfully generated structures that maintained the functional signatures of the original proteins. Recovery rates varied by functional category, with metal-binding and ATP-binding proteins achieving nearly perfect recovery (∼100%). Our approach was particularly effective for certain multi-functional proteins, with DNA+ATP+GTP and DNA+RNA+Metal combinations showing surprisingly high recovery rates despite their complexity.

These results suggest that diffusion models have implicitly learned about evolutionarily conserved fragments and are able to use them for design. Explicitly incorporating fragment representations in these models could help reduce the computational complexity while also providing more direct functional control to generate specific functional proteins.

### Limitations and Future Work

Our current implementation uses a curated library of 40 fragments spanning functions of DNA, RNA, GTP, ATP, and Metal binding. Further studies could explore data-driven approaches to discover novel fragments with unsupervised learning, potentially expanding the representation capacity beyond the functions described here.

A major advantage of our approach is its inherent interpretability. Unlike traditional black-box methods, fragment-based representations provide clear functional insights as they are associated with specific structural motifs and known biological roles. This interpretability could improve generative models by making their outputs more functionally interpretable and allow more control during the design process. Additionally, for protein sequence design, classifying fragments sequences instead of individual amino acid in the backbone could be faster and take into account the sequence bias defined by the fragment function.

While the BagOfNodes approach is very fast, it is less effective when multiple instances of a fragment contribute to distinct functional roles. For example, Zinc finger proteins usually contain three instances of fragment 14, each binding a positively-charged Zinc ion, and all binding negatively-charged DNA (See Fig. 1). More instances of fragment 14 may indicate binding to multiple DNA strands or different regions of the same strand. In these cases, Fragment Graphs with GraphEditDistance provide a more nuanced representation by capturing the connectivity and fragment context, despite their higher computational cost.

Our fragment detection algorithm achieves a good F1 score of 0.85, but is potentially sensitive to subtle variations in torsion angles which could lead to misclassification. For example, a large change in torsion angles of the middle amino acid of a fragment would change the backbone angles for one amino acid only, so it might still be classified similarly. Future work could integrate probabilistic models to quantify detection confidence and providing adjustable sensitivity, allowing users to choose the settings based on their design scenario.

Beyond protein design, fragment-based representations improve biosecurity applications by identifying potentially hazardous structural motifs for that might escape detection in sequence- or structure- based screening systems. By recognizing functional fragments, regardless of their arrangement in the proteins, our approach could provide an additional layer of safety for protein synthesis services.

## Conclusion

We introduced a fragment-based protein representation that encodes structures using a curated library of 40 evolutionarily conserved functional fragments. This approach reduces dimensionality by up to 99% while preserving functional and structural information. Our evaluations demonstrate that fragment-based representations capture functional relationships more effectively than traditional methods in clustering, enable significantly faster database searches with comparable accuracy, and successfully guide RFDiffusion to generate backbones with functional signatures. Unlike black-box representations, our method provides interpretability by linking fragments to biological functions. Fragment-based representations offer a scalable and biologically relevant framework for protein design. By balancing efficiency and interpretability, this approach lays the foundation for the next generation of protein design tools.

## Competing interests

No competing interest is declared.

## Author contributions statement

All authors were involved in the conception of the project and the writing of the manuscript. CWW and KS supervised the work. LVC developed the code, ran the experiments, and produced the figures.

## Acknowledgments

LVC thanks Lorenzo Pisani for his valuable guidance in developing the protein fragment library software.

## References

1. Sarah Alamdari, Nitya Thakkar, Rianne Van Den Berg, Neil Tenenholtz, Robert Strome, Alan M. Moses, Alex X. Lu, Nicolò Fusi, Ava P. Amini, and Kevin K. Yang. Protein generation with evolutionary diffusion: Sequence is all you need, September 2023.
2. Vikram Alva and Andrei N Lupas. From ancestral peptides to designed proteins. *Current Opinion in Structural Biology*, 48:103–109, February 2018.
3. Vikram Alva, Johannes Söding, and Andrei N Lupas. A vocabulary of ancient peptides at the origin of folded proteins. *eLife*, 4:e09410, December 2015.
4. Michael Ashburner, Catherine A. Ball, Judith A. Blake, David Botstein, Heather Butler, J. Michael Cherry, Allan P. Davis, Kara Dolinski, Selina S. Dwight, Janan T. Eppig, Midori A. Harris, David P. Hill, Laurie Issel-Tarver, Andrew Kasarskis, Suzanna Lewis, John C. Matese, Joel E. Richardson, Martin Ringwald, Gerald M. Rubin, and Gavin Sherlock. Gene Ontology: Tool for the unification of biology. *Nature Genetics*, 25(1):25–29, May 2000.
5. H. M. Berman. The Protein Data Bank. *Nucleic Acids Research*, 28(1):235–242, January 2000.
6. Alice Bizeul, Thomas Sutter, Alain Ryser, Bernhard Schölkopf, Julius von Kügelgen, and Julia E. Vogt. From Pixels to Components: Eigenvector Masking for Visual Representation Learning, 2025.
7. Leonardo V Castorina, Rokas Petrenas, Kartic Subr, and Christopher W Wood. PDBench: Evaluating computational methods for protein-sequence design. *Bioinformatics*, 39(1):btad027, January 2023.
8. Leonardo V Castorina, Suleyman Mert Ünal, Kartic Subr, and Christopher W Wood. TIMED-Design: Flexible and accessible protein sequence design with convolutional neural networks. *Protein Engineering, Design and Selection*, 37:gzae002, January 2024.
9. Peter J. A. Cock, Tiago Antao, Jeffrey T. Chang, Brad A. Chapman, Cymon J. Cox, Andrew Dalke, Iddo Friedberg, Thomas Hamelryck, Frank Kauff, Bartek Wilczynski, and Michiel J. L. de Hoon. Biopython: Freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423, June 2009.
10. J. Dauparas, I. Anishchenko, N. Bennett, H. Bai, R. J. Ragotte, L. F. Milles, B. I. M. Wicky, A. Courbet, R. J. de Haas, N. Bethel, P. J. Y. Leung, T. F. Huddy, S. Pellock, D. Tischer, F. Chan, B. Koepnick, H. Nguyen, A. Kang, B. Sankaran, A. K. Bera, N. P. King, and D. Baker. Robust deep learning based protein sequence design using ProteinMPNN, June 2022.
11. Noelia Ferruz, Steffen Schmidt, and Birte Höcker. ProtGPT2 is a deep unsupervised language model for protein design. *Nature Communications*, 13(1):4348, July 2022.
12. Andreas Fischer, Kaspar Riesen, and Horst Bunke. Improved quadratic time approximation of graph edit distance by combining Hausdorff matching and greedy assignment. *Pattern Recognition Letters*, 87:55–62, February 2017.
13. Vincent Frappier, Justin M. Jenson, Jianfu Zhou, Gevorg Grigoryan, and Amy E. Keating. Tertiary Structural Motif Sequence Statistics Enable Facile Prediction and Design of Peptides that Bind Anti-apoptotic Bfl-1 and Mcl-1. *Structure*, 27(4):606–617.e5, April 2019.
14. P. Gainza, F. Sverrisson, F. Monti, E. Rodolà, D. Boscaini, M. M. Bronstein, and B. E. Correia. Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. *Nature Methods*, 17(2):184–192, February 2020.
15. Tymor Hamamsy, James T. Morton, Robert Blackwell, Daniel Berenberg, Nicholas Carriero, Vladimir Gligorijevic, Charlie E. M. Strauss, Julia Koehler Leman, Kyunghyun Cho, and Richard Bonneau. Protein remote homology detection and structural alignment using deep learning. *Nature Biotechnology*, 42(6):975–985, June 2024.
16. Charles R. Harris, K. Jarrod Millman, Stéfan J. Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. Van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández Del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.
17. S Henikoff and J G Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919, November 1992.
18. John B. Ingraham, Max Baranov, Zak Costello, Karl W. Barber, Wujie Wang, Ahmed Ismail, Vincent Frappier, Dana M. Lord, Christopher Ng-Thow-Hing, Erik R. Van Vlack, Shan Tie, Vincent Xue, Sarah C. Cowles, Alan Leung, João V. Rodrigues, Claudio L. Morales-Perez, Alex M. Ayoub, Robin Green, Katherine Puentes, Frank Oplinger, Nishant V. Panwar, Fritz Obermeyer, Adam R. Root, Andrew L. Beam, Frank J. Poelwijk, and Gevorg Grigoryan. Illuminating protein space with a programmable generative model. *Nature*, 623(7989):1070–1078, November 2023.
19. Sunghoon Jung, Se-Eun Bae, Insung Ahn, and Hyeon S. Son. Protein Backbone Torsion Angle-Based Structure Comparison and Secondary Structure Database Web Server. *Genomics & Informatics*, 11(3):155, 2013.
20. Rachel Kolodny, Sergey Nepomnyachiy, Dan S Tawfik, and Nir Ben-Tal. Bridging Themes: Short Protein Segments Found in Different Architectures. *Molecular Biology and Evolution*, 38(6):2191–2208, May 2021.
21. Craig O Mackenzie and Gevorg Grigoryan. Protein structural motifs in prediction and design. *Current Opinion in Structural Biology*, 44:161–167, June 2017.

22. Fabrizio Mastrolorito, Fulvio Ciriaco, Maria Vittoria Togo, Nicola Gambacorta, Daniela Trisciuzzi, Cosimo Damiano Altomare, Nicola Amoroso, Francesca Grisoni, and Orazio Nicolotti. fragSMILES as a chemical string notation for advanced fragment and chirality representation. *Communications Chemistry*, 8(1):26, January 2025.
23. Geraldene Munsamy, Ramiro Illanes-Vicioso, Silvia Funcillo, Ioanna T. Nakou, Sebastian Lindner, Gavin Ayres, Lesley S. Sheehan, Steven Moss, Ulrich Eckhard, Philipp Lorenz, and Noelia Ferruz. Conditional language models enable the efficient design of proficient enzymes, May 2024.
24. Yifei Qi and John Z. H. Zhang. DenseCPD: Improving the Accuracy of Neural-Network-Based Computational Protein Sequence Design with DenseNet. *Journal of Chemical Information and Modeling*, 60(3):1245–1252, March 2020.
25. Jane S. Richardson. Early ribbon drawings of proteins. *Nature Structural Biology*, 7(8):624–625, August 2000.
26. Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, and Rob Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences of the United States of America*, 2019.
27. Georg E. Schulz and R. Heiner Schirmer. *Principles of Protein Structure*. Springer Advanced Texts in Chemistry. Springer New York, New York, NY, 1979.
28. Fabian Sesterhenn, Che Yang, Jaume Bonet, Johannes T. Cramer, Xiaolin Wen, Yimeng Wang, Chi-I Chiang, Luciano A. Abriata, Iga Kucharska, Giacomo Castoro, Sabrina S. Vollers, Marie Galloux, Elie Dheilly, Stéphane Rosset, Patricia Corthésy, Sandrine Georgeon, Mélanie Villard, Charles-Adrien Richard, Delphyne Descamps, Teresa Delgado, Elisa Oricchio, Marie-Anne Rameix-Welti, Vicente Más, Sean Ervin, Jean-François Eléouët, Sabine Riffault, John T. Bates, Jean-Philippe Julien, Yuxing Li, Theodore Jardetzky, Thomas Krey, and Bruno E. Correia. De novo protein design enables the precise induction of RSV-neutralizing antibodies. *Science*, 368(6492):eaay5051, May 2020.
29. Ilya N Shindyalov and Philip E Bourne. Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. *Protein engineering*, 11(9):739–747, 1998.
30. Michel Van Kempen, Stephanie S. Kim, Charlotte Tumescheit, Milot Mirdita, Jeongjae Lee, Cameron L. M. Gilchrist, Johannes Söding, and Martin Steinegger. Fast and accurate protein structure search with Foldseek. *Nature Biotechnology*, 42(2):243–246, February 2024.
31. Joseph L. Watson, David Juergens, Nathaniel R. Bennett, Brian L. Trippe, Jason Yim, Helen E. Eisenach, Woody Ahern, Andrew J. Borst, Robert J. Ragotte, Lukas F. Milles, Basile I. M. Wicky, Nikita Hanikel, Samuel J. Pellock, Alexis Courbet, William Sheffler, Jue Wang, Preetham Venkatesh, Isaac Sappington, Susana Vázquez Torres, Anna Lauko, Valentin De Bortoli, Emile Mathieu, Sergey Ovchinnikov, Regina Barzilay, Tommi S. Jaakkola, Frank DiMaio, Minkyung Baek, and David Baker. De novo design of protein structure and function with RFdiffusion. *Nature*, 620(7976):1089–1100, August 2023.
32. Christopher W Wood, Jack W Heal, Andrew R Thomson, Gail J Bartlett, Amaurys Á Ibarra, R Leo Brady, Richard B Sessions, and Derek N Woolfson. ISAMBARD: An open-source computational environment for biomolecular analysis, modelling and design. *Bioinformatics*, 33(19):3043–3050, October 2017.
33. Andy Hsien-Wei Yeh, Christoffer Norn, Yakov Kipnis, Doug Tischer, Samuel J. Pellock, Declan Evans, Pengchen Ma, Gyu Rie Lee, Jason Z. Zhang, Ivan Anishchenko, Brian Coventry, Longxing Cao, Justas Dauparas, Samer Halabiya, Michelle DeWitt, Lauren Carter, K. N. Houk, and David Baker. De novo design of luciferases using deep learning. *Nature*, 614(7949):774–780, February 2023.
34. Andrzej Zielezinski, Susana Vinga, Jonas Almeida, and Wojciech M. Karlowski. Alignment-free sequence comparison: Benefits, applications, and tools. *Genome Biology*, 18(1):186, December 2017.

# Chapter 6

# Conclusion

## 6.1 Summary of Contributions

This dissertation explored several tools for protein design, starting with benchmarking, moving to developing new models, and finally challenging traditional representations for protein structures.

In Chapter 3, we introduced PDBench, a software toolkit and benchmark dataset designed to evaluate protein sequence design methods. The rapid development of deep learning models in protein design created a need for standardised and biologically meaningful benchmarks. PDBench addresses this gap with a curated set of 595 protein structures across 40 distinct architectures. Additionally, we proposed biologically relevant evaluation metrics, such as prediction bias, allowing for more nuanced performance across different folds and structural resolution. This revealed significant fold-specific strengths and limitations in various protein design methods and was crucial in guiding the development of our TIMED model.

Then, in Chapter 4, we introduced TIMED, a CNN-based model for protein sequence design, with its two flavours of TIMED-Charge and TIMED-Polar. We showed that voxel-based representations can incorporate design constraints such as charge and polarity, allowing designers to specify these constraints during inference. To improve accessibility, we developed and open-sourced Aposteriori, a voxelisation toolkit, and TIMED-Design, a user-friendly interface with both web and command-line components. We benchmarked our models against state-of-the-art sequence design models using PDBench, demonstrating competitive performance across protein folds. Furthermore, we retrained and released the model weights for all the tested models. We validated TIMED in the Adaptyv Bio competition, one submission reaching the top 3% of entries

with a Kd in the $\mu M$ range, and all ten of our submissions achieving high expression levels.

Finally, in Chapter 5, we introduced a new representation for protein structures based on fragments, evolutionarily conserved and functional structural motifs. Rather than using atom-based or amino acid-based representations, we abstracted several groups of residues into fragments in either a Fragment Set or Fragment Graph. This approach significantly reduces the dimensionality of the data by up to 99%, while preserving functional information. We further tested fragment-based representation for functional clustering and database searches, showing significant speed-ups and equal or better accuracy compared to traditional representations. Additionally, we used fragments to condition RFDiffusion, guiding the generation of functionally plausible backbones.

## 6.2 Limitations and Future Work

The PDB is the cornerstone of structural biology. However, inherent biases exist due to the types of organisms studied and most commonly crystallised proteins. These biases lead to an over-representation of the specific folds which, if left unaddressed, will skew performance towards specific folds. We partly addressed this with PDBench, with a balanced test set of structures covering the architectures identified in the CATH dataset. This gives a more nuanced view on performance and identifies prediction biases and pitfalls of models. While this addresses the evaluation stage, protein sequence design models often trained on most of the available training data, increasing the risk of data leakage and potentially memorising rather than generalising.

In the TIMED paper, we addressed this issue by retraining all models on PISCES, a culled dataset of proteins with non-redundant examples of folds available in nature. However, a more robust approach for *de novo* design could involve crafting minimal datasets that explicitly ensure no structural or sequence similarity between test, validation, and train sets. This would allow for predicted sequences with lower similarity to known ones for *de novo* proteins.

The astronomical design sequence space presents both opportunities and challenges for both types of models. Models trained on most of the available data like ProteinMPNN perform impressively on sequence recovery benchmarks, but this approach may bias designs towards sequences similar to those already observed in databases. We see evidence of this in the Adaptyv Bio competition, where Kd strongly correlates with similarity to known sequences or structures in the PDB. Meaning that, models trained

on larger datasets answer a fundamentally different question: reconstructing sequences towards known proteins rather than designing novel ones. Depending on the type of design challenge, these models can still be useful. Future models could be trained on both types of dataset and allow the user to modulate the "creativity" and "precision" required in their design, similar to the temperature settings in LLMs.

A key limitation of CNNs is that frames are predicted independently. In contrast, GNNs, predict autoregressively, taking into account the context of previously predicted amino acids. Future work could explore hybrid GNN-CNN architectures, leveraging CNNs for local interactions, while incorporating GNNs for global interactions and autoregressive predictions, leading to improved and cohesive designed sequences.

Using fragment-based representations, we took a fundamentally different approach to tackle the increasing size and complexity of protein design models. Instead of representing proteins at the atomic or residue level, we represent them as fragments, significantly reducing the dimensionality while maintaining the functional signatures of the protein. Tools and models built on fragment-based representations could be substantially smaller, faster, and inherently simpler to interpret.

However, several aspects were arbitrarily defined, such as the amount of overlap between classified fragments and the classification of unknown connectors or unknown fragments. Future work could involve a data-driven approach, using the PDB to derive statistical rules on how fragments are positioned in protein structure, either top-down with machine learning models or bottom-up using principles of protein structures. Furthermore, our current fragment library is relatively small, and does not capture the full functional spectrum of proteins. Future work could explore different libraries of fragments, whether learned with unsupervised approaches or using pre-compiled libraries such as THEMES [70] or TERMs [51].

Ultimately, I believe that progress in the protein design field will depend on the accessibility and openness of software. I hope that, with my small contribution I have pushed the field **Towards Efficient and Accessible Tools for Protein Design**.

I cannot wait to see what is next for proteins: "From so simple a beginning, endless forms most beautiful and most wonderful have been, and are being,"[34] designed.

# Appendix A

# Supplementary Materials

## A.1 PDBench

# PDBench: Evaluating Computational Methods for Protein Sequence Design

Supplementary Materials

## 1 Benchmark Details

### 1.1 TS500

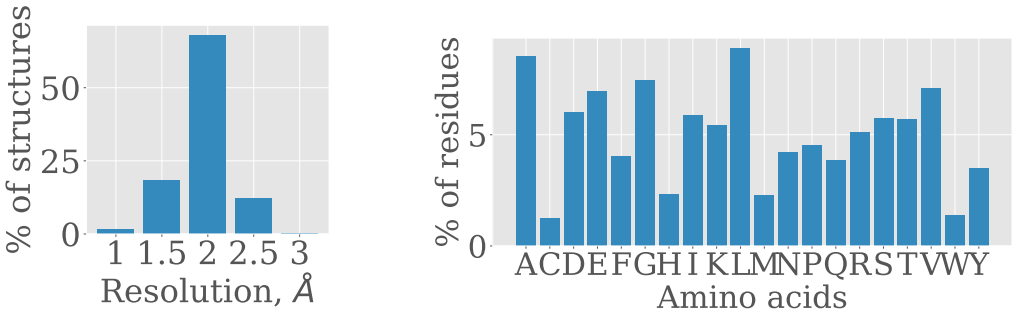



Figure 1: Resolution and amino acid distribution in the TS500 set

### 1.2 PDBench (ours)

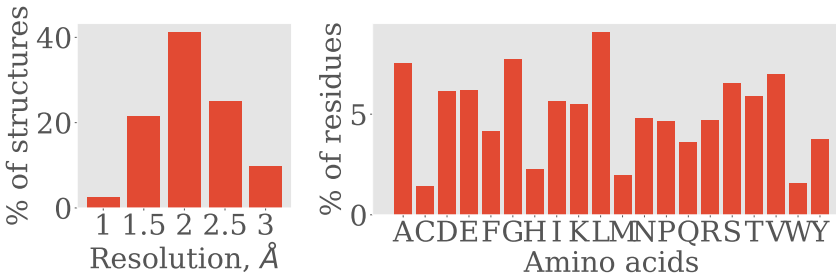



Figure 2: Resolution and amino acid distribution in the PDBench set.

## 2 Benchmark Results

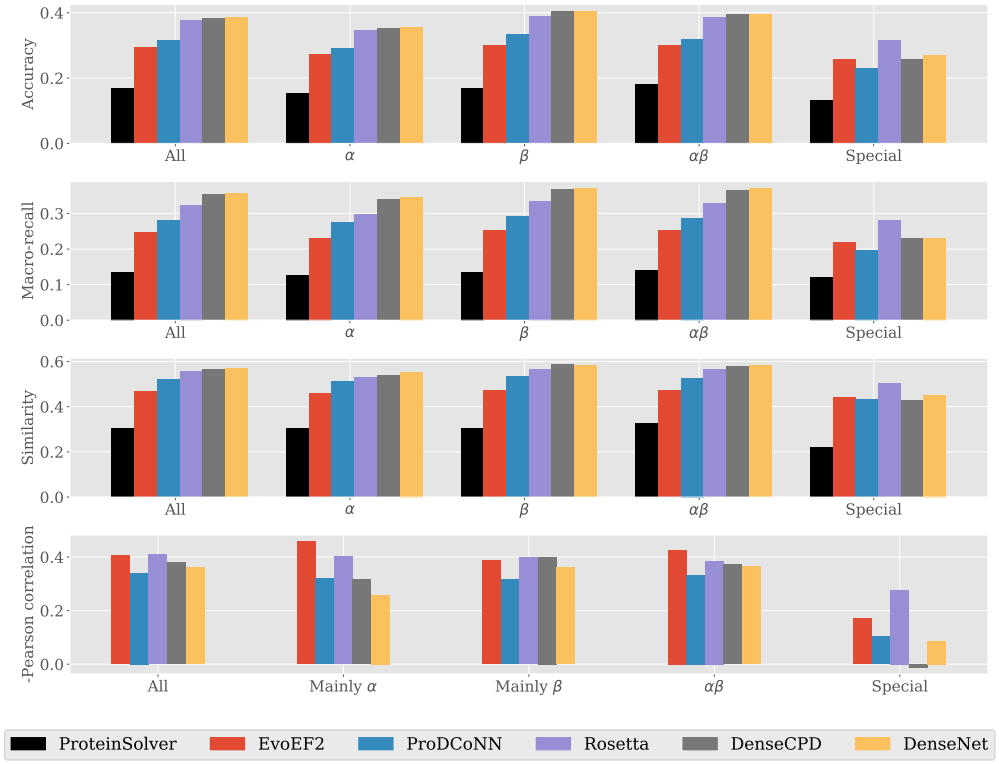



Figure 3: The first two plots compare performances of models across classes of folds. The plot at the bottom shows the negated correlation coefficient (Y axis) between macro-recall and the resolution (in Å) of the input structure. All p-values were significant ($< 10^{-8}$) except for ProteinSolver (0.3) which is excluded from the plot.

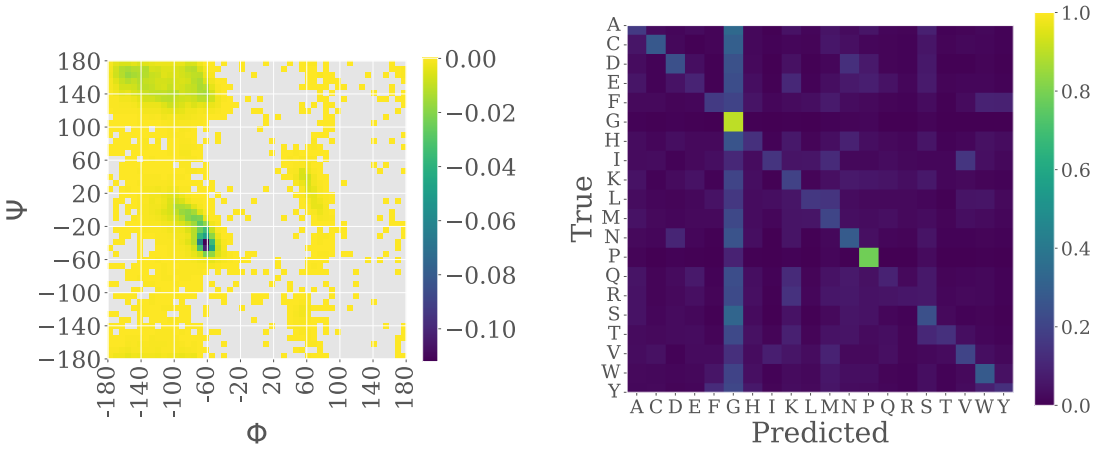



Figure 4: Φ and Ψ plots to explore Glycine overprediction. Left: A torsion angle plot showing normalized frequency difference between true and predicted number of glycine amino acids. Negative values indicate increased glycine frequency in predicted sequences. Right: A confusion matrix showing the confusion frequency for amino acid pairs.

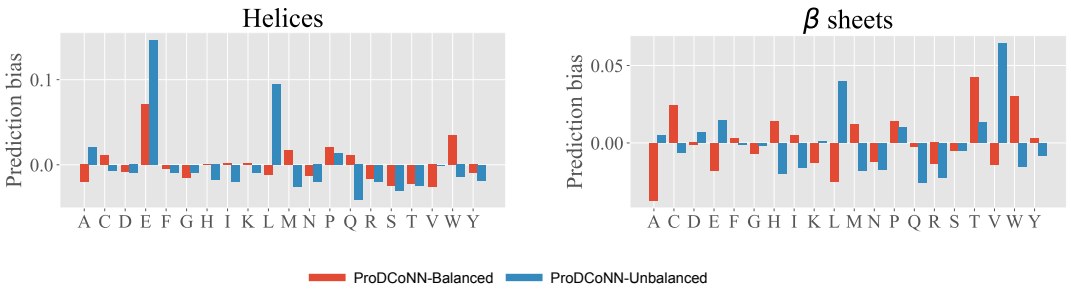



Figure 5: Prediction bias comparison for ProDCoNN-Balanced (red) and ProDCoNN-Unbalanced models across all the structures in the benchmark and for each type of residue. The left plot represents bias on $\alpha$-helical structures, while the plot on the right is for $\beta$-sheets. Prediction bias is calculated as deviation of the predictions from the real frequency of residues in the benchmark structures.

# 3 Input Data Pipeline

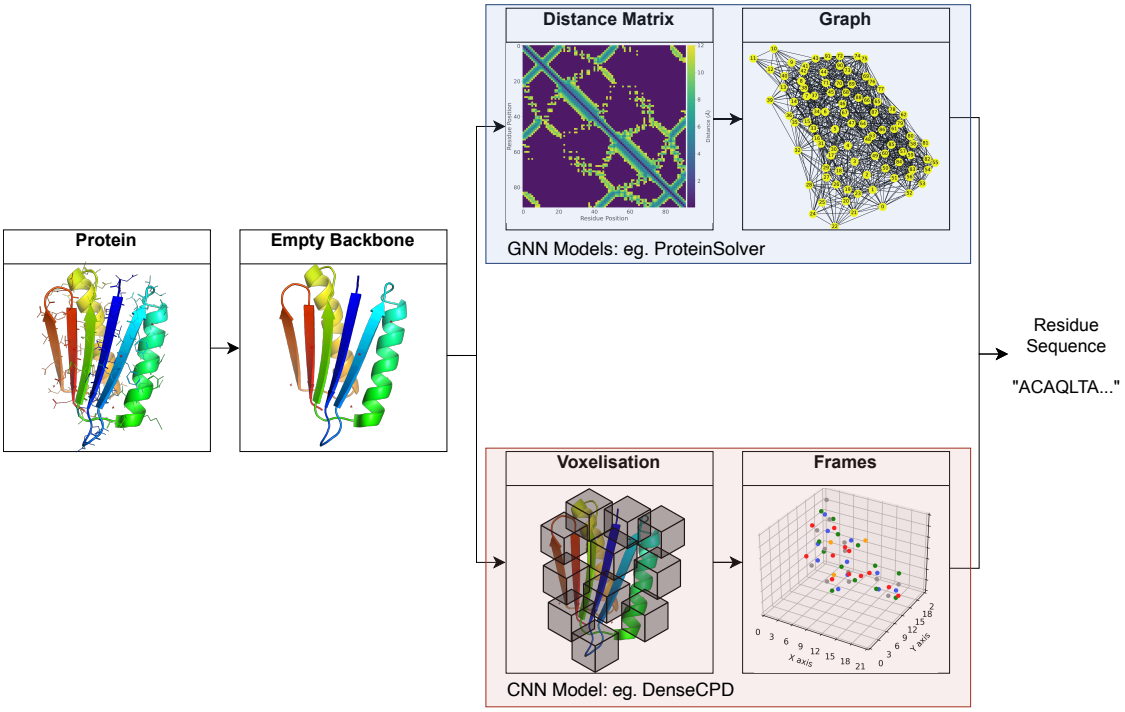



Figure 6: Illustration of the data pipeline. High-quality 3D structures of proteins are obtained from a database. The side-chains of each residues are removed so to produce an empty backbone. The GNN models calculate the distances between each residue in the protein to produce a distance matrix which is used for the production of a graph. The CNN model, on the other hand, voxelises areas of space ("frame") around each residue, with the C$\alpha$ at the center of it. Both models predict the identity of the side-chains of the residues giving a sequence of predicted residues to obtain the input 3D structure.

# 4 ProteinSolver Distance Matrix with and without Poly-Glycine Input

Protein Solver Distance Matrices using 1QYS protein with and without sidechain atoms (labels). ProteinSolver claims to use the heaviest atom in the residue and as confirmed by private conversation received on June 5th at 20:59, they "include both backbone **and side chain atoms** when calculating nearest distances".

The side chains atoms determine the identify of the amino acids and are therefore labels. In a truly *de novo* design setting, only the backbone structure (without side chains) would be available.

The macro-recall performance dropped from 36.6 to 13.3 when using an empty backbone (poly-glycine) input. The performance drop is only observed for ProteinSolver while it remains constant for all other models.

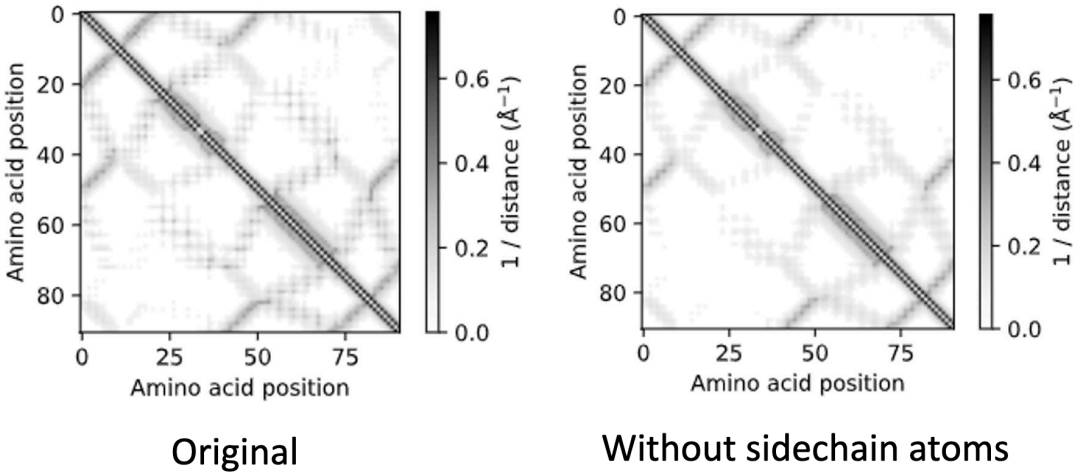



Figure 7: The distance matrix changes if side-chains (labels) are included in the .pdb input. The two distance matrices should be identical as only the empty backbone atoms should be used, as they are the only ones available in *de novo* settings, meaning it probably presents label leakage.

# 5 Physics Models Commands

We evaluated two state-of-the-art physics-based methods: **EvoEF2** (Huang, Pearce, and Zhang 2019) and **Rosetta** (Alford et al. 2017). We used the following commands to run fixed backbone design protocols:

EvoEF2: ./EvoEF2 –command=ProteinDesign –ppint –design_chains=B –pdb=structure.pdb to design chain B from structure.pdb.

Rosetta (version 3.12): ./fixbb.static.linuxgccrelease -s structure.pdb -linmem_ig 10 -ignore_unrecognized_res -resfile file.txt. Resfile was used to select specified chain.

# 6 Benchmark structures

Our benchmark set contains 595 protein structures spanning 40 protein architectures.

**PDB code + chain:** 1xg0C, 3g3zA, 3rf0A, 4i5jA, 2ptrA, 3f0cA, 4a5uB, 2p57A, 2q0oC, 6er6A, 1h32A, 3e3vA, 3cxbA, 1dvoA, 5dicA, 2bnmA, 4pfoA, 2ebfX, 3giaA, 1a41A, 3cexA, 4ebbA, 3jrtA, 3wfdB, 4v1gA, 3qb9A, 3abhA, 3nvoA, 2o1kA, 5x56A, 2ra1A, 4adzA, 2p6vA, 3k4iA, 4lctA, 4adyA, 4zhbA, 4p6zG, 4nq0A, 3dadA, 2vq2A, 4dloA, 2of3A, 4y5jA, 2pm7A, 2hr2A, 3ro3A, 3bqoA, 3ut4A, 2yhcA, 4k6jA, 3iisM, 5agdA, 2fbaA, 3e7jA, 1v7wA, 3a0oA, 4wu0A, 4ozwA, 4cj0A, 1gxmA, 5m7yA, 4fnvA, 5gzkA, 4ayoA, 3wkgA, 3vsnA, 2jg0A, 4j5tA, 4ktpA, 4mqwA, 5lf2A, 5mriA, 5ol4B, 1bx7A, 3ca7A, 3tvjA, 3tbdA, 1uzkA, 5bq8A, 3klkA, 1b8kA, 1v6pA, 4hquA, 4k8wA, 6a2qA, 1lpbA, 3hrzB, 6fmeB, 2aydA, 2ra8A, 4fzqA, 3d4uB, 3wwlA, 2r01A, 1lslA, 3f3fC, 2q4zA, 2de6A, 3d9xA, 2hjeA, 3mcbB, 2y8nB, 3witA, 1ya5T, 2dyiA, 3kyfA, 2v76A, 2e12A, 1g3pA, 4o06A, 3fb9A, 2p38A, 1igqA, 4hhvA, 3teeA, 5j3tA, 5h3xA, 3zbdA, 5d7uA, 5zcjC, 5u1mA, 1wthD, 4rg1A, 1kt6A, 2ja9A, 1i4uA, 4i86A, 1o7iA, 1x8qA, 2ichA, 3dzmA, 3n91A, 1luzA, 4lqzA, 4i1kA, 5xlyB, 3a35A, 3tdqA, 4mxtA, 3wjtA, 3buuA, 3ksnA, 2w7qA, 2yzyA, 4z48A, 3bk5A, 4qa8A, 2byoA, 3bmzA, 4egdA, 4joxA, 3h6jA, 2bhuA, 1pmhX, 6ggrA, 4dqaA, 4v2bA, 4weeA, 2w07B, 4r9pA, 2r2cA, 2r0hA, 4aqoA, 4luqC, 3iagC, 1k5nA, 2ygnA, 3bwzA, 4fmrA, 1njhA, 4hi6A, 1pkhA, 1gp0A, 3q1nA, 2ag4A, 2v3iA, 3ty1A, 1gprA, 3aihA, 4c4aA, 1tulA, 4a02A, 4c08A, 4maiA, 1jovA, 3wmvA, 2fdbM, 1dqgA, 1xzzA, 6i18A, 4i4oA, 4efpA, 5yh4A, 3h6qA, 5bowA, 5vi4A, 2vxtI, 3vwcA, 4lo0C, 1sr4C, 2dpfA, 3dzwA, 3a0eA, 1xd5A, 4h3oA, 4tkcA, 5j76A, 3mezC, 4gc1A, 1b2pA, 4le7A, 4oitA, 6b0gE, 1z1yB, 1vmoA, 2gudA, 4r6rE, 5krpC, 5v6fA, 4pitA, 6flwA, 4ddnD, 3apaA, 5gvyA, 1c3mA, 4mq0A, 3wocA, 3aqgA, 3towA, 2qp2A, 1nykA, 2bmoA, 2gbwA, 1rfsA, 4aivA, 3gkeA, 2nwfA, 1jm1A, 2qpzA, 5cxmB, 3dqyA, 3d89A, 2b1xA, 4qdcA, 2q3wA, 3c7xA, 1genA, 1itvA, 3s18A, 4rt6B, 3cu9A, 3wasA, 6ms3B, 6frwA, 3k1uA, 5aycA, 5c0pA, 4n1iA, 3r4zA, 1tl2A, 4u6dA, 1oygA, 4qqsA, 3qz4A, 5a8cA, 4pvaA, 3kstA, 5flwA, 6gy5A, 1cruA, 1suuA, 3o4pA, 2p4oA, 4mzaA, 5gtqA, 3dr2A, 3dasA, 3g4eA, 2fp8A, 5hx0B, 1npeA, 1s1dA, 2zwaA, 3a72A, 2zb6A, 3scyA, 3b7fA, 3al9A, 3o4hA, 4pxwA, 4wk0A, 2w18A, 5em2A, 1sq9A, 1xipA, 4h5iA, 1jofA, 5ic7A, 5k19A, 6e1zA, 2z2nA, 6e4lA, 6fkwA, 6damA, 1flgA, 4cvbA, 4mh1A, 1z68A, 2z3zA, 4q1vA, 1xfdA, 5d7wA, 1kapP, 3laaA, 1p9hA, 3ultA, 3s6lA, 2xqhA, 4dt5A, 5m5zA, 5lw3A, 1k5cA, 1k4zA, 2ntpA, 3bh7B, 2j8kA, 2vfoA, 3n6zA, 1hf2A, 2x3hB, 1rmgA, 6mfkA, 1l0sA, 2xt2A, 5nzgA, 3kweA, 2w7zA, 1lktA, 3facA, 3pyiB, 2casA, 1gppA, 3maoA, 1ut7A, 1hxrA, 1t61A, 4qjvA, 3lywA, 3dalA, 5hqhA, 3u7zA, 3r90A, 1tp6A, 3s9xA, 2ex5A, 3gbyA, 5kvbA, 2cu3A, 1c1yB, 5f6rA, 4a6qA, 2w56A, 4lqbA, 4oobA, 3oajA, 3n8bA, 3jumA, 2prxA, 5b1rA, 1ewfA, 4m4dA, 2obdA, 6baqA, 1usuB, 3e8tA, 3aotA, 2rckA, 3l6iA, 3uv1A, 3bqwA, 5mprA, 1kkoA, 4cd8A, 1vd6A, 2g0wA, 4lanA, 3s83A, 2v3gA, 3fkrA, 4z0gA, 3sggA, 5zjbA, 2xfrA, 4g8tA, 5n6fA, 1muwA, 2qhqA, 3h35A, 3kluA, 3fn2A, 2od6A, 1kcfA, 3nlcA, 2zw2A, 4ftxA, 3u2aA, 2hiqA, 1xkpC, 6ih0A, 5c12A, 1w4rA, 3c0fB, 3nbmA, 2r6zA, 5hxdA, 1chdA, 3do8A, 3gohA, 1n0eA, 2q82A, 5kxhA, 3oqiA, 2x4lA, 3d3kA, 3l46A, 2fkcA, 5jphA, 3nytA, 3rhtA, 3dkrA, 2psbA, 1tc5A, 3vrdB, 2je3A, 3g5sA, 1jkeA, 4at0A, 1vi4A, 4u8pC, 4ntcA, 5ipyA, 5nakA, 4z24B, 4opcA, 5cdkA, 2b0aA, 4n2pA, 1j5uA, 1vzyB, 1vq0A, 4ipuA, 4dq9A, 4jtmA, 3gs9A, 3adyA, 3mi0A, 3ib7A, 3g91A, 1vr7A, 4zx2A, 1ds1A, 3zwfA, 1hq0A, 3hbcA, 3p8kA, 1wraA, 3t91A, 3c9fA, 2imhA, 1um0A, 5y0mA, 5u4hA, 3zh4A, 3swgA, 5ujsA, 3nvsA, 2o0bA, 2pqcA, 3slhA, 1rf6A, 4n3pA, 5bufA, 3rmtA, 4fqdA, 1ud9A, 1t6lA, 1rwzA, 3ifvA, 1iz5A, 3lx2A, 1u7bA, 5tupA, 5h0tA, 5v7mA, 3fdsC, 3aizA, 1b77A, 3p91A, 1dmlA, 3hslX, 2z0lA, 6nibA, 2jerA, 1xknA, 1zbrA, 3hvmA, 1jdwA, 5wpiA, 1g61A, 1h70A, 5m3qA, 1ynfA, 3wn4A, 1io0A, 4rcaB, 4fcgA, 4ecoA, 3wpcA, 4im6A, 4cnmA, 5hzlB, 4fs7A, 2xwtC, 3e4gA, 4wp6A, 5il7A, 1z7xW, 4u7lA, 6fg8A, 2wfhA, 2fy7A, 5wwdA, 1j3aA, 1omzA, 3emfA, 1xw3A, 3h4rA, 3essA, 1o22A, 4ktbA, 1jh6A, 3n08A, 5tsqA, 3e9vA, 4j7hA, 1i4jA, 2wnfA, 3v1aA, 3coqA, 2f60K, 4zgmA, 1i7wB, 6g6kA, 1pbyC, 1a92A, 3alrA, 2wjvD, 2a26A, 1devB, 4l0nA, 4ayaA, 3zxcA, 4pkfB, 2b1yA, 4dncD, 4jpnA, 4e18B, 3vepX, 3v4yB, 1xawA, 1ykhA, 2p64A, 6bscB, 2z3xA, 4uzzB, 3thfA, 1wq6A, 4ke2A, 4lhfA, 2v66B, 3lczA, 2h4oA, 4wjwA, 3kvpA, 3e56A, 3bk3C, 2ds5A, 3zoqB, 3nfgB, 4ksnA, 3ua0A, 3nrtA, 4a9aC, 6hikL

# 7 DenseCPD Architecture (claimed to be DenseNet)

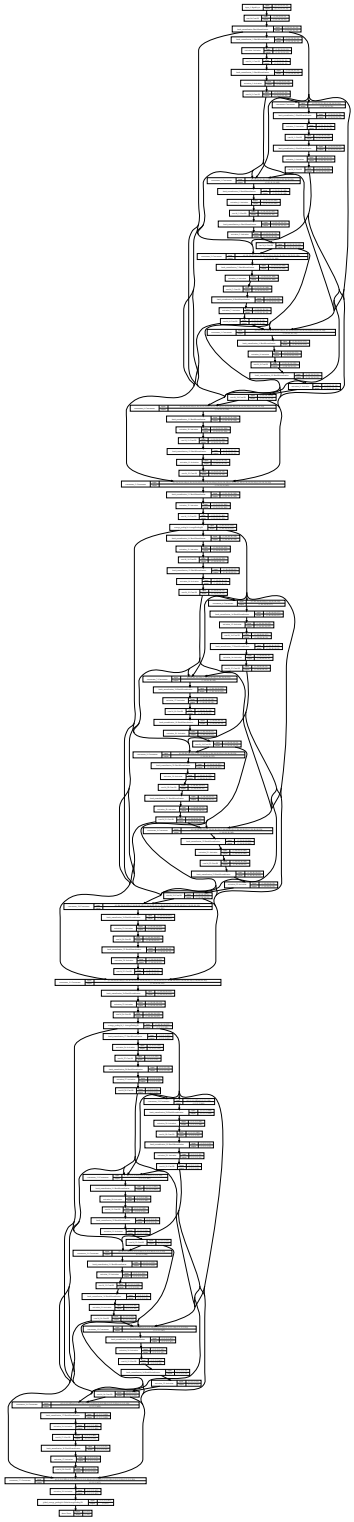


Figure 8: Implementation of "DenseNet" architecture based on the DenseCPD from the paper figure.

# 8 DenseCPD Architecture (actual)

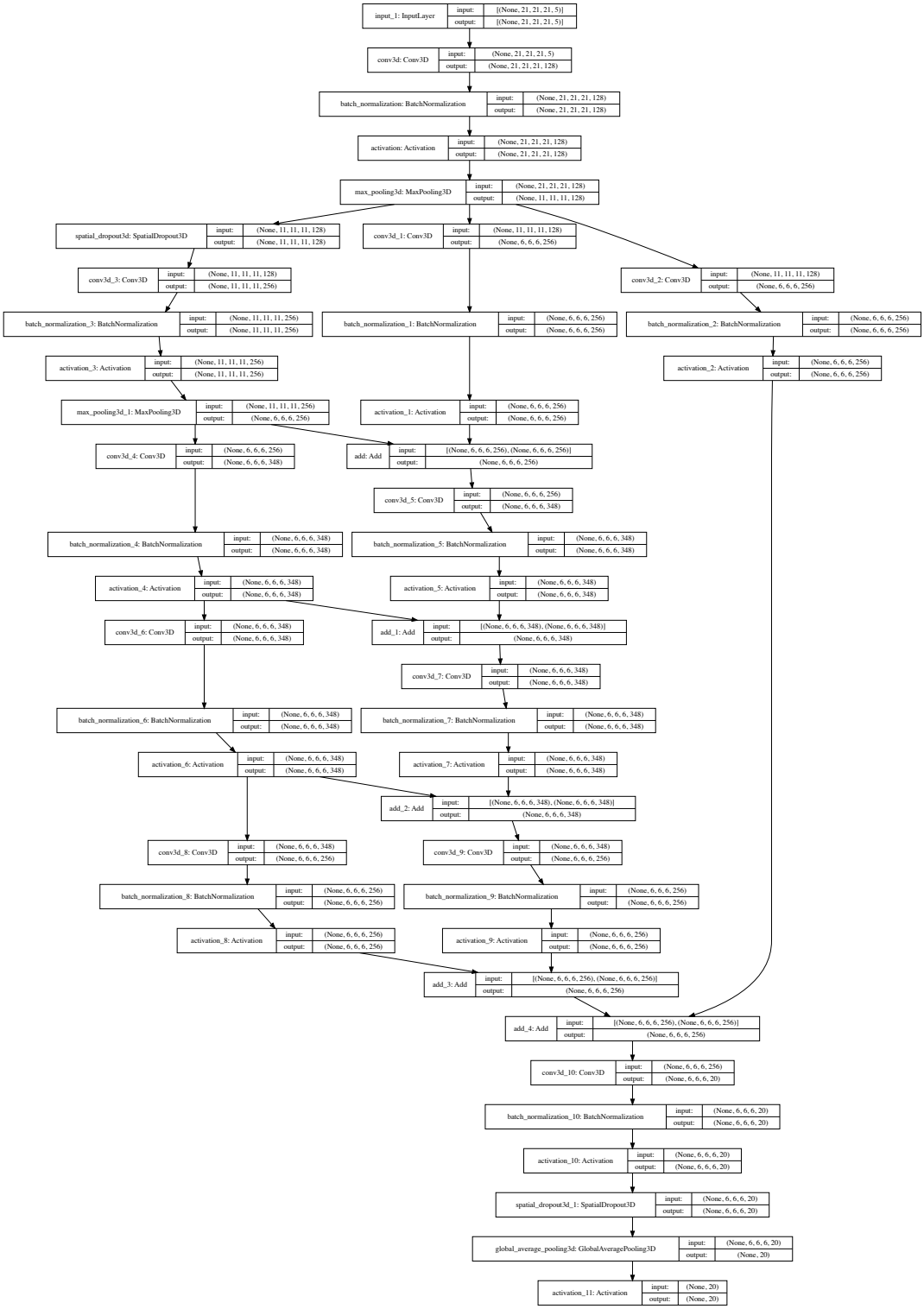



Figure 9: Architecture of the DenseCPD obtained from the model.json file from the authors.

# References

Alford, R. F.; Leaver-Fay, A.; Jeliazkov, J. R.; O'Meara, M. J.; DiMaio, F. P.; Park, H.; Shapovalov, M. V.; Renfrew, P. D.; Mulligan, V. K.; Kappel, K.; Labonte, J. W.; Pacella, M. S.; Bonneau, R.; Bradley, P.; Dunbrack, R. L.; Das, R.; Baker, D.; Kuhlman, B.; Kortemme, T.; and Gray, J. J. 2017. The Rosetta All-Atom Energy Function for Macromolecular Modeling and Design. *Journal of Chemical Theory and Computation*, 13(6): 3031–3048.

Huang, X.; Pearce, R.; and Zhang, Y. 2019. EvoEF2: accurate and fast energy function for computational protein design. *Bioinformatics*, 36(4): 1135–1142.

# A.2 TIMED

## A.2.1 TIMED Architecture

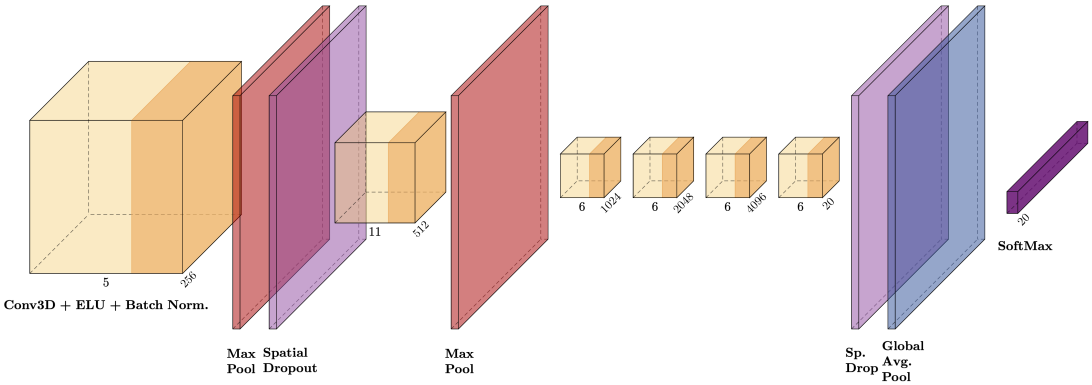
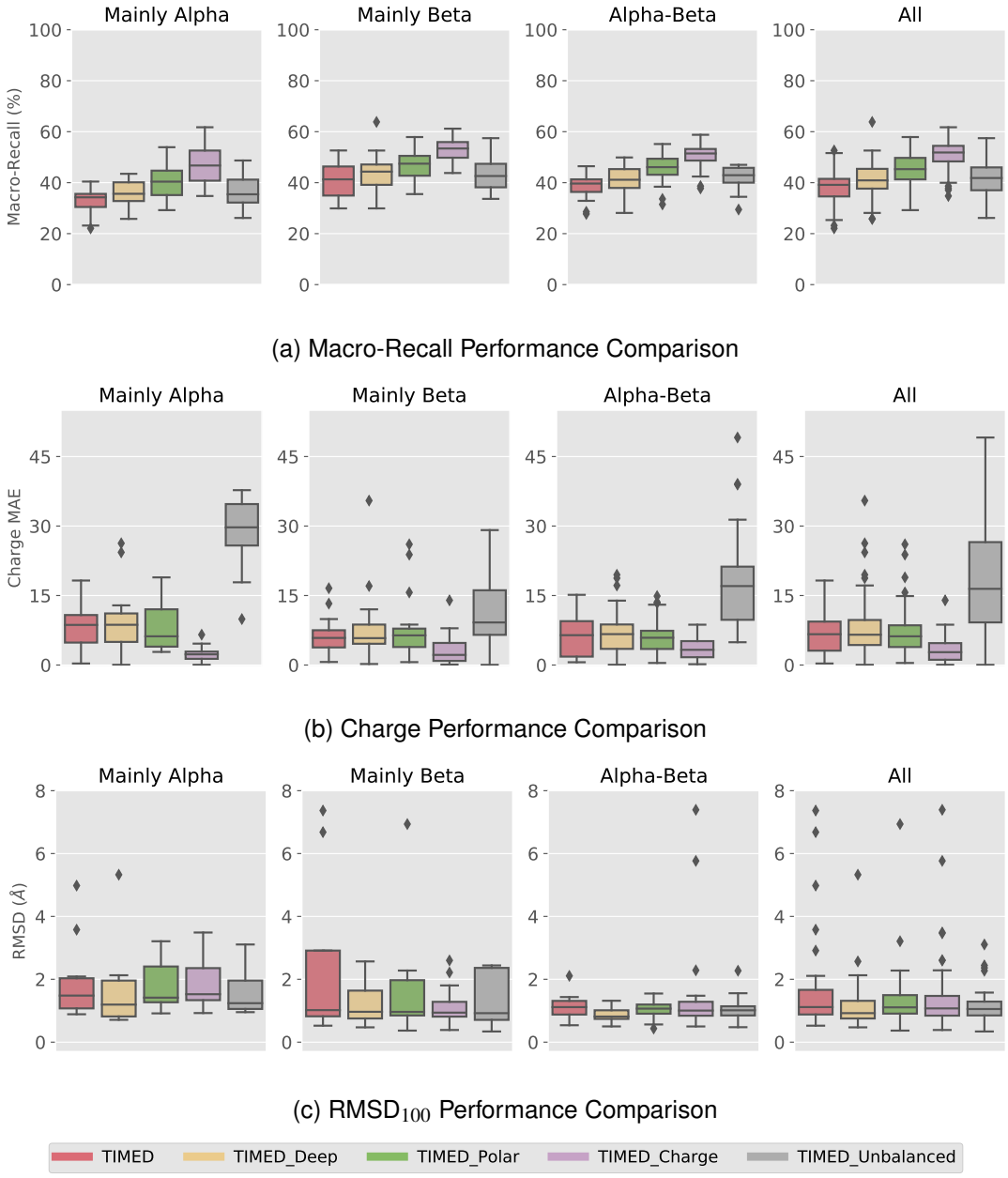



Figure A.1: Architecture of the TIMED family of neural networks.

### A.2.2 Unbalanced Performance Comparison




(a) Macro-Recall Performance Comparison

(b) Charge Performance Comparison

(c) $\text{RMSD}_{100}$ Performance Comparison

Figure A.2: Performance of TIMED networks with unbalanced classes in training data.

# TIMED-Design: Flexible and Accessible Protein Sequence Design with Convolutional Neural Networks

**Supplementary Materials**

## 1 TIMED Architecture

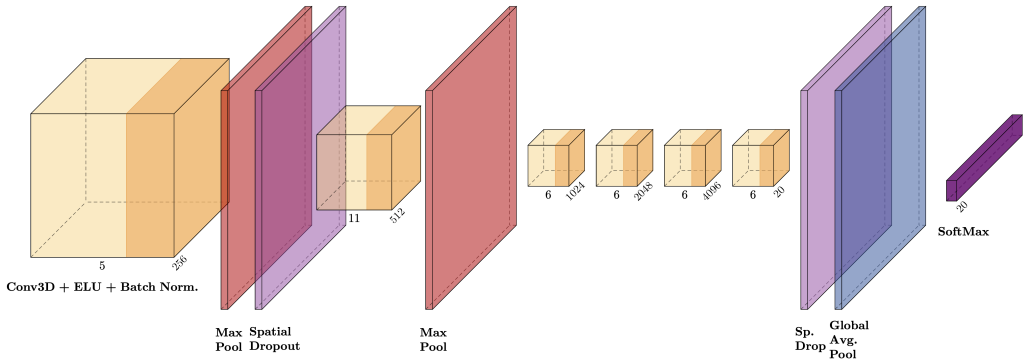



Figure S1: Architecture of the TIMED family of neural networks.

## 2 Charge and Polar Performance Comparison

### 2.1 All Models

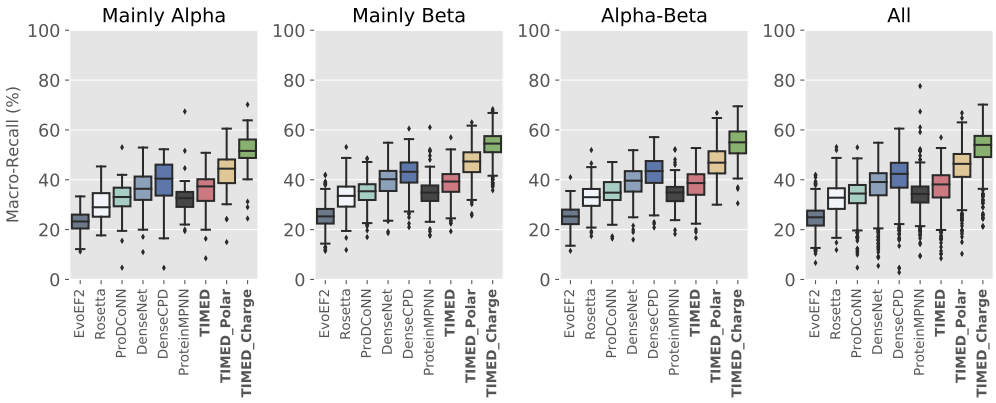



Figure S2: Macro-Recall Performance Comparison

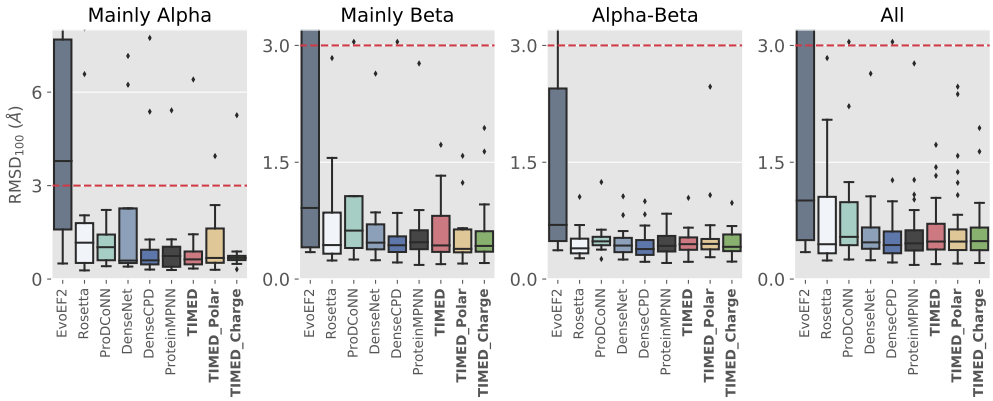



Figure S3: $RMSD_{100}$ Performance Comparison

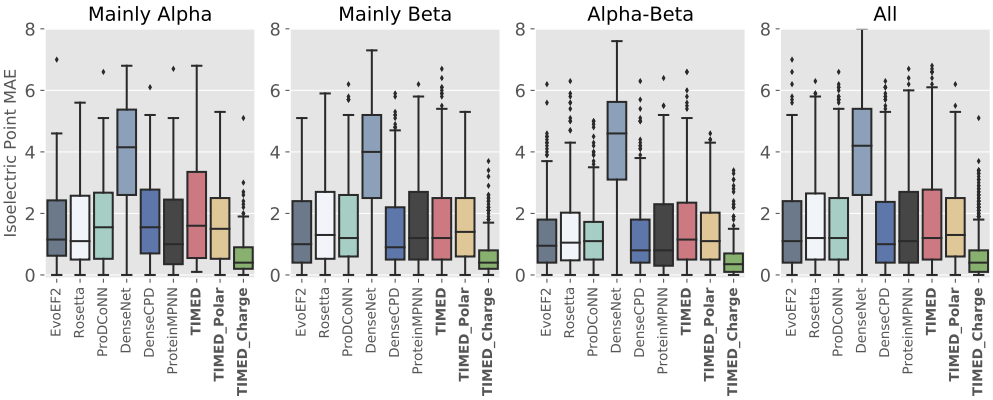



Figure S4: Isoelectric Point Performance Comparison

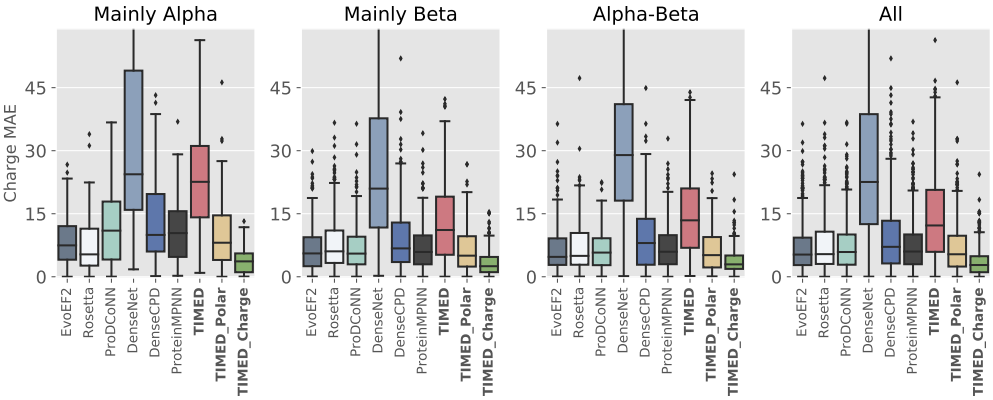



Figure S5: Charge Performance Comparison

## 2.2 TIMED Models Only

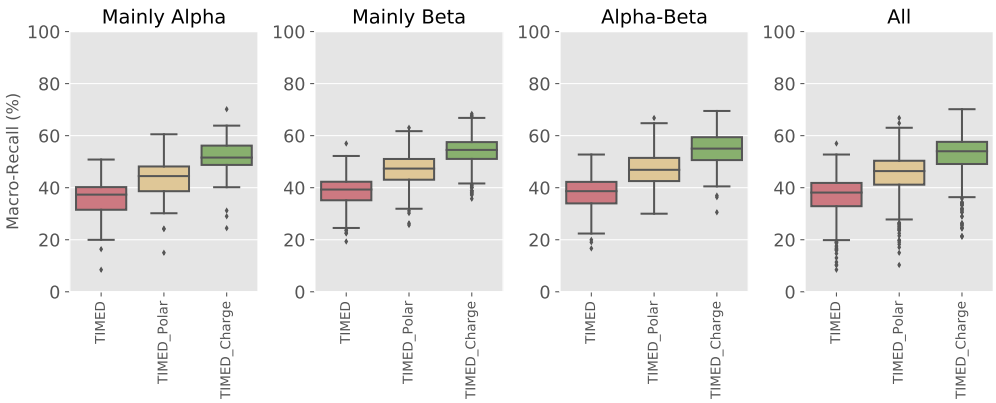



Figure S6: Macro-Recall Performance Comparison

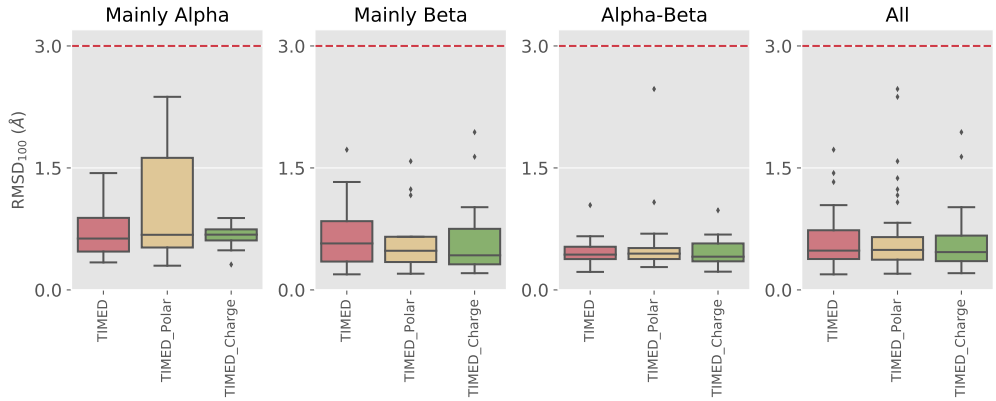



Figure S7: $RMSD_{100}$ Performance Comparison

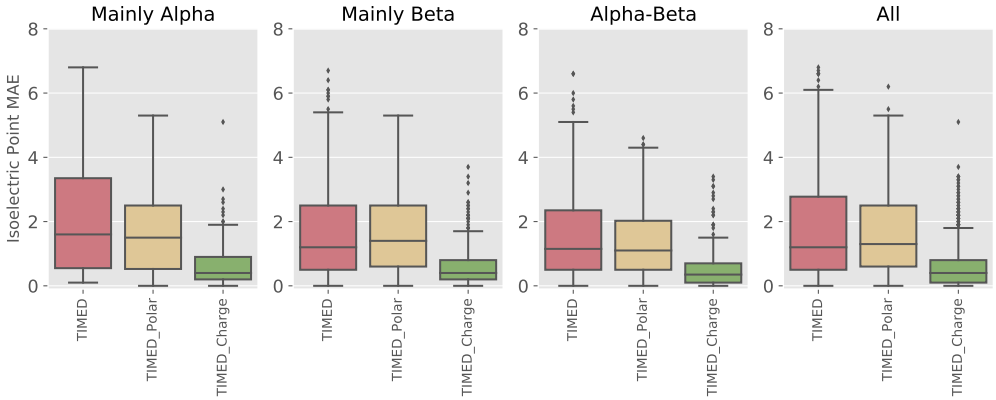



Figure S8: Isoelectric Point Performance Comparison

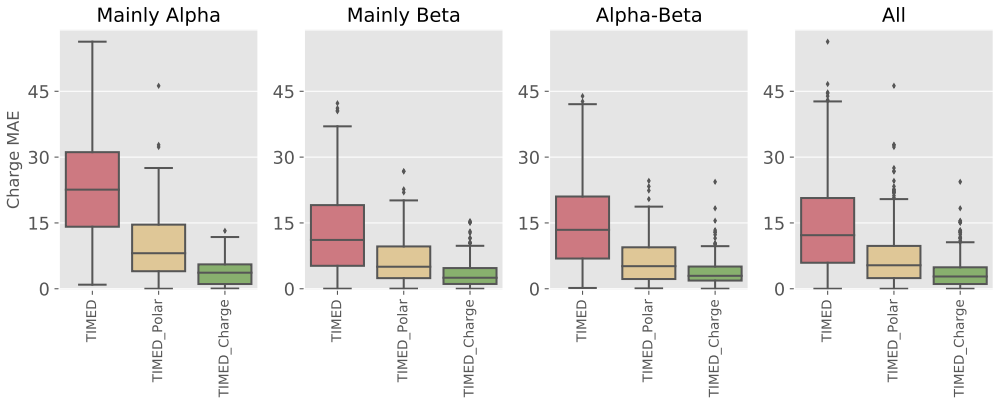



Figure S9: Charge Performance Comparison

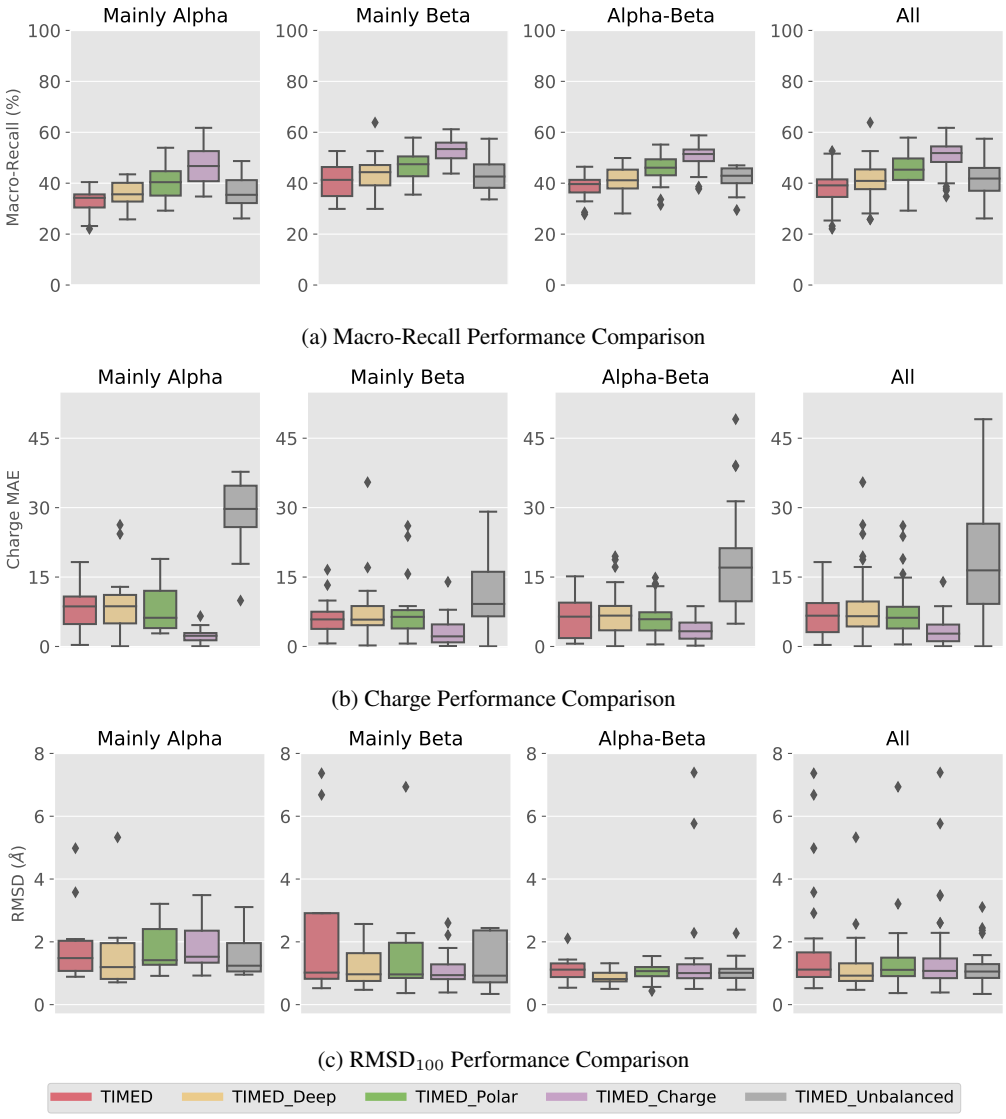
## 3 Retraining Models for Performance Comparison

Models in the literature use different training and validation data, making fair performance comparisons impossible. For this reason we retrained all of the state-of-the-art deep learning models with the same dataset. For benchmarking we used the PDBench structures Castorina et al. [2023] which were removed from the training set.

As several CNN models are not publicly available, we reached out to the authors of ProdCoNN Zhang et al. [2020] and DenseCPD Qi and Zhang [2020] to obtain their neural architecture. We then trained all of the CNN models with the same dataset.

For ProteinMPNN, we had to re-create tooling around custom dataset creation under instructions of the authors, as this code had not been recorded. We first recreated the training set to ensure that the performance was comparable to that reported in the paper. We then retrained ProteinMPNN with the same structures as the CNN models. Tooling, scripts, datasets, training curves, and trained models are available in this repository: `https://github.com/wells-wood-research/ProteinMPNN_custom_training` .

## 4 Unbalanced Performance Comparison




(a) Macro-Recall Performance Comparison

(b) Charge Performance Comparison

(c) $RMSD_{100}$ Performance Comparison

Figure S10: Performance of TIMED networks with unbalanced classes in training data.

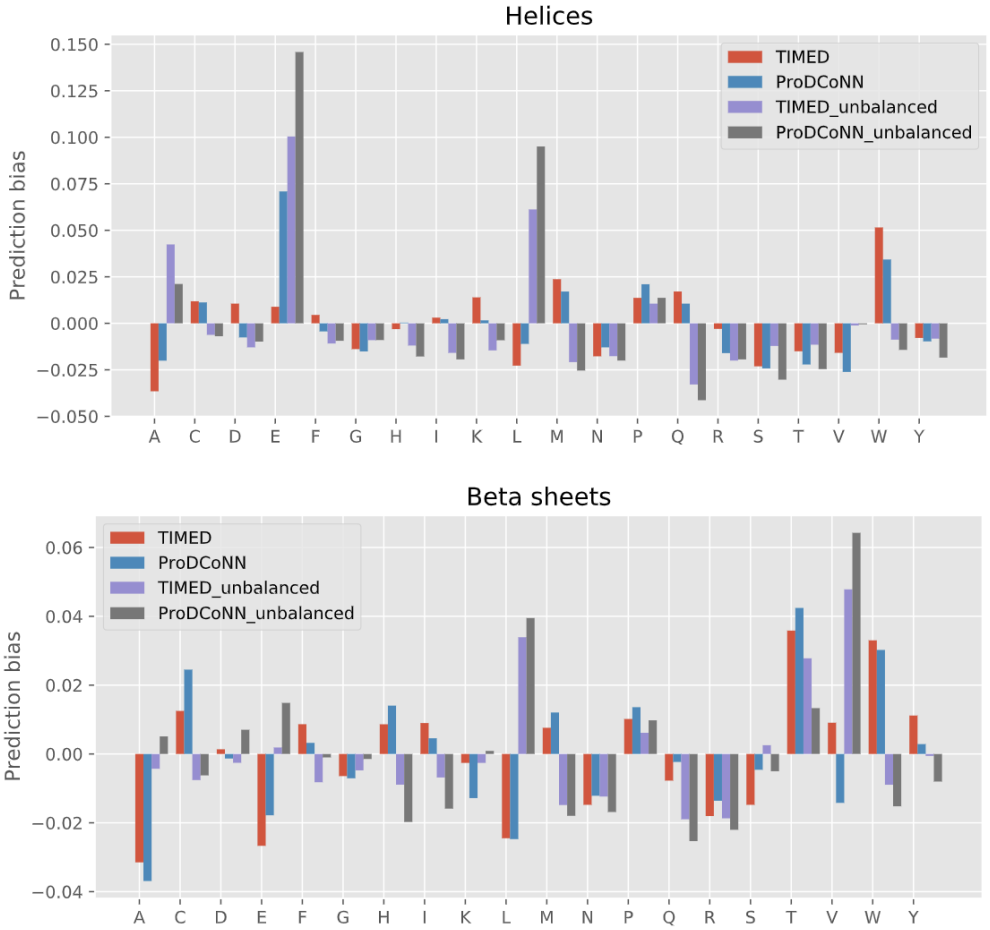



Figure S11: Prediction bias comparison for TIMED and ProDCoNN models both balanced and unbalanced versions.

## 5 Performance Correlation


Figure S12: Performance Correlation Matrix between all models tested

## 6 Monte Carlo Sampling Data

| Class | Temperature | n | Accuracy | Entropy | Packing Density | AlphaFold IDDT | RMSD |
|---|---|---|---|---|---|---|---|
| Mainly Alpha | 0.2 | 3800 | 33.15 | 3.75 | 66.18 | 87.99 | 1.91 |
| Mainly Alpha | 0.6 | 3800 | 25.67 | 3.75 | 66.10 | 70.54 | 3.80 |
| Mainly Alpha | 1.0 | 3800 | 18.46 | 3.75 | 66.10 | 43.30 | 6.44 |
| Mainly Beta | 0.2 | 25830 | 35.62 | 3.61 | 63.56 | 82.17 | 2.44 |
| Mainly Beta | 0.6 | 27550 | 28.33 | 3.61 | 63.60 | 69.06 | 3.54 |
| Mainly Beta | 1.0 | 27055 | 20.92 | 3.61 | 63.59 | 45.30 | 5.94 |
| Alpha-Beta | 0.2 | 19000 | 34.23 | 3.73 | 64.36 | 87.33 | 1.92 |
| Alpha-Beta | 0.6 | 19950 | 26.82 | 3.73 | 64.51 | 75.21 | 2.83 |
| Alpha-Beta | 1.0 | 19000 | 19.19 | 3.72 | 64.50 | 46.27 | 5.79 |

Table 1: Monte Carlo sampling information.

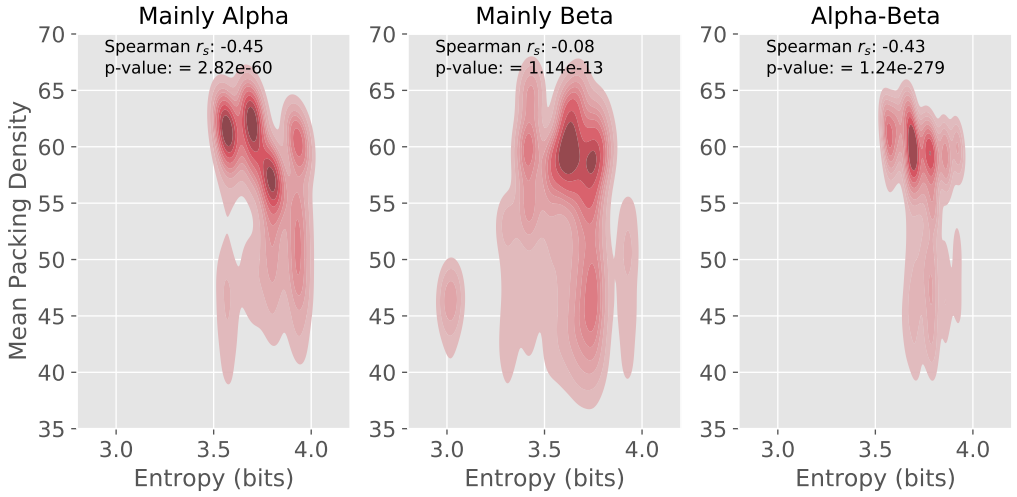



Figure S13: Mean Packing Density and Entropy over Different Folds

## References


Leonardo V Castorina, Rokas Petrenas, Kartic Subr, and Christopher W Wood. PDBench: evaluating computational methods for protein-sequence design. *Bioinformatics*, 39(1):btad027, 01 2023. ISSN 1367-4811. doi: 10.1093/bioinformatics/btad027. URL `https://doi.org/10.1093/bioinformatics/btad027`.

Yifei Qi and John Z. H. Zhang. DenseCPD: Improving the accuracy of neural-network-based computational protein sequence design with DenseNet. *Journal of Chemical Information and Modeling*, 60(3):1245–1252, mar 2020.

Yuan Zhang, Yang Chen, Chenran Wang, Chun-Chao Lo, Xiuwen Liu, Wei Wu, and Jinfeng Zhang. ProDCoNN: Protein design using a convolutional neural network. *Proteins: Structure, Function, and Bioinformatics*, 88(7):819–829, jan 2020.

| Fold | Temp. | Var 1 | Var 2 | Pearson Coeff | p value | Spearman Coeff | p value |
|---|---|---|---|---|---|---|---|
| All | All | Accuracy | RMSD | -0.56797 | 0 | -0.655642 | 0 |
| Mainly Alpha | 0.2 | Accuracy | RMSD | -0.595841 | 2.52517e-39 | -0.350521 | 7.32416e-13 |
| Mainly Beta | 0.2 | Accuracy | RMSD | -0.124207 | 6.84963e-11 | -0.233826 | 2.41201e-35 |
| Alpha-Beta | 0.2 | Accuracy | RMSD | -0.481453 | 2.79076e-116 | -0.603338 | 4.02228e-198 |
| Mainly Alpha | 1.0 | Accuracy | RMSD | -0.37723 | 5.65311e-15 | -0.446121 | 5.86543e-21 |
| Mainly Beta | 1.0 | Accuracy | RMSD | -0.407154 | 7.5916e-115 | -0.446668 | 1.49708e-140 |
| Alpha-Beta | 1.0 | Accuracy | RMSD | -0.479247 | 1.18128e-115 | -0.510681 | 1.28883e-133 |
| Mainly Alpha | 0.6 | Accuracy | RMSD | -0.672088 | 6.87584e-54 | -0.562392 | 9.7606e-35 |
| Mainly Beta | 0.6 | Accuracy | RMSD | -0.32244 | 3.78564e-71 | -0.333515 | 2.75857e-76 |
| Alpha-Beta | 0.6 | Accuracy | RMSD | -0.477962 | 2.55281e-120 | -0.518858 | 4.37809e-145 |
| All | All | Entropy | RMSD | 0.160827 | 9.66338e-94 | 0.217371 | 1.82749e-171 |
| Mainly Alpha | 0.2 | Entropy | RMSD | 0.695966 | 1.71973e-58 | 0.715236 | 3.97986e-63 |
| Mainly Beta | 0.2 | Entropy | RMSD | 0.0724957 | 0.000145837 | 0.0963986 | 4.29356e-07 |
| Alpha-Beta | 0.2 | Entropy | RMSD | 0.572235 | 6.59832e-174 | 0.608614 | 1.69009e-202 |
| Mainly Alpha | 1.0 | Entropy | RMSD | 0.32269 | 3.8102e-11 | 0.35924 | 1.25209e-13 |
| Mainly Beta | 1.0 | Entropy | RMSD | 0.295059 | 1.18885e-58 | 0.367026 | 4.53285e-92 |
| Alpha-Beta | 1.0 | Entropy | RMSD | 0.473319 | 1.79585e-112 | 0.499859 | 3.2139e-127 |
| Mainly Alpha | 0.6 | Entropy | RMSD | 0.728486 | 2.2837e-67 | 0.782364 | 7.31436e-84 |
| Mainly Beta | 0.6 | Entropy | RMSD | 0.257023 | 5.62798e-45 | 0.287796 | 2.00541e-56 |
| Alpha-Beta | 0.6 | Entropy | RMSD | 0.475365 | 7.39946e-119 | 0.532351 | 5.02197e-154 |
| All | All | Packing Density | Accuracy | 0.501944 | 0 | 0.53743 | 0 |
| Mainly Alpha | 0.2 | Packing Density | Accuracy | -0.0509211 | 0.312742 | 0.137834 | 0.00607275 |
| Mainly Beta | 0.2 | Packing Density | Accuracy | 0.0292804 | 0.125445 | 0.164092 | 5.42147e-18 |
| Alpha-Beta | 0.2 | Packing Density | Accuracy | 0.379528 | 2.29439e-69 | 0.372049 | 1.59287e-66 |
| Mainly Alpha | 1.0 | Packing Density | Accuracy | 0.395744 | 1.89666e-16 | 0.416918 | 2.98189e-18 |
| Mainly Beta | 1.0 | Packing Density | Accuracy | 0.258082 | 8.08204e-45 | 0.292361 | 1.43691e-57 |
| Alpha-Beta | 1.0 | Packing Density | Accuracy | 0.515746 | 1.08638e-136 | 0.516685 | 2.88705e-137 |
| Mainly Alpha | 0.6 | Packing Density | Accuracy | 0.647685 | 6.11114e-49 | 0.552489 | 2.40781e-33 |
| Mainly Beta | 0.6 | Packing Density | Accuracy | 0.098282 | 1.13785e-07 | 0.147498 | 1.425e-15 |
| Alpha-Beta | 0.6 | Packing Density | Accuracy | 0.402867 | 8.98781e-83 | 0.399854 | 1.86396e-81 |
| All | All | Packing Density | Entropy | -0.0196157 | 0.0128103 | -0.185332 | 2.24292e-124 |
| Mainly Alpha | 0.2 | Packing Density | Entropy | -0.477704 | 6.55064e-24 | -0.587934 | 4.34216e-38 |
| Mainly Beta | 0.2 | Packing Density | Entropy | 0.240969 | 1.70861e-37 | -0.008196 | 0.668046 |
| Alpha-Beta | 0.2 | Packing Density | Entropy | -0.45292 | 1.72486e-101 | -0.505157 | 1.10878e-129 |
| Mainly Alpha | 1.0 | Packing Density | Entropy | -0.261395 | 1.13269e-07 | -0.252345 | 3.15235e-07 |
| Mainly Beta | 1.0 | Packing Density | Entropy | -0.0666522 | 0.000356959 | -0.19001 | 1.07187e-24 |
| Alpha-Beta | 1.0 | Packing Density | Entropy | -0.496613 | 2.40773e-125 | -0.500975 | 7.21699e-128 |
| Mainly Alpha | 0.6 | Packing Density | Entropy | -0.80347 | 1.13261e-91 | -0.787361 | 1.24796e-85 |
| Mainly Beta | 0.6 | Packing Density | Entropy | 0.0943589 | 3.57023e-07 | -0.0658145 | 0.000390329 |
| Alpha-Beta | 0.6 | Packing Density | Entropy | -0.445769 | 4.63934e-103 | -0.546392 | 9.06595e-164 |
| All | All | Packing Density | RMSD | -0.733224 | 0 | -0.713059 | 0 |
| Mainly Alpha | 0.2 | Packing Density | RMSD | -0.138153 | 0.00595513 | -0.252268 | 3.76985e-07 |
| Mainly Beta | 0.2 | Packing Density | RMSD | -0.317524 | 3.12801e-65 | -0.519443 | 3.17717e-189 |
| Alpha-Beta | 0.2 | Packing Density | RMSD | -0.276956 | 1.85825e-36 | -0.371696 | 2.15964e-66 |
| Mainly Alpha | 1.0 | Packing Density | RMSD | -0.662131 | 8.17734e-52 | -0.682603 | 3.59499e-56 |
| Mainly Beta | 1.0 | Packing Density | RMSD | -0.649463 | 0 | -0.642015 | 0 |
| Alpha-Beta | 1.0 | Packing Density | RMSD | -0.7393 | 0 | -0.755926 | 0 |
| Mainly Alpha | 0.6 | Packing Density | RMSD | -0.745023 | 5.50654e-72 | -0.744231 | 9.33188e-72 |
| Mainly Beta | 0.6 | Packing Density | RMSD | -0.651817 | 0 | -0.65408 | 0 |
| Alpha-Beta | 0.6 | Packing Density | RMSD | -0.674763 | 4.32909e-279 | -0.464115 | 1.14316e-112 |
| Mainly Alpha | 0.2 | Entropy | AlphaFold lDDT | -0.688717 | 7.67484e-57 | -0.571755 | 1.15385e-35 |
| Mainly Beta | 0.2 | Entropy | AlphaFold lDDT | -0.119032 | 4.10154e-10 | -0.157457 | 1.13151e-16 |
| Alpha-Beta | 0.2 | Entropy | AlphaFold lDDT | -0.483338 | 2.62108e-117 | -0.530606 | 2.69442e-145 |
| Mainly Alpha | 1.0 | Entropy | AlphaFold lDDT | -0.350669 | 5.12163e-13 | -0.233948 | 2.24383e-06 |
| Mainly Beta | 1.0 | Entropy | AlphaFold lDDT | -0.22921 | 1.81663e-35 | -0.284929 | 1.19817e-54 |
| Alpha-Beta | 1.0 | Entropy | AlphaFold lDDT | -0.535688 | 2.59553e-149 | -0.484368 | 1.8733e-118 |
| Mainly Alpha | 0.6 | Entropy | AlphaFold lDDT | -0.772877 | 1.24697e-80 | -0.724328 | 2.92692e-66 |
| Mainly Beta | 0.6 | Entropy | AlphaFold lDDT | -0.221967 | 1.06427e-33 | -0.267678 | 9.00821e-49 |
| Alpha-Beta | 0.6 | Entropy | AlphaFold lDDT | -0.481614 | 2.13787e-122 | -0.500194 | 2.3041e-133 |

Table 2: Correlation Coefficients between sequence metrics and shape metrics of Monte-Carlo-sampled sequences.

## A.3 Fragments

### A.3.1 Sliding window algorithm

Fragment detection using a sliding window approach. The algorithm scans the target protein structure $T$ using a fragment $F_k$ of length $n_k$, computing a similarity score at each valid position using a specified distance metric (e.g., LogPr for torsion angles or sequence-based metrics). The step size $s$ controls the stride of the window. The output is a Fragment Distance Matrix $D$, which stores similarity values for each alignment position.

**Algorithm 1** Fragment detection via sliding window

**Require:**

Target structure $T$ of length $L_T$

Fragment $F_k$ of length $n_k$

Distance Metric function $\text{Metric}(\cdot,\cdot)$ (e.g., LogPr)

Step size $s$ (default: $s = 1$)

**Ensure:**

Fragment Distance Matrix $D$ containing similarity values for each valid alignment

Initialize $D$ as an empty list

**for** $i \leftarrow 1$ to $L_T - n_k + 1$ in steps of $s$ **do** ▷ Slide fragment $F_k$ across $T$

    Extract data from $T[i : i + n_k - 1]$ (angles or sequence)

    Compute similarity: $D[i] = \text{Metric}(T[i : i + n_k - 1], F_k)$

**end for**

**return** $D$

### A.3.2 Fragment Detection Metrics and Details

For computational efficiency and robustness, we avoided structural alignment methods based on CEAligner like BioPython [31]. These methods were computationally expensive and, in our tests, exhibited instability and silent failures when applied to large-scale fragment detection.

#### A.3.2.1 Sequence-based distances

**Sequence Identity**

Sequence identity quantifies the fraction of residues that match exactly between a reference sequence and a fragment sequence. The metric is computed as the proportion of identical residues across all positions, where a value of 1 indicates a perfect match, and 0 indicates no matches.

$$\text{Sequence Identity} = 1 - \frac{\sum_{k=1}^{n} \text{match}(s_k^{\text{ref}}, s_k^{\text{frag}})}{n},$$

where:

- $n$: Total number of residues being compared.
- $s_k^{\text{ref}}$: Residue at position $k$ in the reference sequence.
- $s_k^{\text{frag}}$: Residue at position $k$ in the fragment sequence.
- $\text{match}(x, y)$: A function that returns 1 if the residues $x$ and $y$ are identical and 0 otherwise.

**BLOcks SUbstitution Matrix (BLOSUM):**

The BLOSUM metric measures the similarity between two sequences using the BLOSUM62 substitution matrix. Each residue pair is scored based on its substitution score in the matrix. A positive score indicates a likely substitution, while a negative score indicates an unlikely substitution [59].

$$\text{BLOSUM Distance} = 1 - \frac{\sum_{k=1}^{n} \text{BLOSUM62}(s_k^{\text{ref}}, s_k^{\text{frag}} > 0)}{n},$$

where:

- $n$: Total number of residues being compared.
- $s_k^{\text{ref}}$: Residue at position $k$ in the reference sequence.
- $s_k^{\text{frag}}$: Residue at position $k$ in the fragment sequence.
- $\text{BLOSUM62}(x, y)$: Lookup function in the BLOSUM62 matrix, returning the substitution score for residues $x$ and $y$.

#### A.3.2.2 Angle-based distances

**Root Mean Square (RMS)**

The RMS metric measures the delta in root mean square difference between the fragment's angles and the corresponding angles in the target structure. Smaller values indicate a closer match.

$$\text{RMS} = \sqrt{\left(\Delta\phi_j^2 + \Delta\psi_j^2\right)}$$

Where:

- $\Delta\phi_j^2$: Squared difference of $\phi$ angles between the fragment and target at position $j$.
- $\Delta\psi_j^2$: Squared difference of $\psi$ angles between the fragment and target at position $j$.

**Ramachandran RMSD (RamRMSD):**

The RamRMSD is an aggregate metric that accounts for the RMS distances of the $\phi$ and $\psi$ angles across all $n$ residues of interest, derived from their positions on the Ramachandran space [68].

$$\text{RamRMSD} = \sqrt{\frac{\sum_{\text{k}=1}^{\text{n}} \text{RMS}_\text{k}^2}{\text{n}}}$$

Where:

- $\text{RMS}_\text{k}$: The RMS distance of the $k$-th residue as defined above.
- $n$: The total number of residues being compared.

**Log Probability (LogPr):**

The logPr metric quantifies the statistical likelihood of observing the angular alignment between two structures compared to a random environment. It uses the differences in backbone torsion angles ($\phi$ and $\psi$) between a reference and a fragment to compute a log-transformed probability [68]..

$$\text{logPr} = \frac{1}{n}\sum_{k=1}^{n} \log_{10}\left(\frac{1}{180^\circ}\cdot\Delta\phi_k \cdot \frac{1}{180^\circ}\cdot\Delta\psi_k\right),$$

where:

- $n$: Total number of residues being compared.

- $\Delta\phi_k$: Absolute difference between the $\phi$ angles of the reference and fragment at residue $k$, with a small constant ($1e{-}10$) added to avoid zero.
- $\Delta\psi_k$: Absolute difference between the $\psi$ angles of the reference and fragment at residue $k$, with a small constant ($1e{-}10$) added to avoid zero.

### A.3.3 Fragment Detection Accuracy

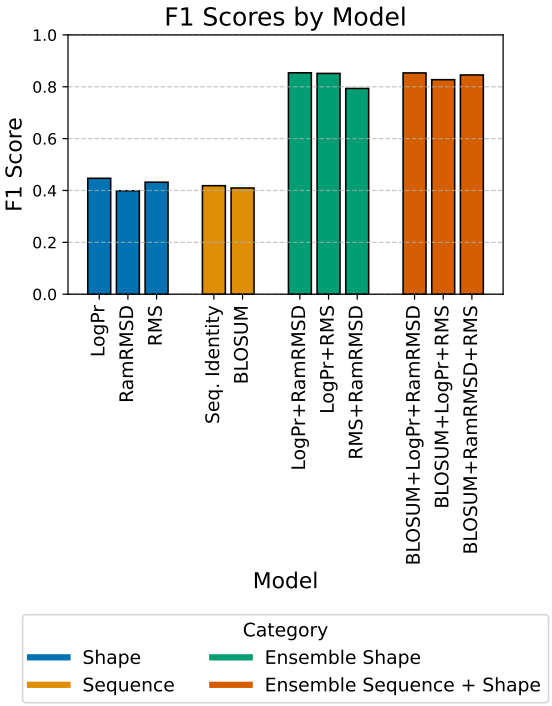



Figure A.3: Comparison of F1 scores for different fragment detection models using various distance metrics. The figure illustrates the performance of individual sequence-based and angle-based metrics, as well as combined models.

### A.3.4 Physico-chemical Properties in Fragments

Fragment coverage quantifies the proportion of a protein structure that is classified to a known fragment in our representation. This analysis evaluates fragment coverage across different protein folds and resolutions using the PDBench dataset. Additionally, we investigate how fragment coverage correlates with key structural and chemical properties, such as hydrogen bonding, solvent accessibility, charge distribution, polarity, and secondary structure content.

To assess these relationships, we compare coverage across different fold classes and resolution ranges, and we analyze whether fragment regions exhibit distinct physico-chemical characteristics compared to non-fragment regions.

### A.3.5 Fragment Coverage By Fold and Resolution

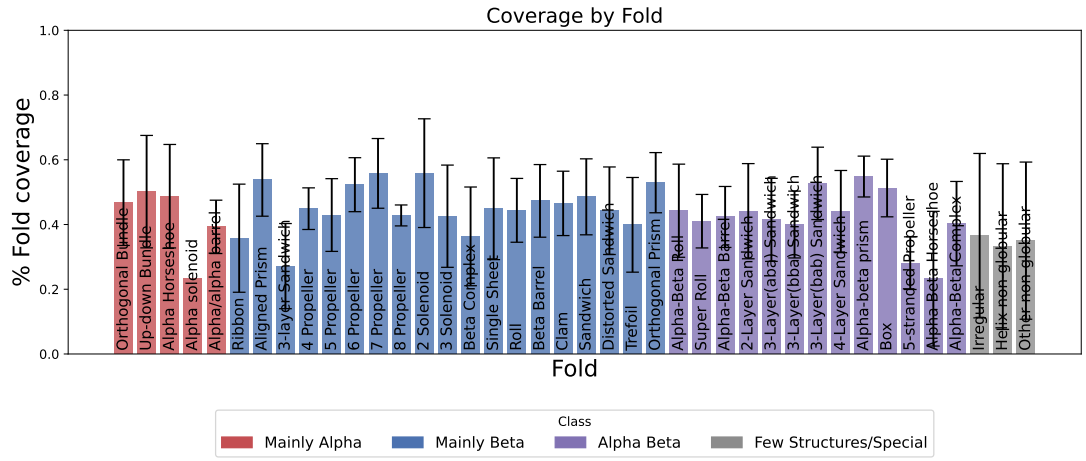



Figure A.4: Coverage of Fragments Across Folds

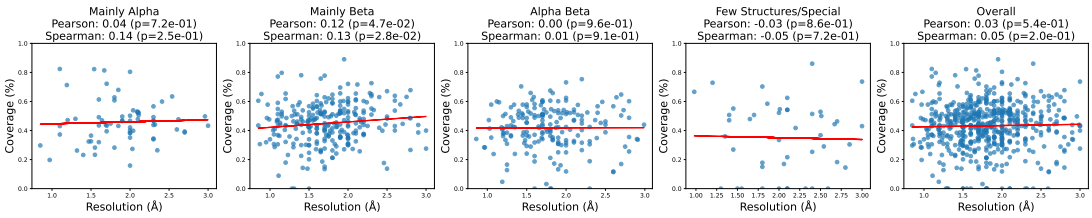



Figure A.5: Coverage of Fragments Across Resolutions

### A.3.6 H-bond Between Fragments Coverage

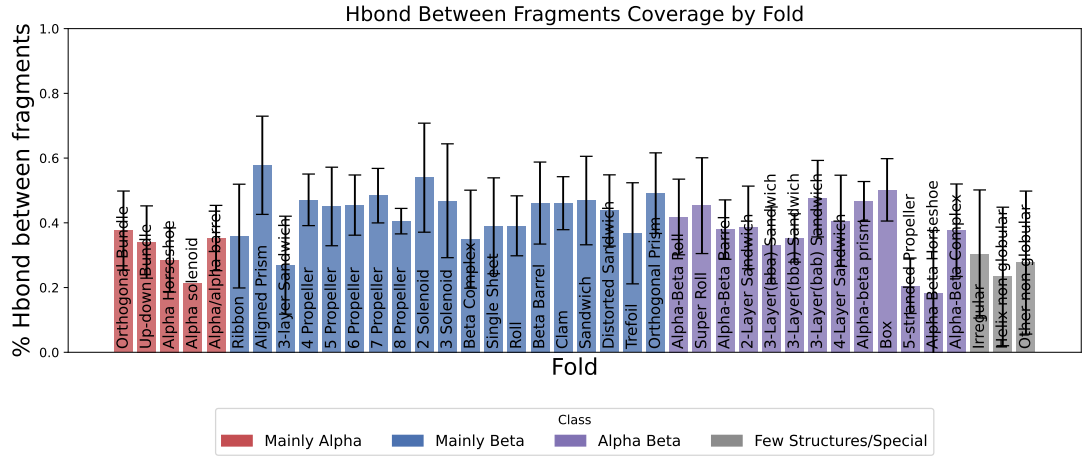



Figure A.6: H-bond Between Fragments Coverage by Fold.

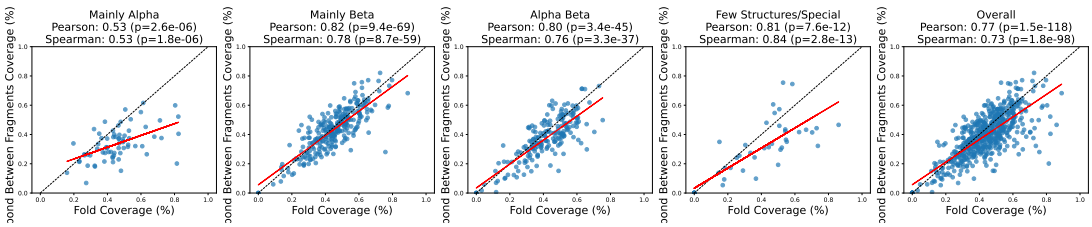



Figure A.7: H-bond Between Fragments vs. Fold Coverage.

### A.3.7 H-bond Within Fragments Coverage

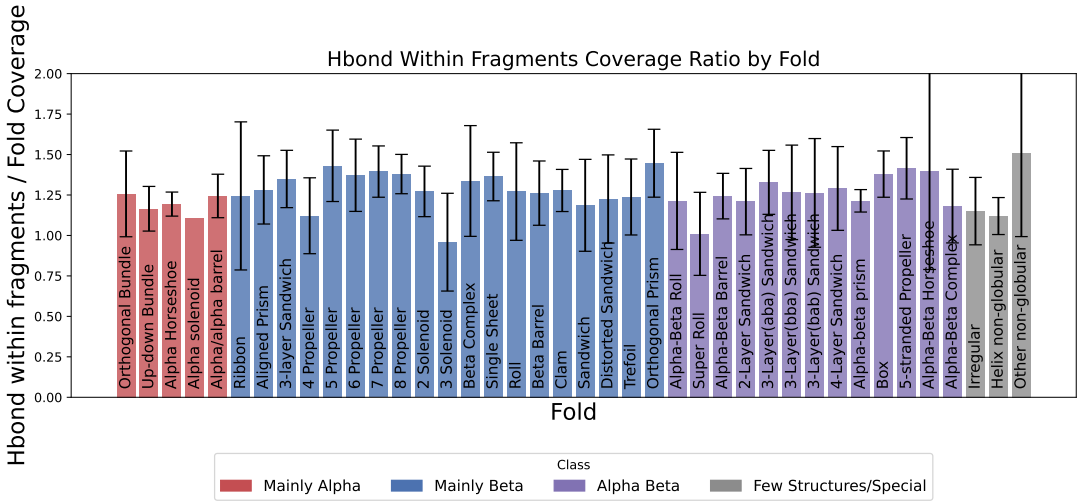



Figure A.8: H-bond Within Fragments Coverage Ratio by Fold.

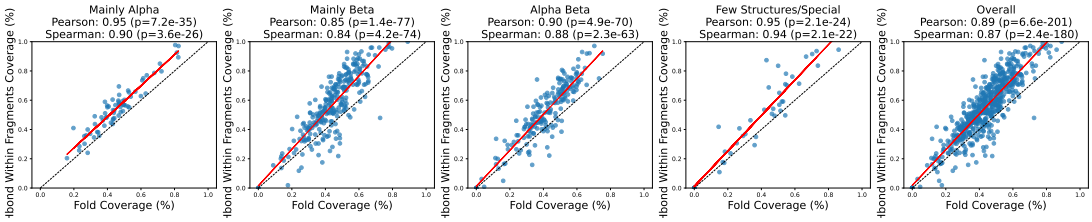



Figure A.9: H-bond Within Fragments vs. Fold Coverage.

### A.3.8 Accessibility Coverage

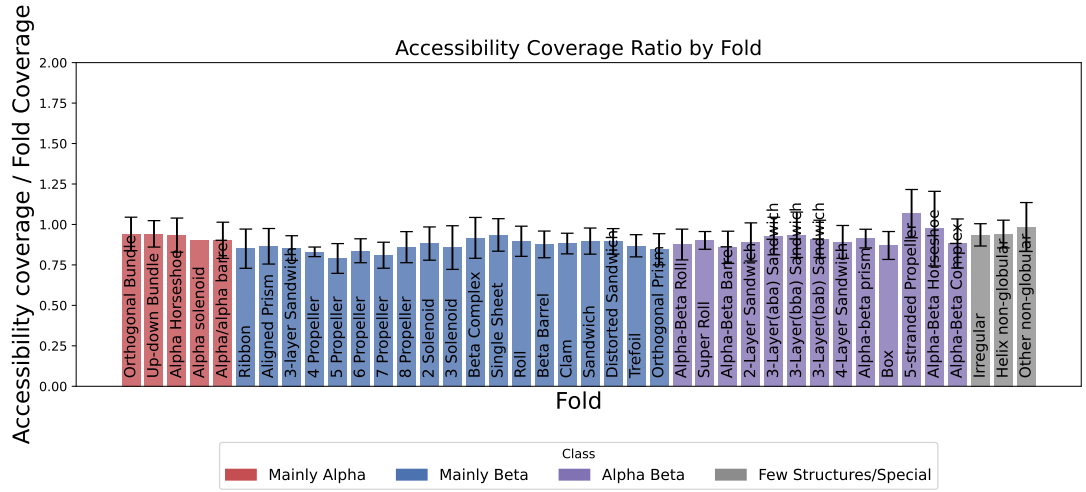



Figure A.10: Accessibility Coverage Ratio by Fold.

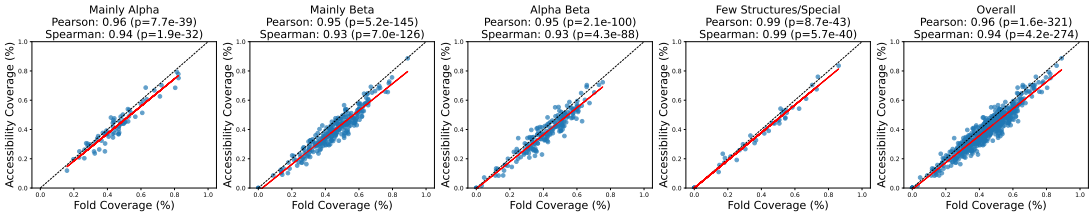



Figure A.11: Accessibility vs. Fold Coverage.

### A.3.9 Charge Coverage

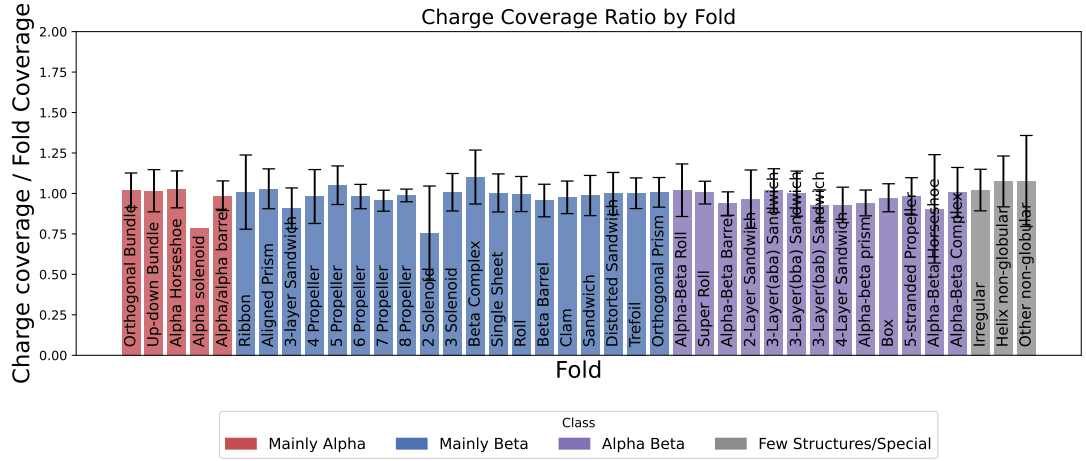



Figure A.12: Charge Coverage Ratio by Fold.

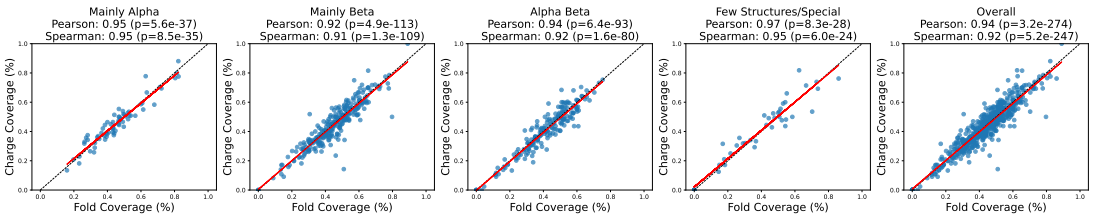



Figure A.13: Charge vs. Fold Coverage.

### A.3.10 Polarity Coverage

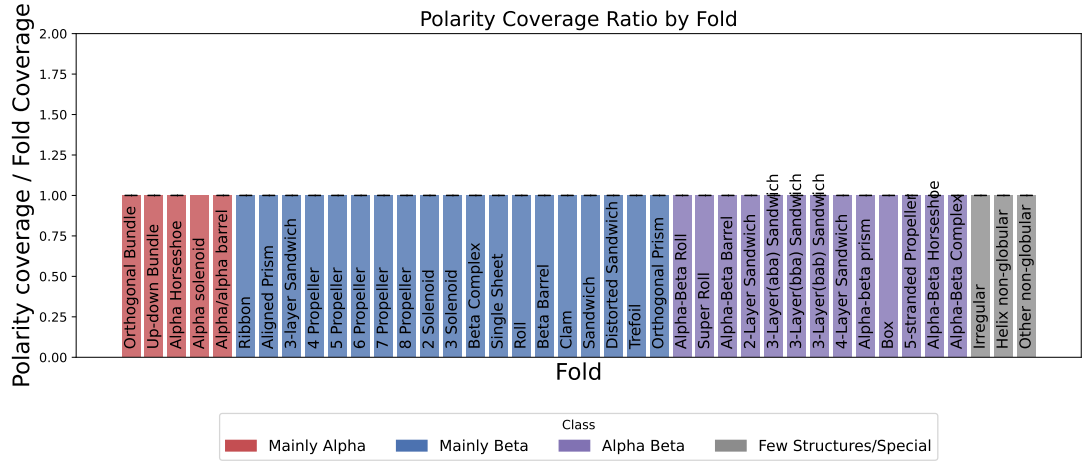



Figure A.14: Polarity Coverage Ratio by Fold.

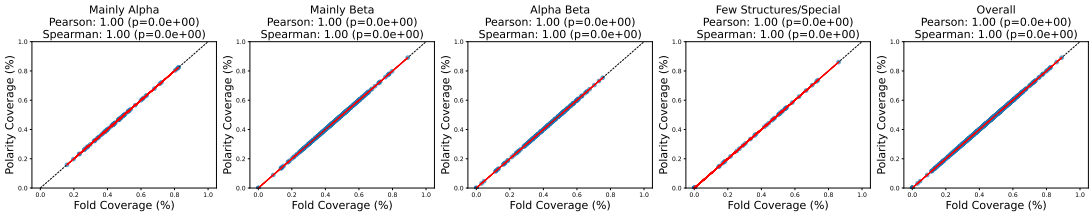



Figure A.15: Polarity vs. Fold Coverage.

### A.3.11 Secondary Structure Coverage

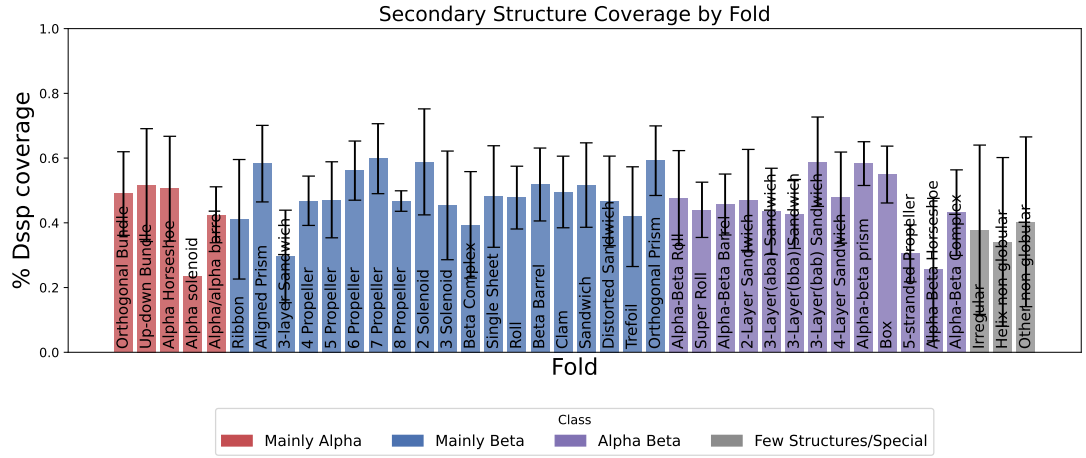



Figure A.16: Secondary Structure Ratio by Fold.

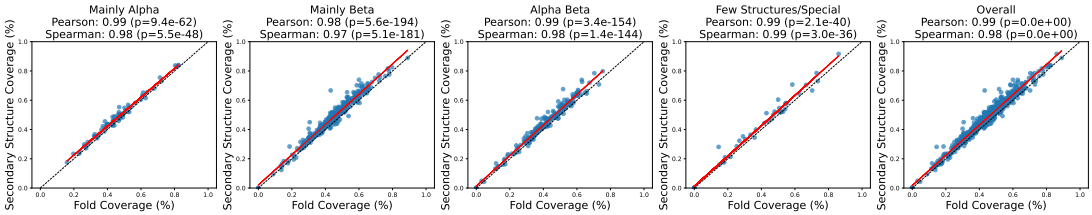



Figure A.17: Secondary Structure vs. Fold Coverage.

### A.3.12 Distance Spearman Correlations

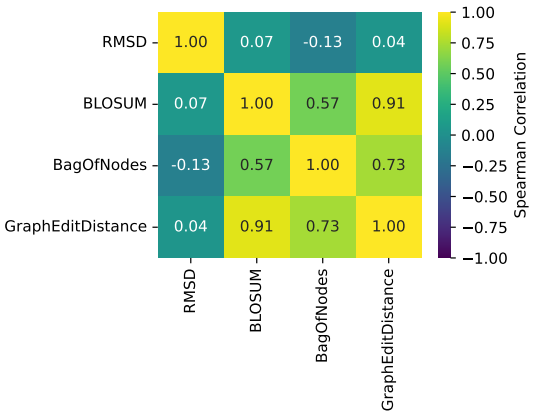



Figure A.18: Spearman correlation matrix between different protein similarity metrics. The figure shows pairwise correlations between BagOfNodes (fragment set representation), Graph Edit Distance (GED, fragment graph representation), BLOSUM (sequence similarity), and RMSD (structural similarity). Spearman correlation values indicate the rank-based relationships between these metrics, capturing how similarity scores compare across different representations.

# From Atoms to Fragments: A Coarse Representation for Efficient and Functional Protein Design

**Supplementary Materials**

## 1 Sliding window algorithm

Fragment detection using a sliding window approach. The algorithm scans the target protein structure $T$ using a fragment $F_k$ of length $n_k$, computing a similarity score at each valid position using a specified distance metric (e.g., LogPr for torsion angles or sequence-based metrics). The step size $s$ controls the stride of the window. The output is a Fragment Distance Matrix $D$, which stores similarity values for each alignment position.

**Algorithm 1** Fragment detection via sliding window

**Require:**
Target structure $T$ of length $L_T$
Fragment $F_k$ of length $n_k$
Distance Metric function Metric$(\cdot, \cdot)$ (e.g., LogPr)
Step size $s$ (default: $s = 1$)
**Ensure:**
Fragment Distance Matrix $D$ containing similarity values for each valid alignment
Initialize $D$ as an empty list
**for** $i \leftarrow 1$ to $L_T - n_k + 1$ in steps of $s$ **do** ▷ Slide fragment $F_k$ across $T$
    Extract data from $T[i : i + n_k - 1]$ (angles or sequence)
    Compute similarity: $D[i] = \text{Metric}(T[i : i + n_k - 1], F_k)$
**end for**
**return** $D$

## 2 Fragment Detection Metrics and Details

For computational efficiency and robustness, we avoided structural alignment methods based on CEAligner like BioPython [2]. These methods were computationally expensive and, in our tests, exhibited instability and silent failures when applied to large-scale fragment detection.

### 2.1 Sequence-based distances

#### Sequence Identity

Sequence identity quantifies the fraction of residues that match exactly between a reference sequence and a fragment sequence. The metric is computed as the proportion of identical residues across all positions, where a value of 1 indicates a perfect match, and 0 indicates no matches.

$$\text{Sequence Identity} = 1 - \frac{\sum_{k=1}^{n} \text{match}(s_k^{\text{ref}}, s_k^{\text{frag}})}{n},$$

where:

- $n$: Total number of residues being compared.
- $s_k^{\text{ref}}$: Residue at position $k$ in the reference sequence.
- $s_k^{\text{frag}}$: Residue at position $k$ in the fragment sequence.
- match$(x, y)$: A function that returns 1 if the residues $x$ and $y$ are identical and 0 otherwise.

**BLOcks SUbstitution Matrix (BLOSUM):**

The BLOSUM metric measures the similarity between two sequences using the BLOSUM62 substitution matrix. Each residue pair is scored based on its substitution score in the matrix. A positive score indicates a likely substitution, while a negative score indicates an unlikely substitution (TODO CITE).

$$\text{BLOSUM Distance} = 1 - \frac{\sum_{k=1}^{n} \text{BLOSUM62}(s_k^{\text{ref}}, s_k^{\text{frag}} > 0)}{n},$$

where:

- $n$: Total number of residues being compared.
- $s_k^{\text{ref}}$: Residue at position $k$ in the reference sequence.
- $s_k^{\text{frag}}$: Residue at position $k$ in the fragment sequence.
- BLOSUM62$(x, y)$: Lookup function in the BLOSUM62 matrix, returning the substitution score for residues $x$ and $y$.

### 2.2 Angle-based distances

**Root Mean Square (RMS)**

The RMS metric measures the delta in root mean square difference between the fragment's angles and the corresponding angles in the target structure. Smaller values indicate a closer match.

$$\text{RMS} = \sqrt{(\Delta\phi_j^2 + \Delta\psi_j^2)}$$

Where:

- $\Delta\phi_j^2$: Squared difference of $\phi$ angles between the fragment and target at position $j$.
- $\Delta\psi_j^2$: Squared difference of $\psi$ angles between the fragment and target at position $j$.

**Ramachandran RMSD (RamRMSD):**

The RamRMSD is an aggregate metric that accounts for the RMS distances of the $\phi$ and $\psi$ angles across all $n$ residues of interest, derived from their positions on the Ramachandran space (TODO Cite).

$$\text{RamRMSD} = \sqrt{\frac{\sum_{\text{k}=1}^{\text{n}} \text{RMS}_{\text{k}}^2}{\text{n}}}$$

Where:

- $\text{RMS}_{\text{k}}$: The RMS distance of the $k$-th residue as defined above.
- $n$: The total number of residues being compared.

**Log Probability (LogPr):**

The logPr metric quantifies the statistical likelihood of observing the angular alignment between two structures compared to a random environment. It uses the differences in backbone torsion angles ($\phi$ and $\psi$) between a reference and a fragment to compute a log-transformed probability (TODO CITE).

$$\text{logPr} = \frac{1}{n} \sum_{k=1}^{n} \log_{10} \left( \frac{1}{180^\circ} \cdot \Delta\phi_k \cdot \frac{1}{180^\circ} \cdot \Delta\psi_k \right),$$

where:

- $n$: Total number of residues being compared.
- $\Delta\phi_k$: Absolute difference between the $\phi$ angles of the reference and fragment at residue $k$, with a small constant ($1e{-}10$) added to avoid zero.
- $\Delta\psi_k$: Absolute difference between the $\psi$ angles of the reference and fragment at residue $k$, with a small constant ($1e{-}10$) added to avoid zero.

## 3 Number of Instance for Each Fragment

| Fragment Number | Count |
|---|---|
| 1 | 18 |
| 2 | 14 |
| 3 | 9 |
| 4 | 2 |
| 5 | 6 |
| 6 | 7 |
| 7 | 8 |
| 8 | 10 |
| 9 | 4 |
| 10 | 7 |
| 11 | 1 |
| 12 | 10 |
| 13 | 1 |
| 14 | 4 |
| 15 | 8 |
| 16 | 6 |
| 17 | 8 |
| 18 | 6 |
| 19 | 4 |
| 20 | 3 |
| 21 | 3 |
| 22 | 2 |
| 23 | 7 |
| 24 | 5 |
| 25 | 4 |
| 26 | 4 |
| 27 | 1 |
| 28 | 12 |
| 29 | 1 |
| 30 | 2 |
| 31 | 2 |
| 32 | 7 |
| 33 | 5 |
| 34 | 5 |
| 35 | 3 |
| 36 | 6 |
| 37 | 6 |
| 38 | 3 |
| 39 | 2 |
| 40 | 3 |

Table S1: Number of instances for each fragment in our fibrary.

## 4 Number of Proteins for each Function

| Function | Count |
|---|---|
| DNA+RNA | 10 |
| RNA | 10 |
| RNA+ATP+METAL | 10 |
| RNA+GTP | 10 |
| RNA+ATP | 10 |
| DNA+RNA+ATP | 9 |
| ATP+GTP+METAL | 10 |
| RNA+GTP+METAL | 10 |
| RNA+ATP+GTP | 10 |
| GTP | 11 |
| ATP+METAL | 10 |
| RNA+METAL | 9 |
| METAL | 10 |
| ATP | 9 |
| DNA+ATP+METAL | 10 |
| DNA+GTP | 6 |
| DNA | 8 |
| DNA+ATP+GTP | 5 |
| GTP+METAL | 10 |
| ATP+GTP | 9 |
| DNA+METAL | 10 |
| DNA+RNA+METAL | 9 |
| DNA+ATP | 10 |

Table S2: Number of Structures for each Function in the Protein Function Dataset (PFD).

## 5 Fragment Detection Accuracy

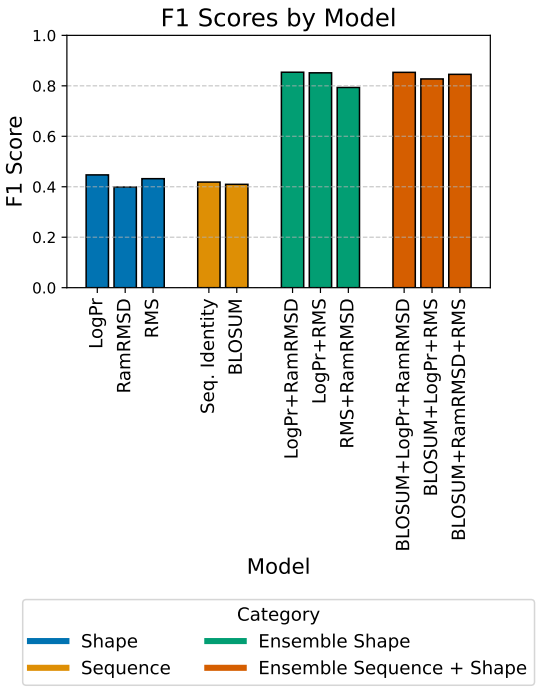



Figure S1: Comparison of F1 scores for different fragment detection models using various distance metrics. The figure illustrates the performance of individual sequence-based and angle-based metrics, as well as combined models.

## 6 Distance Spearman Correlations

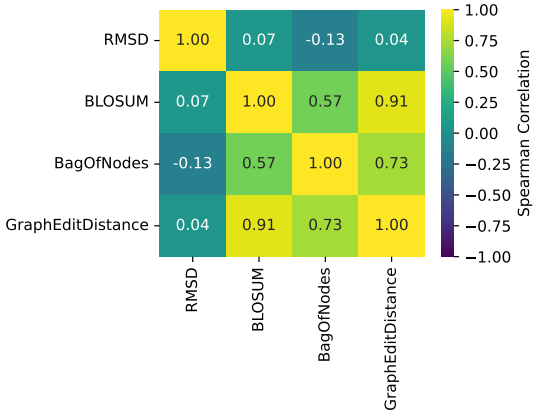



Figure S2: Spearman correlation matrix between different protein similarity metrics. The figure shows pairwise correlations between BagOfNodes (fragment set representation), Graph Edit Distance (GED, fragment graph representation), BLOSUM (sequence similarity), and RMSD (structural similarity). Spearman correlation values indicate the rank-based relationships between these metrics, capturing how similarity scores compare across different representations.

# 7 Functional Clustering

| Clustering Method | Dimensionality Reduction | Distance | ARI | V-Measure | Silhouette | Trustworthiness | Distance Spearman | F1 |
|---|---|---|---|---|---|---|---|---|
| GMM | PCoA | BagOfNodes | 0.0050 | 0.3455 | 0.8227 | 0.9991 | 0.9998 | 0.1660 |
| GMM | PCoA | BLOSUM | 0.0027 | 0.2933 | 0.0248 | 0.9923 | 0.9966 | 0.1640 |
| GMM | PCoA | GraphEditDistance | 0.0458 | 0.3832 | 0.0766 | 0.9915 | 0.9829 | 0.1985 |
| GMM | PCoA | RMSD | 0.0357 | 0.3863 | -0.0334 | 0.9598 | 0.8647 | 0.1957 |
| GMM | t-SNE | BagOfNodes | 0.0004 | 0.3431 | 0.8163 | 0.9945 | 0.9371 | 0.1485 |
| GMM | t-SNE | BLOSUM | 0.0043 | 0.3484 | -0.0004 | 0.8724 | 0.6607 | 0.1468 |
| GMM | t-SNE | GraphEditDistance | 0.0078 | 0.3499 | 0.1595 | 0.9787 | 0.8032 | 0.1632 |
| GMM | t-SNE | RMSD | 0.0370 | 0.3827 | 0.0006 | 0.6558 | 0.1752 | 0.1891 |
| GMM | UMAP | BagOfNodes | -0.0081 | 0.3283 | 0.6886 | 0.9466 | 0.5405 | 0.1478 |
| GMM | UMAP | BLOSUM | 0.0030 | 0.3452 | -0.0248 | 0.8717 | 0.6229 | 0.1392 |
| GMM | UMAP | GraphEditDistance | 0.0085 | 0.3545 | 0.1522 | 0.9707 | 0.5987 | 0.1656 |
| GMM | UMAP | RMSD | 0.0237 | 0.3741 | -0.0087 | 0.6681 | 0.1804 | 0.2038 |
| K-Means | PCoA | BagOfNodes | 0.0071 | 0.3486 | 0.8261 | 0.9991 | 0.9998 | 0.1562 |
| K-Means | PCoA | BLOSUM | 0.0020 | 0.2788 | 0.0238 | 0.9923 | 0.9966 | 0.1540 |
| K-Means | PCoA | GraphEditDistance | 0.0363 | 0.3621 | 0.1009 | 0.9915 | 0.9829 | 0.1868 |
| K-Means | PCoA | RMSD | 0.0486 | 0.3990 | -0.0249 | 0.9598 | 0.8647 | 0.2470 |
| K-Means | t-SNE | BagOfNodes | 0.0021 | 0.3459 | 0.8327 | 0.9945 | 0.9371 | 0.1635 |
| K-Means | t-SNE | BLOSUM | -0.0032 | 0.3375 | 0.0006 | 0.8724 | 0.6607 | 0.1670 |
| K-Means | t-SNE | GraphEditDistance | 0.0079 | 0.3536 | 0.1558 | 0.9787 | 0.8032 | 0.1782 |
| K-Means | t-SNE | RMSD | 0.0356 | 0.3859 | 0.0111 | 0.6558 | 0.1752 | 0.2248 |
| K-Means | UMAP | BagOfNodes | -0.0061 | 0.3278 | 0.6735 | 0.9466 | 0.5405 | 0.1406 |
| K-Means | UMAP | BLOSUM | 0.0023 | 0.3481 | -0.0126 | 0.8717 | 0.6229 | 0.1516 |
| K-Means | UMAP | GraphEditDistance | 0.0024 | 0.3462 | 0.1520 | 0.9707 | 0.5987 | 0.1490 |
| K-Means | UMAP | RMSD | 0.0240 | 0.3745 | 0.0032 | 0.6681 | 0.1804 | 0.1782 |

Table S3: Clustering performance across different methods, distance metrics, and dimensionality reduction techniques. The table reports Adjusted Rand Index (ARI), V-Measure, Silhouette score, Trustworthiness, Spearman correlation with original distances, and F1 score for Gaussian Mixture Models (GMM) and K-Means clustering. Each method is evaluated using four different distance metrics—BagOfNodes, BLOSUM, Graph Edit Distance (GED), and RMSD—applied to embeddings obtained via Principal Coordinate Analysis (PCoA), t-SNE, and UMAP.

# 8 Search Performance

We use the following metrics:

- **Normalized Discounted Cumulative Gain (NDCG)**: evaluates the quality of ranking by comparing the actual ranking with an ideal one, emphasizing higher relevance at the top of the list. It calculates a weighted relevance score for ranked proteins, where the highest possible score (ideal) is compared to the actual retrieval. NDCG accounts for multi-functional proteins by assigning partial relevance to those containing the query function.

- **Area Under the Receiver Operating Characteristic (AUROC)**: measures how effectively the method ranks proteins sharing the same function higher than those with different functions. For each protein in the dataset, we identify relevant proteins (i.e., those matching the query's function) and compute the AUROC by comparing the distances between proteins, with closer distances indicating higher relevance. Distances are converted into scores, and the AUROC quantifies the likelihood that a relevant protein is ranked above a non-relevant one. This metric is aggregated across all proteins and functional categories in the dataset to assess overall retrieval success.

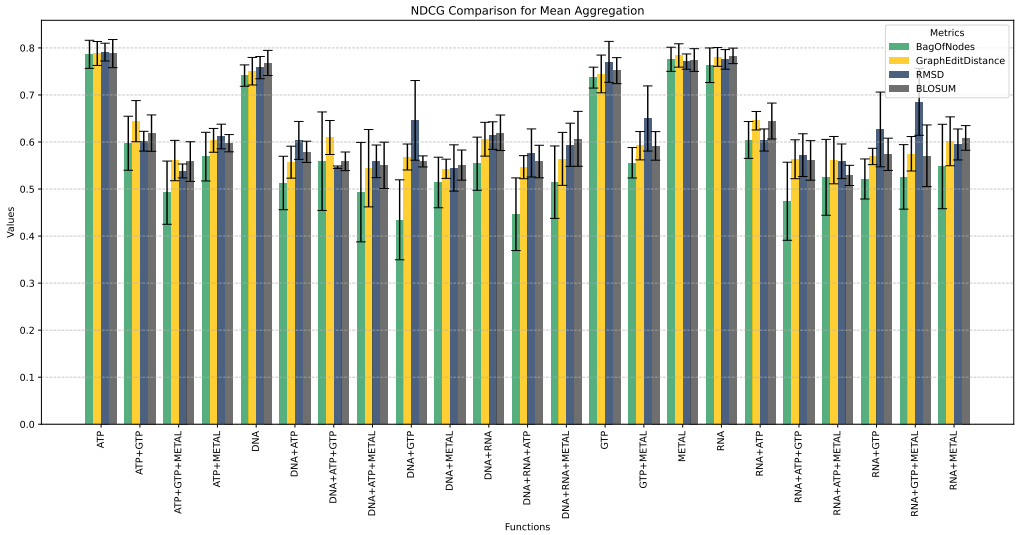



Figure S3: Normalized Discounted Cumulative Gain (NDCG) scores for protein similarity search across different distance metrics. Higher values indicate that functionally relevant proteins are ranked higher in the search results.

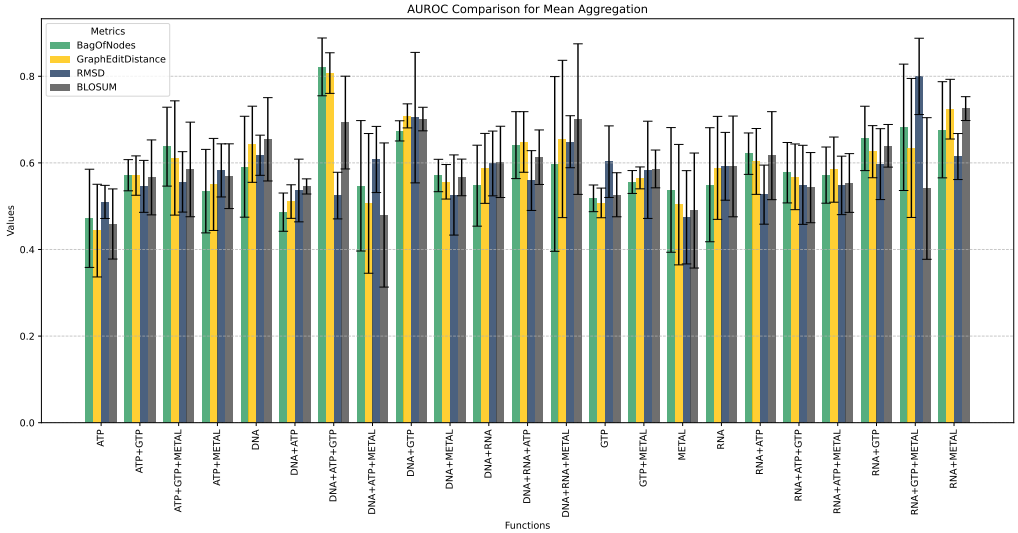



Figure S4: Area Under the Receiver Operating Characteristic Curve (AUROC) scores for protein similarity search across different distance metrics. Higher values indicate better separation between relevant and non-relevant search results.

## 9 ROC Threshold

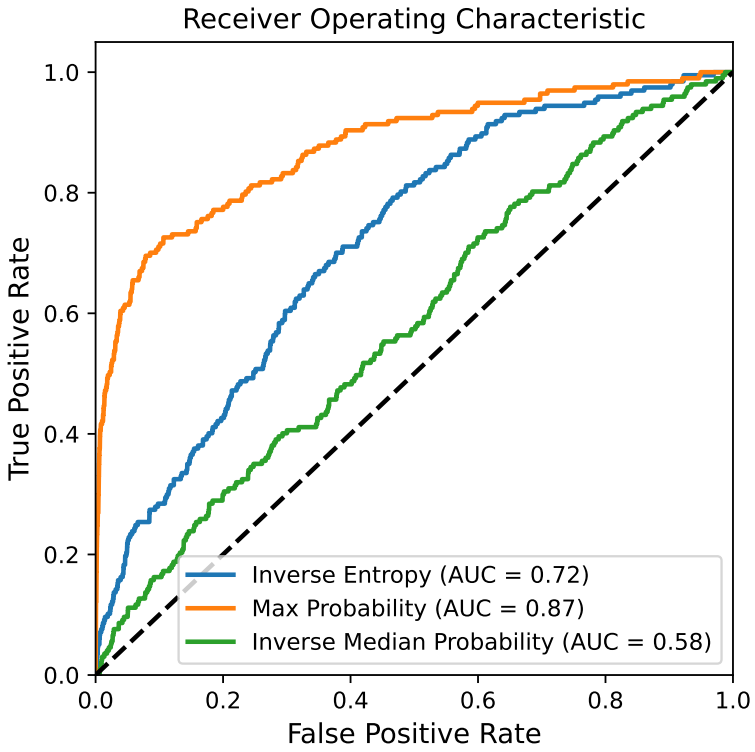



Figure S5: Receiver Operating Characteristic (ROC) curve for fragment detection. The ROC curve illustrates the true positive rate (sensitivity) versus the false positive rate (1 - specificity) for different classification thresholds. The area under the ROC curve (AUROC) quantifies the overall performance of the fragment detection algorithm, with higher values indicating better discrimination between true and false fragment classifications.

## 10 Physico-chemical Properties in Fragments

Fragment coverage quantifies the proportion of a protein structure that is classified to a known fragment in our representation. This analysis evaluates fragment coverage across different protein folds and resolutions using the PDBench dataset[1]. Additionally, we investigate how fragment coverage correlates with key structural and chemical properties, such as hydrogen bonding, solvent accessibility, charge distribution, polarity, and secondary structure content.

To assess these relationships, we compare coverage across different fold classes and resolution ranges, and we analyze whether fragment regions exhibit distinct physicochemical characteristics compared to non-fragment regions.

## 10.1 Fragment Coverage By Fold and Resolution

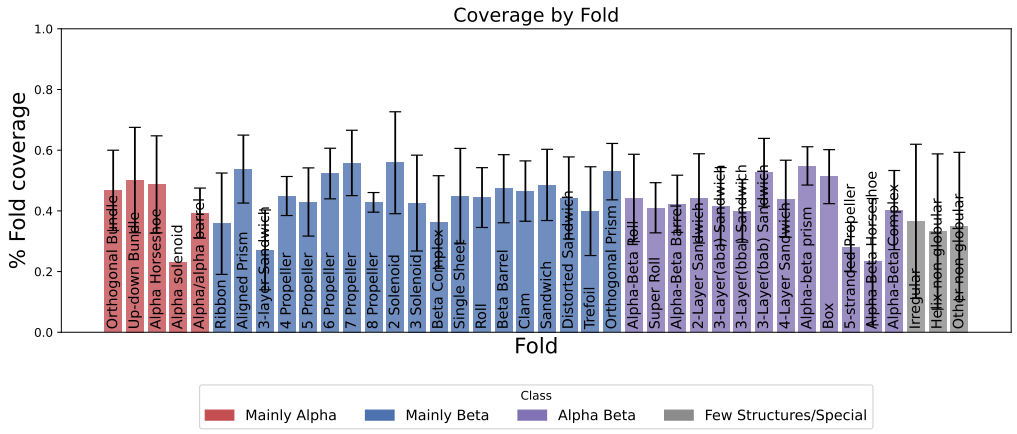



Figure S6: Coverage of Fragments Across Folds

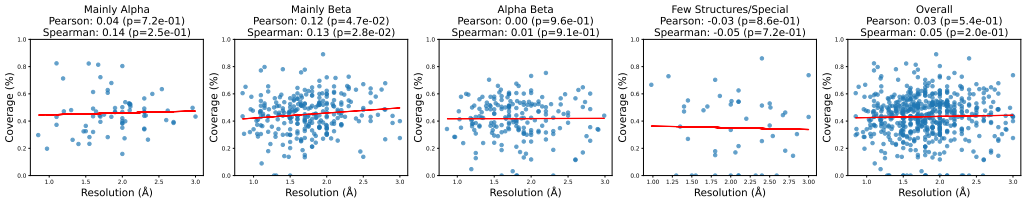



Figure S7: Coverage of Fragments Across Resolutions

## 10.2 H-bond Between Fragments Coverage

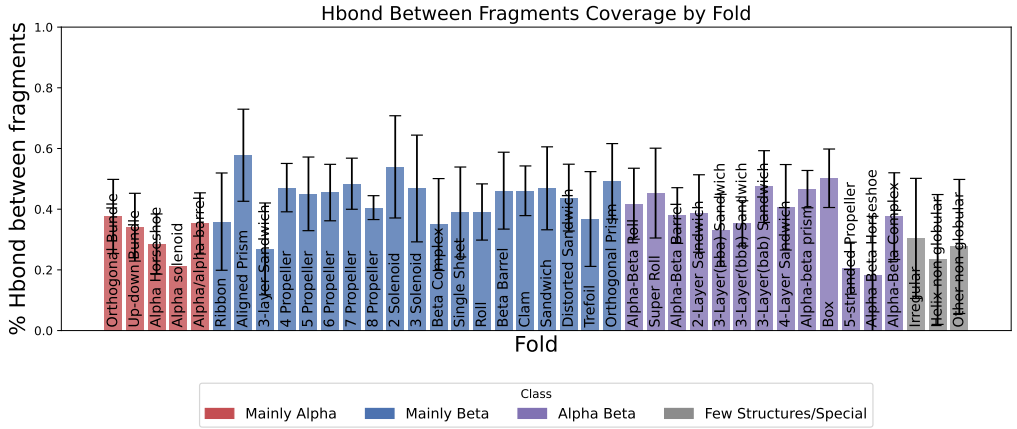



Figure S8: H-bond Between Fragments Coverage by Fold.

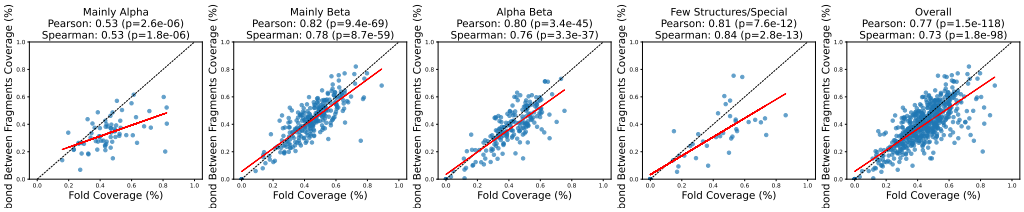



Figure S9: H-bond Between Fragments vs. Fold Coverage.

### 10.3 H-bond Within Fragments Coverage

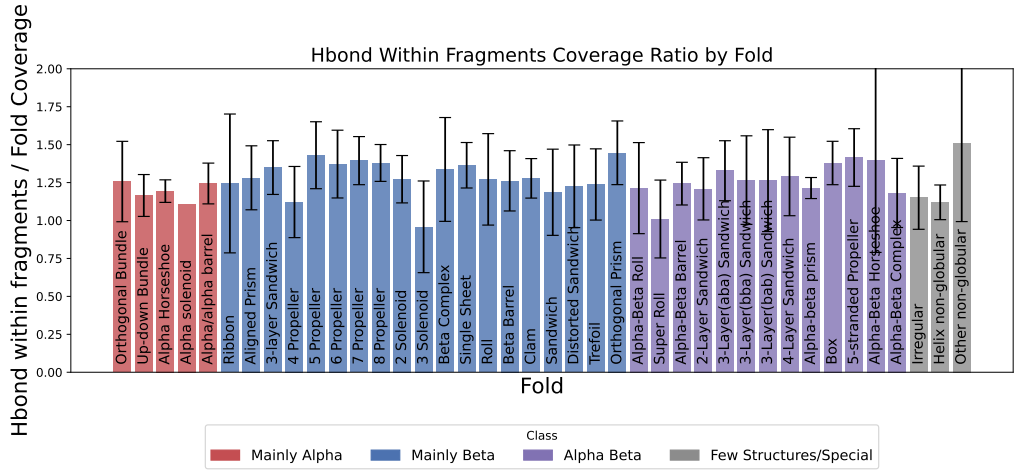



Figure S10: H-bond Within Fragments Coverage Ratio by Fold.

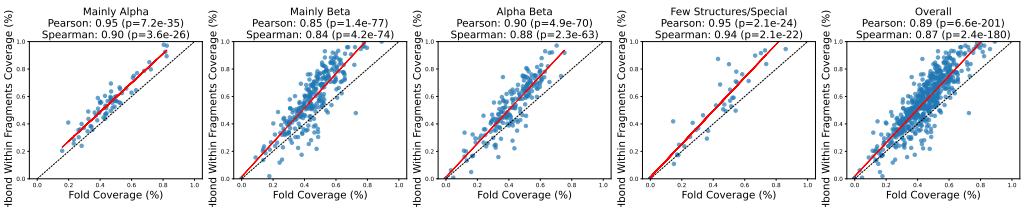



Figure S11: H-bond Within Fragments vs. Fold Coverage.

## 10.4 Accessibility Coverage

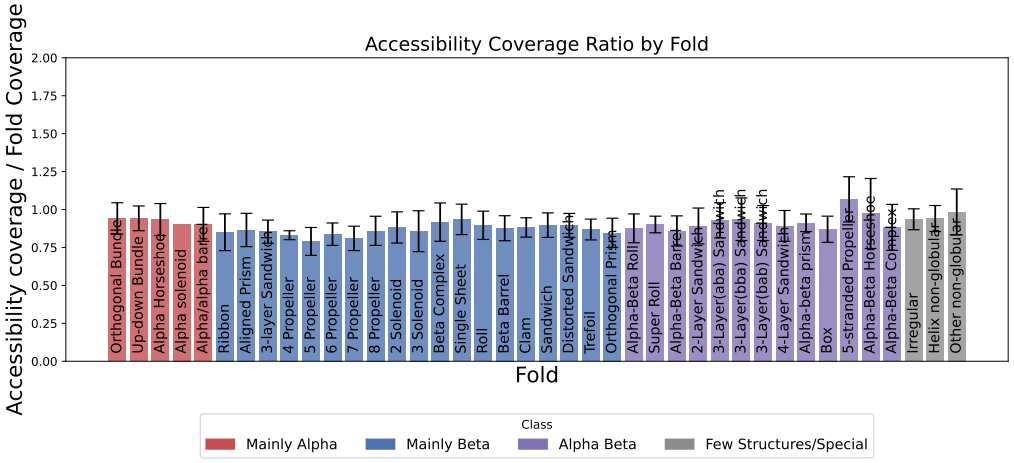



Figure S12: Accessibility Coverage Ratio by Fold.

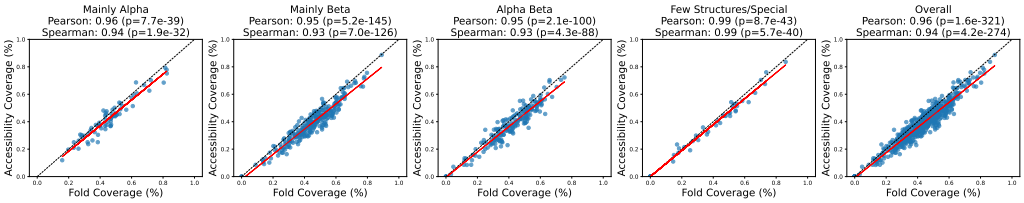



Figure S13: Accessibility vs. Fold Coverage.

## 10.5 Charge Coverage

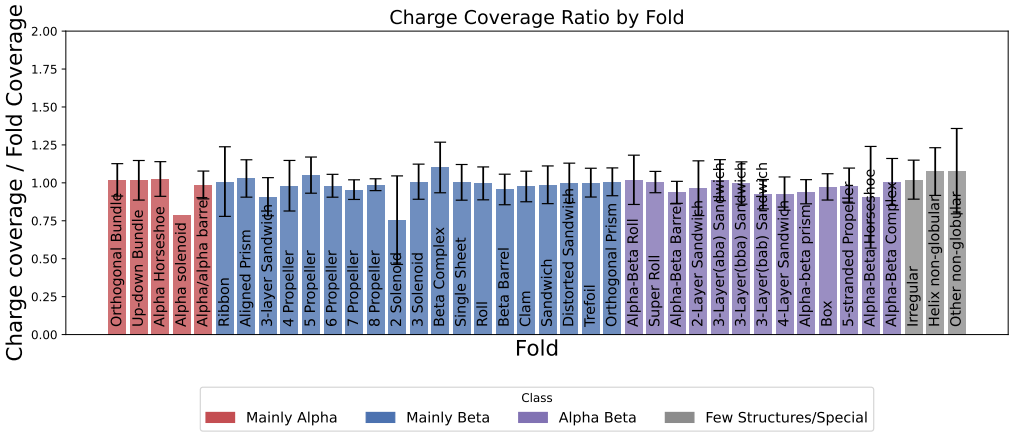



Figure S14: Charge Coverage Ratio by Fold.

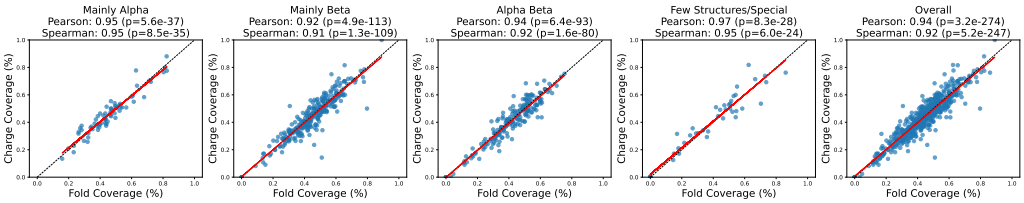



Figure S15: Charge vs. Fold Coverage.

## 10.6 Polarity Coverage

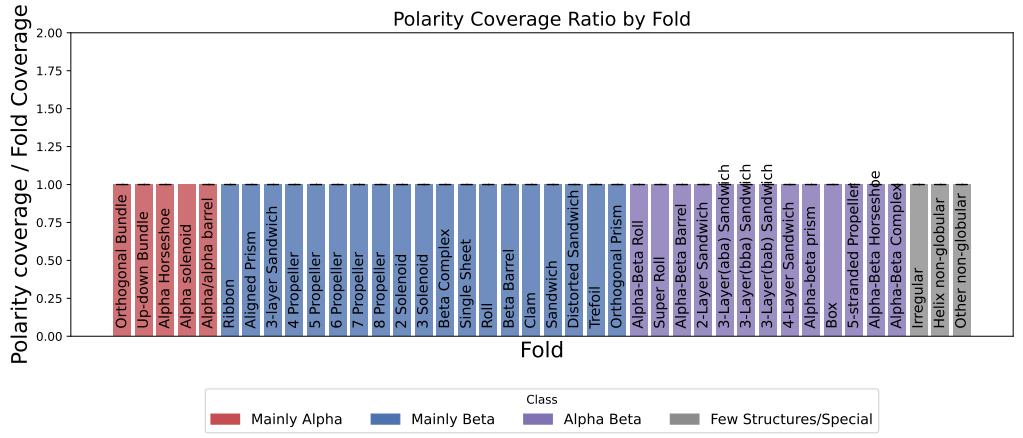



Figure S16: Polarity Coverage Ratio by Fold.

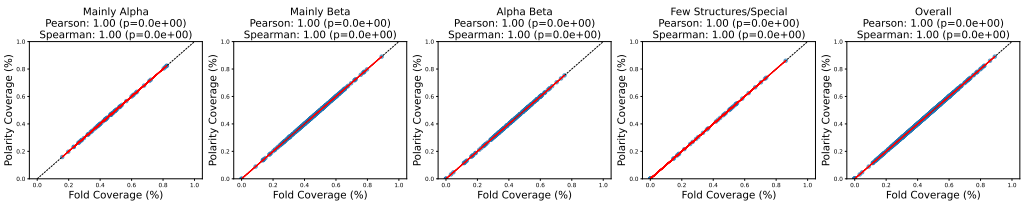



Figure S17: Polarity vs. Fold Coverage.

## 10.7 Secondary Structure Coverage

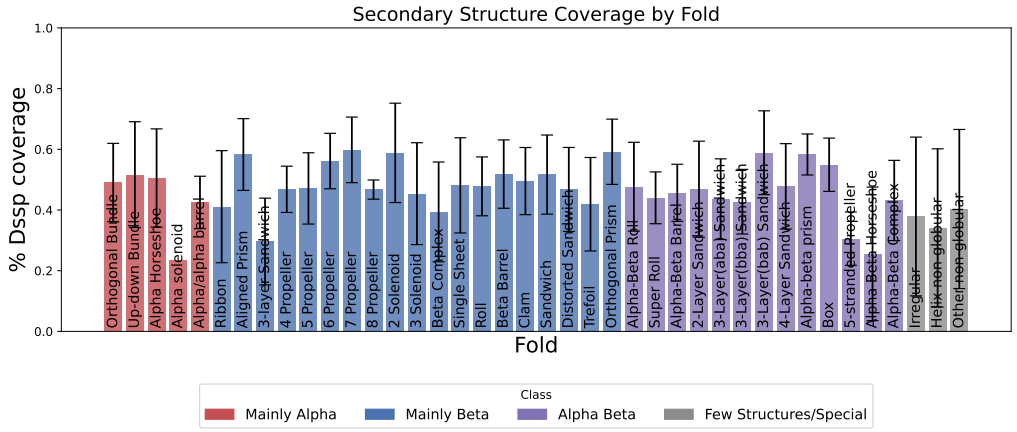



Figure S18: Secondary Structure Ratio by Fold.

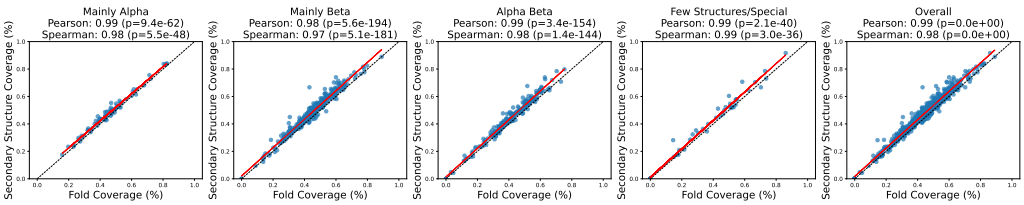



Figure S19: Secondary Structure vs. Fold Coverage.

## 11 Design Success Rate

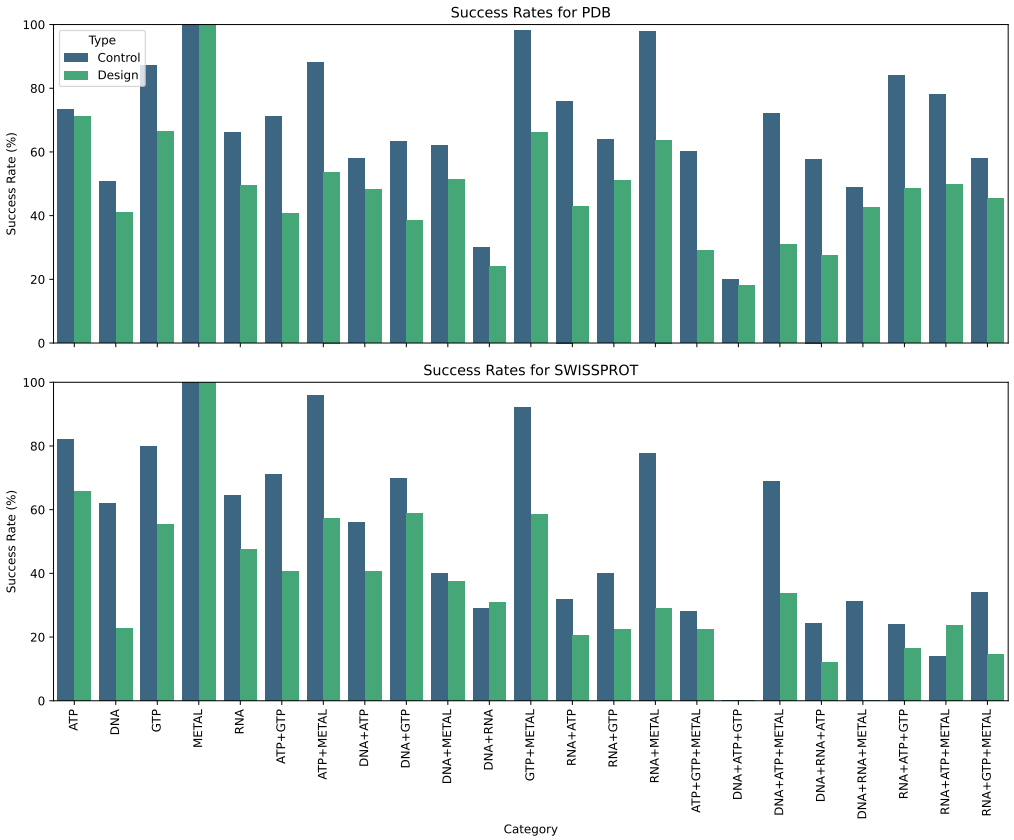



Figure S20: Success rates for fragment-constrained backbone generation. Each protein was used as a template for generating five backbone designs, guided by detected functional fragments. The success rate is defined as the proportion of generated backbones whose top 10 structural matches in FoldSeek share the same Gene Ontology (GO) function as the original protein.

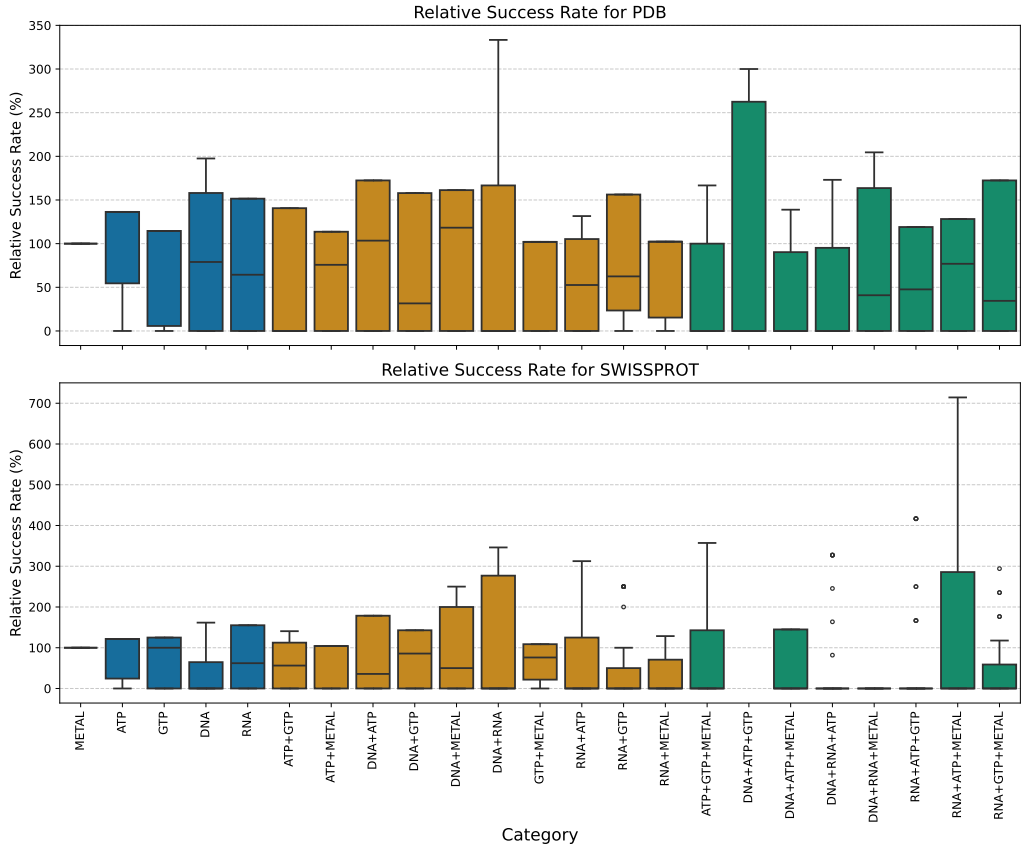



Figure S21: Boxplot of relative success rates for fragment-constrained backbone generation. The relative success rate compares the recovery of functionally similar proteins between generated backbones and their original templates. Values greater than 1 indicate improved functional recovery over the original structure, while values below 1 suggest reduced functional specificity in the generated designs.

## 12 Data Point Distributions of Fragments and traditional methods

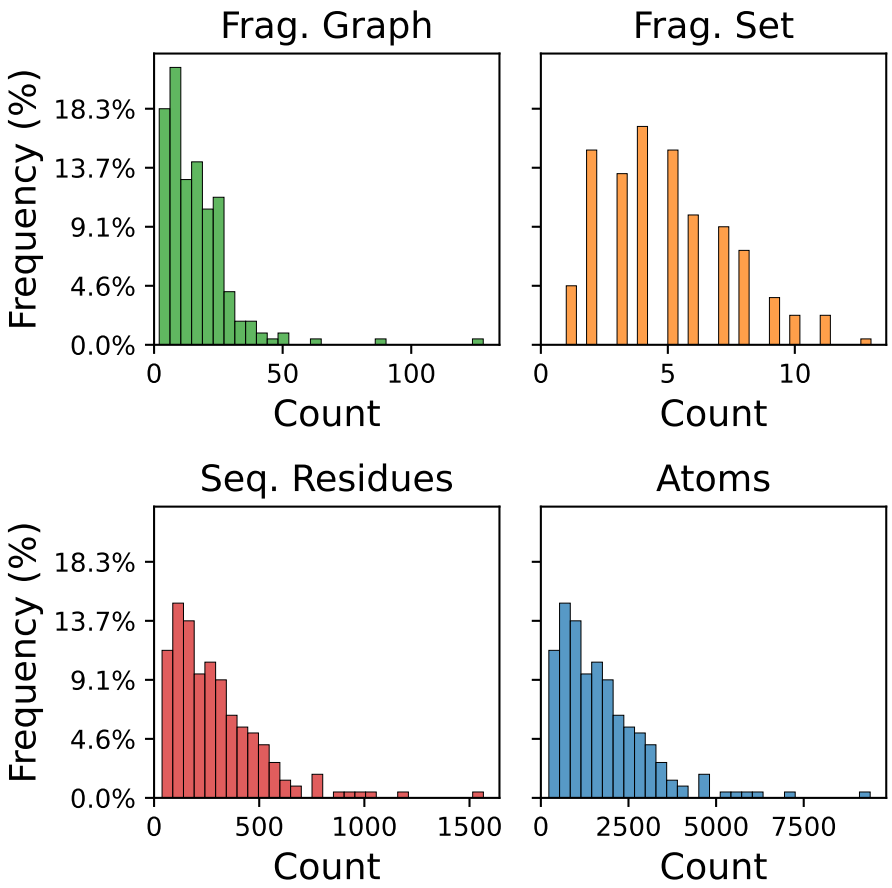



Figure S22: Distribution of the number of data points required to represent proteins using different methods: Fragment Nodes (Fragment Graphs), Fragment Sets, Residues (Sequence-based representation), and Atoms (Shape-based representation). The figure illustrates how different levels of abstraction affect the number of data points required for each representation, highlighting variations in data complexity across methods.

## References


[1] Leonardo V Castorina, Rokas Petrenas, Kartic Subr, and Christopher W Wood. PDBench: Evaluating computational methods for protein-sequence design. *Bioinformatics*, 39(1):btad027, January 2023.

[2] Peter J. A. Cock, Tiago Antao, Jeffrey T. Chang, Brad A. Chapman, Cymon J. Cox, Andrew Dalke, Iddo Friedberg, Thomas Hamelryck, Frank Kauff, Bartek Wilczynski, and Michiel J. L. de Hoon. Biopython: Freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423, June 2009.

# Bibliography

[1] Ahdritz, G., Bouatta, N., Floristean, C., Kadyan, S., Xia, Q., Gerecke, W., O'Donnell, T. J., Berenberg, D., Fisk, I., Zanichelli, N., Zhang, B., Nowaczynski, A., Wang, B., Stepniewska-Dziubinska, M. M., Zhang, S., Ojewole, A., Guney, M. E., Biderman, S., Watkins, A. M., Ra, S., Lorenzo, P. R., Nivon, L., Weitzner, B., Ban, Y.-E. A., Chen, S., Zhang, M., Li, C., Song, S. L., He, Y., Sorger, P. K., Mostaque, E., Zhang, Z., Bonneau, R., and AlQuraishi, M. (2024). OpenFold: Retraining AlphaFold2 yields new insights into its learning mechanisms and capacity for generalization. *Nature Methods*, 21(8):1514–1524.

[2] Alamdari, S., Thakkar, N., Van Den Berg, R., Tenenholtz, N., Strome, R., Moses, A. M., Lu, A. X., Fusi, N., Amini, A. P., and Yang, K. K. (2023). Protein generation with evolutionary diffusion: Sequence is all you need.

[3] Albanese, K. I., Barbe, S., Tagami, S., Woolfson, D. N., and Schiex, T. (2025). Computational protein design. *Nature Reviews Methods Primers*, 5(1):13.

[4] Alberts, B. (2015). *Molecular Biology of the Cell*. Garland Science, Taylor and Francis Group, New York, NY, sixth edition edition.

[5] AlQuraishi, M. (2019). AlphaFold at CASP13. *Bioinformatics*, 35(22):4862–4865.

[6] Alva, V. and Lupas, A. N. (2018). From ancestral peptides to designed proteins. *Current Opinion in Structural Biology*, 48:103–109.

[7] Alva, V., Söding, J., and Lupas, A. N. (2015). A vocabulary of ancient peptides at the origin of folded proteins. *eLife*, 4:e09410.

[8] Anand, N., Eguchi, R., and Huang, P.-S. (2019). Fully differentiable full-atom protein backbone generation.

[9] Anand, N. and Huang, P. (2018). Generative modeling for protein structures. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett,

R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

[10] Anfinsen, C. B. (1973). Principles that Govern the Folding of Protein Chains. *Science*, 181(4096):223–230.

[11] Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Davis, A. P., Dolinski, K., Dwight, S. S., Eppig, J. T., Harris, M. A., Hill, D. P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J. C., Richardson, J. E., Ringwald, M., Rubin, G. M., and Sherlock, G. (2000). Gene Ontology: Tool for the unification of biology. *Nature Genetics*, 25(1):25–29.

[12] Baker, D. (2019). What has de novo protein design taught us about protein folding and biophysics? *Protein Science*, 28(4):678–683.

[13] Bennett, N. R., Coventry, B., Goreshnik, I., Huang, B., Allen, A., Vafeados, D., Peng, Y. P., Dauparas, J., Baek, M., Stewart, L., DiMaio, F., De Munck, S., Savvides, S. N., and Baker, D. (2023). Improving de novo protein binder design with deep learning. *Nature Communications*, 14(1):2625.

[14] Berman, H. M. (2000). The Protein Data Bank. *Nucleic Acids Research*, 28(1):235–242.

[15] Bertoline, L. M. F., Lima, A. N., Krieger, J. E., and Teixeira, S. K. (2023). Before and after AlphaFold2: An overview of protein structure prediction. *Frontiers in Bioinformatics*, 3:1120370.

[16] Biewald, L. (2020). Experiment tracking with weights and biases.

[17] Bonet, J., Wehrle, S., Schriever, K., Yang, C., Billet, A., Sesterhenn, F., Scheck, A., Sverrisson, F., Veselkova, B., Vollers, S., Lourman, R., Villard, M., Rosset, S., Krey, T., and Correia, B. E. (2018). Rosetta FunFolDes – A general framework for the computational design of functional proteins. *PLOS Computational Biology*, 14(11):e1006623.

[18] Bracewell, R. N. (1966). The fourier transform and its applications.

[19] Brooks, B. R., Brooks, C. L., Mackerell, A. D., Nilsson, L., Petrella, R. J., Roux, B., Won, Y., Archontis, G., Bartels, C., Boresch, S., Caflisch, A., Caves, L., Cui, Q., Dinner, A. R., Feig, M., Fischer, S., Gao, J., Hodoscek, M., Im, W., Kuczera, K.,

Lazaridis, T., Ma, J., Ovchinnikov, V., Paci, E., Pastor, R. W., Post, C. B., Pu, J. Z., Schaefer, M., Tidor, B., Venable, R. M., Woodcock, H. L., Wu, X., Yang, W., York, D. M., and Karplus, M. (2009). CHARMM: The biomolecular simulation program. *Journal of Computational Chemistry*, 30(10):1545–1614.

[20] Cao, L., Coventry, B., Goreshnik, I., Huang, B., Sheffler, W., Park, J. S., Jude, K. M., Marković, I., Kadam, R. U., Verschueren, K. H. G., Verstraete, K., Walsh, S. T. R., Bennett, N., Phal, A., Yang, A., Kozodoy, L., DeWitt, M., Picton, L., Miller, L., Strauch, E.-M., DeBouver, N. D., Pires, A., Bera, A. K., Halabiya, S., Hammerson, B., Yang, W., Bernard, S., Stewart, L., Wilson, I. A., Ruohola-Baker, H., Schlessinger, J., Lee, S., Savvides, S. N., Garcia, K. C., and Baker, D. (2022). Design of protein-binding proteins from the target structure alone. *Nature*, 605(7910):551–560.

[21] Cao, L., Goreshnik, I., Coventry, B., Case, J. B., Miller, L., Kozodoy, L., Chen, R. E., Carter, L., Walls, A. C., Park, Y.-J., Strauch, E.-M., Stewart, L., Diamond, M. S., Veesler, D., and Baker, D. (2020). De novo design of picomolar SARS-CoV-2 miniprotein inhibitors. *Science*, 370(6515):426–431.

[22] Carugo, O. and Pongor, S. (2001). A normalized root-mean-spuare distance for comparing protein three-dimensional structures. *Protein Science*, 10(7):1470–1473.

[23] Case, D. A., Cheatham, T. E., Darden, T., Gohlke, H., Luo, R., Merz, K. M., Onufriev, A., Simmerling, C., Wang, B., and Woods, R. J. (2005). The Amber biomolecular simulation programs. *Journal of Computational Chemistry*, 26(16):1668–1688.

[24] Castorina, L. V., Petrenas, R., Subr, K., and Wood, C. W. (2023). PDBench: Evaluating computational methods for protein-sequence design. *Bioinformatics*, 39(1):btad027.

[25] Castorina, L. V., Ünal, S. M., Subr, K., and Wood, C. W. (2024). TIMED-Design: Flexible and accessible protein sequence design with convolutional neural networks. *Protein Engineering, Design and Selection*, 37:gzae002.

[26] Castorina, L. V., Wood, C. W., and Subr, K. (2025). From atoms to fragments: A coarse representation for functional and efficient protein design. *bioRxiv : the preprint server for biology*.

[27] Castro, K. M., Scheck, A., Xiao, S., and Correia, B. E. (2022). Computational design of vaccine immunogens. *Current Opinion in Biotechnology*, 78:102821.

[28] Chai Discovery, Boitreaud, J., Dent, J., McPartlon, M., Meier, J., Reis, V., Rogozhnikov, A., and Wu, K. (2024). Chai-1: Decoding the molecular interactions of life.

[29] Chandonia, J.-M., Guan, L., Lin, S., Yu, C., Fox, N. K., and Brenner, S. E. (2022). SCOPe: Improvements to the structural classification of proteins – extended database to facilitate variant interpretation and machine learning. *Nucleic Acids Research*, 50(D1):D553–D559.

[30] Chothia, C. and Lesk, A. (1986). The relation between the divergence of sequence and structure in proteins. *The EMBO Journal*, 5(4):823–826.

[31] Cock, P. J. A., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B., and de Hoon, M. J. L. (2009). Biopython: Freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423.

[32] Correia, B. E., Bates, J. T., Loomis, R. J., Baneyx, G., Carrico, C., Jardine, J. G., Rupert, P., Correnti, C., Kalyuzhniy, O., Vittal, V., Connell, M. J., Stevens, E., Schroeter, A., Chen, M., MacPherson, S., Serra, A. M., Adachi, Y., Holmes, M. A., Li, Y., Klevit, R. E., Graham, B. S., Wyatt, R. T., Baker, D., Strong, R. K., Crowe, J. E., Johnson, P. R., and Schief, W. R. (2014). Proof of principle for epitope-focused vaccine design. *Nature*, 507(7491):201–206.

[33] Cotet, T.-S., Krawczuk, I., Stocco, F., Ferruz, N., Gitter, A., Kurumida, Y., de Almeida Machado, L., Paesani, F., Calia, C. N., Challacombe, C. A., et al. (2025). Crowdsourced protein design: Lessons from the adaptyv egfr binder competition. *bioRxiv*, pages 2025–04.

[34] Darwin, C. (2011). *The Origin of Species*. HarperCollins, London.

[35] Dauparas, J., Anishchenko, I., Bennett, N., Bai, H., Ragotte, R. J., Milles, L. F., Wicky, B. I. M., Courbet, A., De Haas, R. J., Bethel, N., Leung, P. J. Y., Huddy, T. F., Pellock, S., Tischer, D., Chan, F., Koepnick, B., Nguyen, H., Kang, A., Sankaran, B., Bera, A. K., King, N. P., and Baker, D. (2022a). Robust deep learning–based protein sequence design using ProteinMPNN. *Science*, 378(6615):49–56.

[36] Dauparas, J., Anishchenko, I., Bennett, N., Bai, H., Ragotte, R. J., Milles, L. F., Wicky, B. I. M., Courbet, A., de Haas, R. J., Bethel, N., Leung, P. J. Y., Huddy, T. F., Pellock, S., Tischer, D., Chan, F., Koepnick, B., Nguyen, H., Kang, A., Sankaran, B., Bera, A. K., King, N. P., and Baker, D. (2022b). Robust deep learning based protein sequence design using ProteinMPNN.

[37] Dawson, N. L., Lewis, T. E., Das, S., Lees, J. G., Lee, D., Ashford, P., Orengo, C. A., and Sillitoe, I. (2017). CATH: An expanded resource to predict protein function through structure and sequence. *Nucleic Acids Research*, 45(Database issue):D289–D295.

[38] Defresne, M., Barbe, S., and Schiex, T. (2021). Protein Design with Deep Learning. *International Journal of Molecular Sciences*, 22(21):11741.

[39] Dill, K. A. and Chan, H. S. (1997). From Levinthal to pathways to funnels. *Nature Structural & Molecular Biology*, 4(1):10–19.

[40] Doerr, S., Harvey, M. J., Noé, F., and De Fabritiis, G. (2016). Htmd: High-throughput molecular dynamics for molecular discovery. *Journal of Chemical Theory and Computation*, 12(4):1845–1852.

[41] Dürr, S. L., Levy, A., and Rothlisberger, U. (2023). Metal3d: a general deep learning framework for accurate metal ion location prediction in proteins. *Nature Communications*, 14(1).

[42] Eaton, W. A. (2021). Modern Kinetics and Mechanism of Protein Folding: A Retrospective. *The Journal of Physical Chemistry B*, 125(14):3452–3467.

[43] Eguchi, R. R., Choe, C. A., and Huang, P.-S. (2022). Ig-VAE: Generative modeling of protein structure by direct 3D coordinate generation. *PLOS Computational Biology*, 18(6):e1010271.

[44] Elnaggar, A., Heinzinger, M., Dallago, C., Rehawi, G., Wang, Y., Jones, L., Gibbs, T., Feher, T., Angerer, C., Steinegger, M., Bhowmik, D., and Rost, B. (2022). ProtTrans: Toward Understanding the Language of Life Through Self-Supervised Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):7112–7127.

[45] Ferraz, M. V. F., Adan, W. C. S., Lima, T. E., Santos, A. J. C., De Paula, S. O., Dhalia, R., Wallau, G. L., Wade, R. C., Viana, I. F. T., and Lins, R. D. (2024). Design

of nanobody targeting SARS-CoV-2 spike glycoprotein using CDR-grafting assisted by molecular simulation and machine learning.

[46] Ferruz, N., Heinzinger, M., Akdel, M., Goncearenco, A., Naef, L., and Dallago, C. (2023). From sequence to function through structure: Deep learning for protein design. *Computational and Structural Biotechnology Journal*, 21:238–250.

[47] Ferruz, N., Michel, F., Lobos, F., Schmidt, S., and Höcker, B. (2021). Fuzzle 2.0: Ligand binding in natural protein building blocks. *Frontiers in Molecular Biosciences*, 8.

[48] Ferruz, N., Schmidt, S., and Höcker, B. (2022). ProtGPT2 is a deep unsupervised language model for protein design. *Nature Communications*, 13(1):4348.

[49] Ferruz, N. and Stein, A. (2024). Computational methods for protein design. *Protein Engineering, Design and Selection*, 37:gzae011.

[50] Fox, N. K., Brenner, S. E., and Chandonia, J.-M. (2015). The value of protein structure classification information—Surveying the scientific literature. *Proteins: Structure, Function, and Bioinformatics*, 83(11):2025–2038.

[51] Frappier, V., Jenson, J. M., Zhou, J., Grigoryan, G., and Keating, A. E. (2019). Tertiary Structural Motif Sequence Statistics Enable Facile Prediction and Design of Peptides that Bind Anti-apoptotic Bfl-1 and Mcl-1. *Structure*, 27(4):606–617.e5.

[52] Gainza, P., Sverrisson, F., Monti, F., Rodolà, E., Boscaini, D., Bronstein, M. M., and Correia, B. E. (2020). Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. *Nature Methods*, 17(2):184–192.

[53] Gasser, H.-C., Oyarzún, D. A., Rajan, A., and Alfaro, J. A. (2024). Guiding a language-model based protein design method towards MHC Class-I immune-visibility targets in vaccines and therapeutics. *ImmunoInformatics*, 14:100035.

[54] Grassmann, G., Di Rienzo, L., Gosti, G., Leonetti, M., Ruocco, G., Miotto, M., and Milanetti, E. (2023). Electrostatic complementarity at the interface drives transient protein-protein interactions. *Scientific Reports*, 13(1).

[55] Gront, D., Kulp, D. W., Vernon, R. M., Strauss, C. E. M., and Baker, D. (2011). Generalized Fragment Picking in Rosetta: Design, Protocols and Applications. *PLoS ONE*, 6(8):e23294.

[56] Guo, X., Du, Y., Tadepalli, S., Zhao, L., and Shehu, A. (2021). Generating tertiary protein structures via interpretable graph variational autoencoders. *Bioinformatics Advances*, 1(1):vbab036.

[57] Hayes, T., Rao, R., Akin, H., Sofroniew, N. J., Oktay, D., Lin, Z., Verkuil, R., Tran, V. Q., Deaton, J., Wiggert, M., Badkundri, R., Shafkat, I., Gong, J., Derry, A., Molina, R. S., Thomas, N., Khan, Y. A., Mishra, C., Kim, C., Bartie, L. J., Nemeth, M., Hsu, P. D., Sercu, T., Candido, S., and Rives, A. (2025). Simulating 500 million years of evolution with a language model. *Science*, 387(6736):850–858.

[58] Heinzelman, P., Romero, P. A., and Arnold, F. H. (2013). *Efficient Sampling of SCHEMA Chimera Families to Identify Useful Sequence Elements*, page 351–368. Elsevier.

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

[60] Henzler-Wildman, K. and Kern, D. (2007). Dynamic personalities of proteins. *Nature*, 450(7172):964–972.

[61] Hill, C. P., Anderson, D. H., Wesson, L., DeGrado, W. F., and Eisenberg, D. (1990). Crystal Structure of $A_1$ : Implications for Protein Design. *Science*, 249(4968):543–546.

[62] Ho, S. P. and DeGrado, W. F. (1987). Design of a 4-helix bundle protein: Synthesis of peptides which self-associate into a helical protein. *Journal of the American Chemical Society*, 109(22):6751–6758.

[63] Höcker, B. (2014). Design of proteins from smaller fragments—learning from evolution. *Current Opinion in Structural Biology*, 27:56–62.

[64] Huang, X., Pearce, R., and Zhang, Y. (2020). EvoEF2: Accurate and fast energy function for computational protein design. *Bioinformatics (Oxford, England)*, 36(4):1135–1142.

[65] Ingraham, J. B., Baranov, M., Costello, Z., Barber, K. W., Wang, W., Ismail, A., Frappier, V., Lord, D. M., Ng-Thow-Hing, C., Van Vlack, E. R., Tie, S., Xue, V., Cowles, S. C., Leung, A., Rodrigues, J. V., Morales-Perez, C. L., Ayoub, A. M.,

Green, R., Puentes, K., Oplinger, F., Panwar, N. V., Obermeyer, F., Root, A. R., Beam, A. L., Poelwijk, F. J., and Grigoryan, G. (2023). Illuminating protein space with a programmable generative model. *Nature*, 623(7989):1070–1078.

[66] Jacobs, T. M., Williams, B., Williams, T., Xu, X., Eletsky, A., Federizon, J. F., Szyperski, T., and Kuhlman, B. (2016). Design of structurally distinct proteins using strategies inspired by evolution. *Science*, 352(6286):687–690.

[67] Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., Back, T., Petersen, S., Reiman, D., Clancy, E., Zielinski, M., Steinegger, M., Pacholska, M., Berghammer, T., Bodenstein, S., Silver, D., Vinyals, O., Senior, A. W., Kavukcuoglu, K., Kohli, P., and Hassabis, D. (2021). Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589.

[68] Jung, S., Bae, S.-E., Ahn, I., and Son, H. S. (2013). Protein Backbone Torsion Angle-Based Structure Comparison and Secondary Structure Database Web Server. *Genomics & Informatics*, 11(3):155.

[69] Khakzad, H., Igashov, I., Schneuing, A., Goverde, C., Bronstein, M., and Correia, B. (2023). A new age in protein design empowered by deep learning. *Cell Systems*, 14(11):925–939.

[70] Kolodny, R., Nepomnyachiy, S., Tawfik, D. S., and Ben-Tal, N. (2021). Bridging Themes: Short Protein Segments Found in Different Architectures. *Molecular Biology and Evolution*, 38(6):2191–2208.

[71] Kuhlman, B., Dantas, G., Ireton, G. C., Varani, G., Stoddard, B. L., and Baker, D. (2003). Design of a Novel Globular Protein Fold with Atomic-Level Accuracy. *Science*, 302(5649):1364–1368.

[72] Kuriata, A., Iglesias, V., Pujols, J., Kurcinski, M., Kmiecik, S., and Ventura, S. (2019). Aggrescan3d (a3d) 2.0: prediction and engineering of protein solubility. *Nucleic Acids Research*, 47(W1):W300–W307.

[73] Levinthal, C. (1968). Are there pathways for protein folding? *Journal de Chimie Physique*, 65:44–45.

[74] Lewandowski, J. R., Halse, M. E., Blackledge, M., and Emsley, L. (2015). Direct observation of hierarchical protein dynamics. *Science*, 348(6234):578–581.

[75] Limozin, L., Bongrand, P., and Robert, P. (2016). A rough energy landscape to describe surface-linked antibody and antigen bond formation. *Scientific Reports*, 6(1).

[76] Lin, M., Chen, Q., and Yan, S. (2013). Network In Network.

[77] Lin, Z., Akin, H., Rao, R., Hie, B., Zhu, Z., Lu, W., Smetanin, N., Verkuil, R., Kabeli, O., Shmueli, Y., Dos Santos Costa, A., Fazel-Zarandi, M., Sercu, T., Candido, S., and Rives, A. (2023). Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130.

[78] Ludwiczak, J., Jarmula, A., and Dunin-Horkawicz, S. (2018). Combining Rosetta with molecular dynamics (MD): A benchmark of the MD-based ensemble protein design. *Journal of Structural Biology*, 203(1):54–61.

[79] Lupas, A. N. (2014). What I cannot create, I do not understand. *Science*, 346(6216):1455–1456.

[80] Mackenzie, C. O., Zhou, J., and Grigoryan, G. (2016). Tertiary alphabet for the observable protein structural universe. *Proceedings of the National Academy of Sciences*, 113(47).

[81] Mastrolorito, F., Ciriaco, F., Togo, M. V., Gambacorta, N., Trisciuzzi, D., Altomare, C. D., Amoroso, N., Grisoni, F., and Nicolotti, O. (2025). fragSMILES as a chemical string notation for advanced fragment and chirality representation. *Communications Chemistry*, 8(1):26.

[82] McIntosh-Smith, S., Price, J., Sessions, R. B., and Ibarra, A. A. (2014). High performance in silico virtual drug screening on many-core processors. *The International Journal of High Performance Computing Applications*, 29(2):119–134.

[83] McPartlon, M., Lai, B., and Xu, J. (2022). A Deep SE(3)-Equivariant Model for Learning Inverse Protein Folding.

[84] Merlicek, L. P., Neumann, J., Lear, A., Degiorgi, V., De Waal, M., Cotet, T.-S., Mulholland, A. J., and Bunzel, H. A. (2025). AI.zymes – A modular platform for evolutionary enzyme design.

[85] Mirdita, M., Schütze, K., Moriwaki, Y., Heo, L., Ovchinnikov, S., and Steinegger, M. (2022). ColabFold: Making protein folding accessible to all. *Nature Methods*, 19(6):679–682.

[86] Mistry, J., Chuguransky, S., Williams, L., Qureshi, M., Salazar, G. A., Sonnhammer, E. L. L., Tosatto, S. C. E., Paladin, L., Raj, S., Richardson, L. J., Finn, R. D., and Bateman, A. (2021). Pfam: The protein families database in 2021. *Nucleic Acids Research*, 49(D1):D412–D419.

[87] Moult, J., Pedersen, J. T., Judson, R., and Fidelis, K. (1995). A large-scale experiment to assess protein structure prediction methods. *Proteins: Structure, Function, and Bioinformatics*, 23(3).

[88] Munsamy, G., Illanes-Vicioso, R., Funcillo, S., Nakou, I. T., Lindner, S., Ayres, G., Sheehan, L. S., Moss, S., Eckhard, U., Lorenz, P., and Ferruz, N. (2024). Conditional language models enable the efficient design of proficient enzymes.

[89] O'Connell, J., Li, Z., Hanson, J., Heffernan, R., Lyons, J., Paliwal, K., Dehzangi, A., Yang, Y., and Zhou, Y. (2018). SPIN2: Predicting sequence profiles from protein structures using deep neural networks. *Proteins: Structure, Function, and Bioinformatics*, 86(6):629–633.

[90] Pacesa, M., Nickel, L., Schellhaas, C., Schmidt, J., Pyatova, E., Kissling, L., Barendse, P., Choudhury, J., Kapoor, S., Alcaraz-Serna, A., Cho, Y., Ghamary, K. H., Vinué, L., Yachnin, B. J., Wollacott, A. M., Buckley, S., Westphal, A. H., Lindhoud, S., Georgeon, S., Goverde, C. A., Hatzopoulos, G. N., Gönczy, P., Muller, Y. D., Schwank, G., Swarts, D. C., Vecchio, A. J., Schneider, B. L., Ovchinnikov, S., and Correia, B. E. (2024). BindCraft: One-shot design of functional protein binders.

[91] Peri, C., Morra, G., and Colombo, G. (2016). Surface energetics and protein-protein interactions: analysis and mechanistic implications. *Scientific Reports*, 6(1).

[92] Pokkuluri, P., Raffen, R., Dieckman, L., Boogaard, C., Stevens, F., and Schiffer, M. (2002). Increasing protein stability by polar surface residues: Domain-wide consequences of interactions within a loop. *Biophysical Journal*, 82(1):391–398.

[93] Qi, Y. and Zhang, J. Z. H. (2020). DenseCPD: Improving the Accuracy of Neural-Network-Based Computational Protein Sequence Design with DenseNet. *Journal of Chemical Information and Modeling*, 60(3):1245–1252.

[94] Ramachandran, G., Ramakrishnan, C., and Sasisekharan, V. (1963). Stereochemistry of polypeptide chain configurations. *Journal of Molecular Biology*, 7(1):95–99.

[95] Regan, L. and Clarke, N. D. (1990). A tetrahedral zinc(II)-binding site introduced into a designed protein. *Biochemistry*, 29(49):10878–10883.

[96] Richardson, J. S. (2000). Early ribbon drawings of proteins. *Nature Structural Biology*, 7(8):624–625.

[97] Rives, A., Meier, J., Sercu, T., Goyal, S., Lin, Z., Liu, J., Guo, D., Ott, M., Zitnick, C. L., Ma, J., and Fergus, R. (2019). Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences of the United States of America*.

[98] Roseman, A. M. (2000). Docking structures of domains into maps from cryo-electron microscopy using local correlation. *Acta Crystallographica Section D Biological Crystallography*, 56(10):1332–1340.

[99] Röthlisberger, D., Khersonsky, O., Wollacott, A. M., Jiang, L., DeChancie, J., Betker, J., Gallaher, J. L., Althoff, E. A., Zanghellini, A., Dym, O., Albeck, S., Houk, K. N., Tawfik, D. S., and Baker, D. (2008). Kemp elimination catalysts by computational enzyme design. *Nature*, 453(7192):190–195.

[100] Schrödinger, LLC (2015). The PyMOL molecular graphics system, version 1.8.

[101] Schulz, G. E. and Schirmer, R. H. (1979). *Principles of Protein Structure*. Springer Advanced Texts in Chemistry. Springer New York, New York, NY.

[102] Sesterhenn, F., Yang, C., Bonet, J., Cramer, J. T., Wen, X., Wang, Y., Chiang, C.-I., Abriata, L. A., Kucharska, I., Castoro, G., Vollers, S. S., Galloux, M., Dheilly, E., Rosset, S., Corthésy, P., Georgeon, S., Villard, M., Richard, C.-A., Descamps, D., Delgado, T., Oricchio, E., Rameix-Welti, M.-A., Más, V., Ervin, S., Eléouët, J.-F., Riffault, S., Bates, J. T., Julien, J.-P., Li, Y., Jardetzky, T., Krey, T., and Correia, B. E. (2020). De novo protein design enables the precise induction of RSV-neutralizing antibodies. *Science*, 368(6492):eaay5051.

[103] Shrimpton-Phoenix, E., Notari, E., Kluonis, T., and Wood, C. W. (2024). drmd: Molecular dynamics for experimentalists. *Journal of Molecular Biology*, page 168918.

[104] Stam, M. J. and Wood, C. W. (2021). DE-STRESS: A user-friendly web application for the evaluation of protein designs. *Protein Engineering, Design and Selection*, 34:gzab029.

[105] Stocco, F., Artigues-Lleixa, M., Hunklinger, A., Widatalla, T., Guell, M., and Ferruz, N. (2024). Guiding generative protein language models with reinforcement learning.

[106] Stollar, E. J. and Smith, D. P. (2020). Uncovering protein structure. *Essays in Biochemistry*, 64(4):649–680.

[107] Strokach, A., Becerra, D., Corbi-Verge, C., Perez-Riba, A., and Kim, P. M. (2020). Fast and Flexible Protein Design Using Deep Graph Neural Networks. *Cell Systems*, 11(4):402–411.e4.

[108] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2014). Going Deeper with Convolutions.

[109] Thumuluri, V., Martiny, H.-M., Almagro Armenteros, J. J., Salomon, J., Nielsen, H., and Johansen, A. R. (2022). NetSolP: Predicting protein solubility in *Escherichia coli* using language models. *Bioinformatics*, 38(4):941–946.

[110] Tompson, J., Goroshin, R., Jain, A., LeCun, Y., and Bregler, C. (2014). Efficient Object Localization Using Convolutional Networks.

[111] Van Kempen, M., Kim, S. S., Tumescheit, C., Mirdita, M., Lee, J., Gilchrist, C. L. M., Söding, J., and Steinegger, M. (2024). Fast and accurate protein structure search with Foldseek. *Nature Biotechnology*, 42(2):243–246.

[112] Vascon, F., Gasparotto, M., Giacomello, M., Cendron, L., Bergantino, E., Filippini, F., and Righetto, I. (2020). Protein electrostatics: From computational and structural analysis to discovery of functional fingerprints and biotechnological design. *Computational and Structural Biotechnology Journal*, 18:1774–1789.

[113] Vopson, M. M. (2021). Estimation of the information contained in the visible matter of the universe. *AIP Advances*, 11(10):105317.

[114] Wang, G. and Dunbrack, R. L. (2003). PISCES: A protein sequence culling server. *Bioinformatics*, 19(12):1589–1591.

[115] Wang, H., Liu, D., Zhao, K., Wang, Y., and Zhang, G. (2024). SPDesign: Protein sequence designer based on structural sequence profile using ultrafast shape recognition. *Briefings in Bioinformatics*, 25(3):bbae146.

[116] Watson, J. L., Juergens, D., Bennett, N. R., Trippe, B. L., Yim, J., Eisenach, H. E., Ahern, W., Borst, A. J., Ragotte, R. J., Milles, L. F., Wicky, B. I. M., Hanikel, N., Pellock, S. J., Courbet, A., Sheffler, W., Wang, J., Venkatesh, P., Sappington, I., Torres, S. V., Lauko, A., De Bortoli, V., Mathieu, E., Ovchinnikov, S., Barzilay, R., Jaakkola, T. S., DiMaio, F., Baek, M., and Baker, D. (2023). De novo design of protein structure and function with RFdiffusion. *Nature*, 620(7976):1089–1100.

[117] Weininger, D. (1988). SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 28(1):31–36.

[118] Weiss, M. (2010). Biomolecular crystallography: Principles, practice, and applications to structural biology. by bernhard rupp. new york: Garland science, taylor and francis group, 2010. pp. xxi + 809. price (hardback) usd 145.00. isbn 978-0-8153-4081-2. *Acta Crystallographica Section D Biological Crystallography*, 66(5):640–641.

[119] Winnifrith, A., Outeiral, C., and Hie, B. L. (2024). Generative artificial intelligence for de novo protein design. *Current Opinion in Structural Biology*, 86:102794.

[120] Woolfson, D. N. (2021). A Brief History of De Novo Protein Design: Minimal, Rational, and Computational. *Journal of Molecular Biology*, 433(20):167160.

[121] Wu, J.-N., Wang, T., Chen, Y., Tang, L.-J., Wu, H.-L., and Yu, R.-Q. (2024). T-SMILES: A fragment-based molecular representation framework for de novo ligand design. *Nature Communications*, 15(1):4993.

[122] Wu, R., Ding, F., Wang, R., Shen, R., Zhang, X., Luo, S., Su, C., Wu, Z., Xie, Q., Berger, B., Ma, J., and Peng, J. (2022). High-resolution de novo structure prediction from primary sequence. *bioRxiv*, page 2022.07.21.500999.

[123] Yeh, A. H.-W., Norn, C., Kipnis, Y., Tischer, D., Pellock, S. J., Evans, D., Ma, P., Lee, G. R., Zhang, J. Z., Anishchenko, I., Coventry, B., Cao, L., Dauparas, J., Halabiya, S., DeWitt, M., Carter, L., Houk, K. N., and Baker, D. (2023). De novo design of luciferases using deep learning. *Nature*, 614(7949):774–780.

[124] Zhang, Y. (2005). TM-align: A protein structure alignment algorithm based on the TM-score. *Nucleic Acids Research*, 33(7):2302–2309.

[125] Zhang, Y., Chen, Y., Wang, C., Lo, C.-C., Liu, X., Wu, W., and Zhang, J. (2020). ProDCoNN: Protein design using a convolutional neural network. *Proteins: Structure, Function, and Bioinformatics*, 88(7):819–829.

[126] Zheng, Z., Zhang, B., Zhong, B., Liu, K., Li, Z., Zhu, J., Yu, J., Wei, T., and Chen, H.-F. (2024). Scaffold-Lab: Critical Evaluation and Ranking of Protein Backbone Generation Methods in A Unified Framework.

[127] Zhou, H.-X. and Pang, X. (2018). Electrostatic interactions in protein structure, folding, binding, and condensation. *Chemical Reviews*, 118(4):1691–1741.

[128] Zhou, J., Panaitiu, A. E., and Grigoryan, G. (2020). A general-purpose protein design framework based on mining sequence–structure relationships in known protein structures. *Proceedings of the National Academy of Sciences*, 117(2):1059–1068.

[129] Zimmerman, J., Eliezer, N., and Simha, R. (1968). The characterization of amino acid sequences in proteins by statistical methods. *Journal of Theoretical Biology*, 21(2):170–201.