Language Models towards Conditional Generative Modelsof Proteins Sequences

Barthélémy Meynard-Piganeau Meynard

Language Models towards Conditional Generative Models of Proteins Sequences

Barthelemy Meynard-Piganeau

Sous la direction de Martin Weigt et Riccardo Zecchina

Jury:

Directeur du jury:	M. Elodie Laine
Rapporteurs:	Mme. Armita Nourmohammad, M. David Gfeller
Examinateurs:	M. Sergei Grudinin, M. David Bikard
Directeurs:	M. Martin Weigt, M. Riccardo Zecchina

Réalisée au Laboratoire de Biologie Computationnelle et Quantitative

Spécialité Informatique, École doctorale 130: Informatique, Télécommunications et électronique (Paris)

Abstract

English This thesis explores the intersection of artificial intelligence (AI) and biology, focusing on how generative models can innovate in protein sequence design. Our research unfolds in three distinct yet interconnected stages, each building upon the insights of the previous to enhance the model's applicability and performance in protein engineering.

We begin by examining what makes a generative model effective for protein sequences. In our first study, "Interpretable Pairwise Distillations for Generative Protein Sequence Models," we compare complex neural network models to simpler, pairwise distribution models. This comparison highlights that deep learning strategy mainly models second-order interactions, highlighting their fundamental role in modeling protein families.

In a second part, we try to expand this principle of using second-order interaction to inverse folding. We explore structure conditioning in "Uncovering Sequence Diversity from a Known Protein Structure". Here, we present InvMSAFold, a method that produces diverse protein sequences designed to fold into a specific structure. This approach tries to combine two different traditions of protein modeling: the MSA-based models that try to capture the entire fitness landscape, and the inverse folding types of model that focus on recovering one specific sequence. This is a first step towards the possibility of conditioning the fitness landscape by considering the protein's final structure in the design process, enabling the generation of sequences that are not only diverse but also maintain their intended structural integrity.

Finally, we delve into sequence conditioning with "Generating Interacting Protein Sequences using Domain-to-Domain Translation." This study introduces a novel approach to generate protein sequences that can interact with specific other proteins. By treating this as a translation problem, similar to methods used in language processing, we create sequences with intended functionalities. Furthermore, we address the critical challenge of T-cell receptor (TCR) and epitope interaction prediction in "TULIP—a Transformer based Unsupervised Language model for Interacting Peptides and T-cell receptors." This study introduces an unsupervised learning approach to accurately predict TCR-epitope bindings, overcoming limitations in data quality and training bias inherent in previous models. These advancements underline the potential of sequence conditioning in creating functionally specific and interaction-aware protein designs.

Français Nous commençons par examiner ce qui rend un modèle génératif efficace pour les séquences de protéines. Dans notre première étude, "Interpretable Pairwise Distillations for Generative Protein Sequence Models" nous comparons les modèles de réseaux de neurones complexes à des modèles de distributions pair à pair plus simples. Cette comparaison révèle que les modèles plus simples peuvent égaler de près la performance des modèles plus complexes dans la prédiction de l'effet des

mutations sur les protéines. Cette découverte remet en question l'hypothèse selon laquelle les modèles plus complexes sont toujours meilleurs, préparant le terrain pour de plus amples explorations.

Dans une seconde partie, nous nous penchons sur le conditionnement de séquence avec "Generating Interacting Protein Sequences using Domain-to-Domain Translation" Cette étude introduit une approche novatrice pour générer des séquences de protéines qui peuvent interagir avec d'autres protéines spécifiques. En traitant cela comme un problème de traduction, similaire aux méthodes utilisées dans le traitement du langage naturel, nous créons des séquences avec des fonctionnalités intentionnelles. De plus, nous abordons le défi crucial de la prédiction de l'interaction entre le récepteur des cellules T (TCR) et l'épitope dans "TULIP—a Transformer based Unsupervised Language model for Interacting Peptides and T-cell receptors" Cette étude introduit une approche d'apprentissage non supervisée pour prédire avec précision les liaisons TCR-épitope, surmontant les limitations de qualité des données et les biais de formation inhérents aux modèles précédents. Ces avancées soulignent le potentiel du conditionnement de séquence dans la création de designs de protéines fonctionnellement spécifiques et conscients de l'interaction.

Enfin, nous explorons le conditionnement de structure dans "Uncovering Sequence Diversity from a Known Protein Structure". Ici, nous présentons InvMSAFold, une méthode qui produit des séquences de protéines diverses conçues pour se plier dans une structure spécifique. Cette approche met en lumière l'importance de considérer la structure finale de la protéine dans le processus de conception, permettant la génération de séquences qui sont non seulement diverses mais maintiennent également leur intégrité structurelle prévue.

Remerciements

Tout d'abord, j'aimerais remercier tous les étudiants de mon groupe avec qui j'ai passé cette période de la thèse, en particulier Jeanne, Matteo, Sabrina, Francesco avec qui j'ai passé des moments inoubliables à Cuba, ainsi que les nouveaux étudiants: Lorenzo Alya et Roberto. Finalement je remercie mes camarades de Milan, en particular Lisa, Veronica, Patrick, Luca et Paolo. Un merci particulier pour Cristina qui m'a appris l'italien et Pierre pour soutien moral. Ensuite je dois remercier mes superviseurs, en particulier Martin qui a su me soutneir dans les moments les plus difficiles, et Riccardo pour ma fabuleuse experience italienne. Je dois aussi remercier chaleureusement Christoph Feinauer pour son aide énorme et la collaboration que j'ai eu avec lui tout au long de ma these. Son influence a été decisive dans le bon deroulement de ma thèse. Finalement, je termine ces remrciements par l'équipe de Gistave Roussy: Gregoire Baptiste, Anne, Jhen, Theo et les autres pour leur aide et accueil.

Contents

Chapter 1

Introduction

1.1 Overview of Protein Engineering and Design

1.1.1 Introduction to Proteins and Their Importance in Biology

Proteins are fundamental components of all living organisms, serving as the building blocks of cells and playing critical roles in virtually every biological process. Composed of long chains of amino acids, proteins fold into specific three-dimensional structures that determine their function. These versatile molecules are involved in a vast array of biological functions, including catalyzing metabolic reactions as enzymes, providing structural support to cells and tissues, mediating cell signaling and communication, and defending the organism against pathogens. The diversity and complexity of proteins underpin the biological mechanisms that sustain life, making them a central subject of study in biology and a key target for therapeutic and biotechnological applications.

The Synthesis of Proteins: The Role of DNA and RNA The synthesis of proteins is a complex process. The DNA encodes the genetic information to produce each protein. This process can be divided into two main stages: transcription and translation, involving the synthesis of an intermediary molecule, the RNA.

Transcription (From DNA to RNA): DNA contains the instructions for protein synthesis. During transcription, a segment of DNA is copied into messenger RNA (mRNA) by the RNA polymerase. This mRNA serves as a portable copy of the genetic instructions.

Translation (From RNA to Protein): The mRNA is then processed by the ribosome, the cell's protein factory. Here, transfer RNA (tRNA) molecules bring amino acids to the ribosome, where they are added to the growing polypeptide chain according to the sequence of the mRNA. The specific order of amino acids, determined by the mRNA template, leads to the protein's unique structure and function.

Proteins as amino acids chains Proteins are built from 20 different amino acids, each defined by a unique side chain attached to a core structure. This core consists of

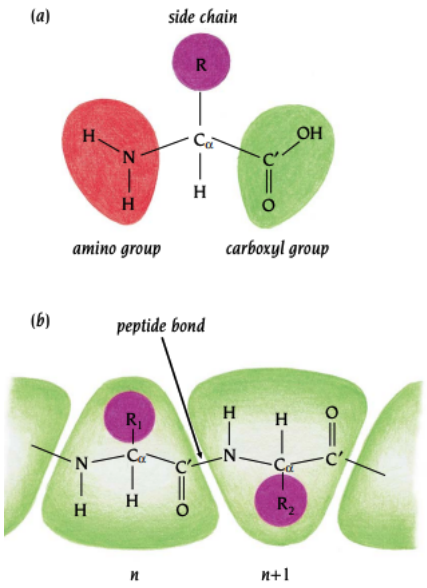


Figure 1.1: Representation of the link between two amino acids. The repetition of this link between several amino acids forms the protein macromolecule. Figure from [1]

a central carbon atom bonded to a hydrogen atom, an amino group (NH2), and a carboxyl group (COOH). The variety in amino acids comes from their distinct side chains, known as R-groups, which are also attached to the central carbon. This diversity allows for a wide range of proteins that perform various functions in living organisms.

The assembly of proteins occurs through the formation of peptide bonds, linking amino acids together. This linkage results from the carboxyl group of one amino acid condensing with the amino group of another, thereby creating a continuous chain. The outcome is a polypeptide chain with distinct amino and carboxyl termini. This chain's backbone is composed of the Cα atom, an NH group, and a C'=O carbonyl group, with peptide bonds connecting the carbonyl carbon (C') of one amino acid to the nitrogen of the next (see 1.1).

Classification of Amino Acids Based on the properties of their side chains (see Fig. 1.2), amino acids can be categorized into groups such as hydrophobic, hydrophilic (polar), and charged. Hydrophobic amino acids tend to be found in the interior of proteins, stabilizing the structure, whereas hydrophilic and charged amino acids are likely to be located on the surface, interacting with the aqueous environment or other molecules. Some amino acids, like phenylalanine or tryptophan, have aromatic ring structures that can interact with each other. These aromatic side chains can be bulky or more compact depending on their structure. Other amino acids, like histidine and leucine, have large, bulky side chains that influence protein folding and interactions. In contrast, glycine and alanine have smaller side chains, allowing

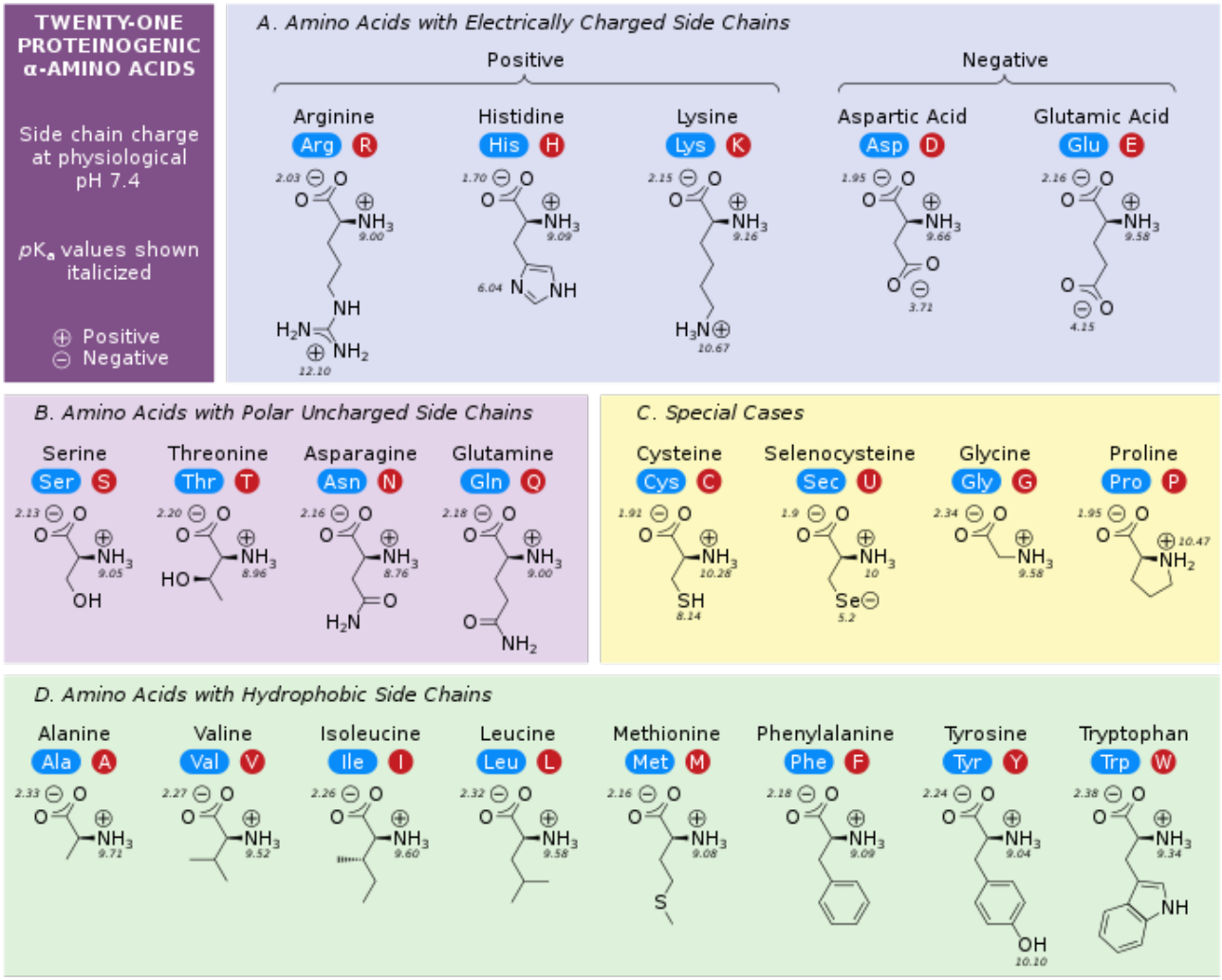


Figure 1.2: Classification of amino acids according to their chemical properties, from `https://en.wikipedia.org/wiki/Amino_acid`

for tighter packing within the protein.

Protein Structure and Function The function of a protein is intricately linked to its structure, which can be described at four levels:

The primary structure is the linear sequence of amino acids. The secondary structure is composed of localized folding patterns such as α-helices and β-sheets, formed by hydrogen bonding. The tertiary structure is the three-dimensional conformation of a single polypeptide chain, determined by various interactions among amino acids. Finally, the quaternary structure is the assembly of multiple polypeptide chains into a functional protein complex.

Importance of Protein-Protein Interactions with some examples Protein-protein interactions are pivotal in nearly every process within a biological cell, facilitating the formation of complex signaling networks and structural assemblies. These interactions are essential for the cell to carry out its functions properly, including cell signaling, immune responses, and metabolic control.

Signal Transduction: This process involves the transmission of chemical or physical signals from a cell's exterior to its interior, leading to a cellular response. These signaling pathways are achieved by a chain of proteins phosphorylating each other. They control cell growth, division, and death, playing critical roles in maintaining cellular health and responding to environmental changes.

Immune Response: Protein-protein interactions are crucial in the immune system, where antibodies and T-Cell Receptor (TCR) bind to specific piece of a pathogen protein called antigens, and various immune cells communicate to coordinate an immune response. These interactions help the body recognize and respond to pathogens effectively.

Metabolic Pathways: Enzymes, which are proteins, are sometimes assembled in complexes to carry out metabolic processes [2]. The interaction between these enzymes allows for the regulation of metabolic pathways, ensuring that essential substances are synthesized and broken down as needed.

Regulatory Complexes: Proteins often regulate the activity of other proteins through interactions that can either activate or inhibit their functions. This regulation is crucial for controlling the cell cycle, gene expression, and protein degradation, among other processes.

Understanding protein-protein interactions is therefore essential for deciphering the complex web of cellular processes.

1.1.2 Historical Perspective on Protein Engineering

Proteins orchestrate a vast array of biological functions critical to life. Their functional diversity, intricately encoded within their amino acid sequences, renders proteins indispensable for various processes in living organisms. The goal of protein

design is to engineer novel proteins, aiming to either enhance existing functionalities or introduce entirely new capabilities.

The vastness of the potential protein design space, however, presents an immediate challenge. Given that the number of possible amino acid sequences far exceeds the number of atoms in the universe (for example, for a protein of 100 amino acids we get 20^{100} possibilities), identifying sequences that have desired functions becomes a needle-in-a-haystack problem. This immense design space is sparsely populated with functionally relevant sequences, making the search for efficacious proteins highly non-trivial. Indeed the mapping from this space to its functionality, known as the 'fitness landscape', is unknown and hard to model. The quest to navigate and model this landscape necessitates strategies that effectively narrow the search to a more manageable subset of the vast possibilities.

Historically, approaches to protein design have spanned from rational design, leveraging deep insights into protein structure and function, to experimental methods such as directed evolution and combinatorial libraries, which explore a wider array of variants. The development of biophysics-based computational models has further augmented our capacity to predict protein structure, folding, and interactions, marking a significant advancement in the computational design domain.

Rational Design Minimal design was the first attempt to design proteins. It utilizes straightforward patterns of hydrophobic and polar residues to influence the folding and assembly of proteins in water, aiming to create basic protein-like structures [3].

As the field matured, it shifted towards more sophisticated rational design strategies. Rational design in protein engineering combines a deep understanding of protein structure and function with sophisticated computational and experimental methods. This approach leverages high-resolution structural data, such as X-ray crystallography or NMR spectroscopy, to identify and modify key residues. It builds on minimal design strategies that use simple chemical principles for protein folding, enhancing specificity and functionality through detailed biochemical and bioinformatics insights. While rational design has significantly advanced the field [3], enabling the creation of enzymes with novel activities [4], and the introduction of metal-binding sites into protein structures [5], it faces challenges. These include the complexity of accurately predicting the effects of sequence modifications and the potential evolutionary bias in consensus approaches. Despite these limitations, rational design has been instrumental in creating functional proteins that bridge natural protein architectures and unexplored "dark matter" of protein structure space, offering a powerful toolkit for expanding the functional repertoire of proteins.

Direct Evolution Directed evolution, pioneered by Frances H. Arnold [6], circumvents the need for detailed structural knowledge by mimicking the process of natural selection in the laboratory. This technique involves generating a library of protein variants via mutagenesis, subjecting them to a selection process that favors the de-

sired activity, and iterating this process to evolve proteins with enhanced properties. Directed evolution has played a crucial role in developing proteins with new functions, such as enzymes capable of catalyzing reactions not found in nature [7]. This approach's success lies in its ability to explore the sequence spaces around a starting point. Indeed the improvement needs to be local, making it particularly efficient for optimization. It can however be labor-intensive and requires efficient methods for screening large numbers of variants.

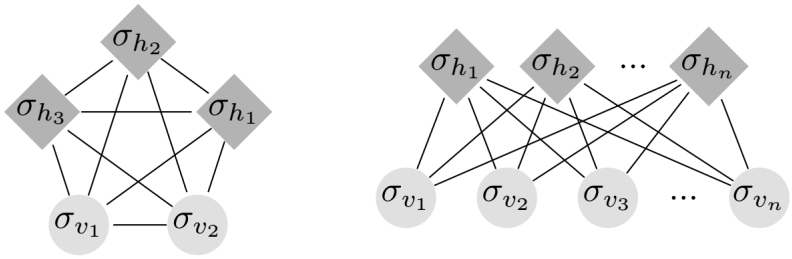
Computational Design in the Pre Deep Learning Era A variety of computational methods were instrumental in pioneering the field, and despite the rise of new deep learning approaches, they remain heavily used. These tools addressed various aspects of the design process.

Rosetta stands out for its comprehensive suite of algorithms tailored for protein design. [8–10] It facilitates the design of novel protein structures, interfaces, and functions by employing an all-atom energy function and sophisticated sampling methods. RosettaDesign [11], a component of Rosetta, allows researchers to redesign protein cores and surfaces, enabling the creation of proteins with new functions or improved stability. Its ability to model protein-protein interactions through RosettaDock [12] has also been crucial for designing synthetic biological systems.

FoldX [13, 14] was developed as a force field for predicting the effect of mutations on protein stability and protein-protein interaction affinities. Its fast and quantitative assessments made it a valuable tool for protein engineering, where predicting the impacts of amino acid substitutions is crucial for designing proteins with desired properties. FoldX has been applied to improve enzyme catalysis or design protein-protein interactions.

1.2 Generative Models: A Machine Learning Perspective

Generative modeling has evolved significantly since the 1980s, with applications ranging from image synthesis to drug discovery. Central to generative models is the concept of creating a probability distribution over the data space that mimics the training data distribution. Early models like energy-based models faced scalability issues with complex data, but recent advancements in deep learning and data availability have led to breakthroughs in efficiency. Additionally, self-supervised learning, closely related to generative modeling, focuses on learning representations for unsupervised downstream tasks, enhancing the versatility of generative models. "representation" refers to the way in which data is encoded by a model to capture essential features that are useful for making predictions or decisions. It means encoding data in different directions to capture diverse and higher-level features from the data, enabling them to discern more abstract patterns and relationships.



(a) Boltzmann machine. (b) Restricted Boltzmann machine.

Figure 1.3: Representation of the graph underlying the energy function of the Boltzmann machine and the Restricted Boltzmann Machine. A specific unit of σ is either hidden or visible, represented by the σ_h and σ_v in the figures.

1.2.1 Energy Based Models

Energy-based models (EBMs) utilize an energy function $E(x) : \mathbb{X} \to \mathbb{R}$ to map input data x from the data space $\mathbb{X}$ to a real number, associating low energy values with realistic data points and high values with unrealistic ones. This approach is formalized by expressing any probability density function $p(x)$ as a function of the energy of x, given by the equation:

$$p(x) = \frac{e^{-E(x)}}{\int_{\tilde{x}\in\mathbb{X}} e^{-E(\tilde{x})}}.$$

EBMs offer several advantages, such as simplicity in training, feature sharing which minimizes parameters, and elimination of assumptions about the data distribution. Despite their appeal, scaling EBMs to high-dimensional data remains challenging, although recent developments have shown promise. A notable challenge in EBMs is optimization, particularly due to the intractable nature of the equation's denominator.

The Boltzmann Machine (BM) and its variant, the Restricted Boltzmann Machine (RBM), are cornerstone models in statistical learning and generative modeling. Both are energy-based models that learn to represent complex data distributions through a network of units (nodes) that represent stochastic variables. These models define a probability distribution over binary vectors of fixed dimensions, making them powerful tools for understanding and generating data.

Boltzmann Machine (BM)

A Boltzmann Machine [15] is a type of energy-based model, based on a fully connected graph (see Fig. 1.3 .a). The energy of a state $\sigma = (\sigma_1, ..., \sigma_N) \in \{\pm 1\}^N$ in a BM is defined by the equation:

$$E(\sigma) = -\sum_{i<j} J_{ij}(\sigma_i, \sigma_j) - \sum_i h_i(\sigma_i),$$

where J_{ij} represents the weight of the connection between units i and j, σ_i is the state of unit i, and h_i is the bias term for unit i. σ_i can either be a visible or a hidden variable. This formulation allows for a broad range of applications in learning and pattern recognition.

The probability of a configuration σ is given by the Boltzmann distribution:

$$p(\sigma) = \frac{e^{-E(\sigma)}}{Z},$$

where Z is the partition function, a normalization term that is the sum of $e^{-E(\sigma)}$ over all possible states σ. The fully connected nature of BMs makes them very expressive but also computationally intensive, especially due to the difficulty in computing Z, and the need for sampling to approximate the gradients during training as explained later 1.2.1.

Restricted Boltzmann Machine (RBM)

The Restricted Boltzmann Machine [16] modifies the structure of the BM by dividing the units into two layers (see Fig. 1.3.b): a visible layer that represents observed data and a hidden layer that captures dependencies between observed variables. Crucially, units within the same layer are not connected. This restriction significantly reduces the computational complexity of the model and allows for more efficient training algorithms. The energy of a state in an RBM is defined as:

$$E(\sigma_{\mathbf{v}}, \sigma_{\mathbf{h}}) = -\mathbf{b}^T \sigma_{\mathbf{v}} - \mathbf{c}^T \sigma_{\mathbf{h}} - \sigma_{\mathbf{v}}^T \mathbf{W} \sigma_{\mathbf{h}},$$

where $\sigma_{\mathbf{h}}$ and $\sigma_{\mathbf{v}}$ represent the state of the hidden units and visible units. The vector $\mathbf{b}$ is the bias for the visible layer. It influences the energy level of the system based on the state of the visible units, effectively acting as an external field that biases the probability of certain states of the visible units. The vector $\mathbf{c}$ is the bias for the hidden layer. Finally $\mathbf{W}$ is the weight matrix between the visible and hidden layers. It encodes the strength and type (excitatory or inhibitory) of the interactions between visible and hidden units.

Training Boltzmann Machines

Training both BMs and RBMs involves adjusting the model parameters to maximize the likelihood of the observed data. This process requires estimating the gradient of the log-likelihood with respect to the parameters. However, exact computation is infeasible due to the partition function Z, leading to the adoption of approximation methods.

Monte Carlo methods, specifically Markov Chain Monte Carlo (MCMC), are used to sample from the model's distribution to estimate gradients. For RBMs, Gibbs sampling is particularly efficient due to the bipartite structure. The conditional independence of a type of node given the other one allows for parallel updating of all hidden units given visible units and vice versa.

A workaround involves using Contrastive Divergence (CD) [17], focusing on decreasing the energy of real data points while increasing it for samples from the model's energy distribution. The process leverages the gradient of the negative log-likelihood loss, approximated as:

$$\nabla_\theta L = \mathbb{E}_{\sigma^+ \sim p_d}[\nabla_\theta E_\theta(\sigma^+)] - \mathbb{E}_{\sigma^- \sim p_\theta}[\nabla_\theta E_\theta(\sigma^-)],$$

where p_d is the training data and σ^- is a sample generated through Markov Chain Monte Carlo (MCMC) processes where the chains are initiated from the training data and run for a small number of steps.

Persistent Contrastive Divergence (PCD) [18] enhances the CD method by maintaining a set of Markov chains that are updated continuously across training iterations, rather than restarting from the training data at each step. Unlike CD, which uses short chains starting anew from training data at each iteration, PCD's persistently updated chains evolve over time, exploring the model's distribution space more thoroughly. This persistent updating allows the chains to more closely approximate the model's equilibrium distribution, leading to improved accuracy in gradient estimates for model parameters.

1.2.2 Variational Auto-Encoders

Variational Autoencoders (VAEs) [19] are a class of generative models that leverage principles from deep learning and Bayesian inference to model complex data distributions. VAEs are particularly notable for their ability to learn latent representations of input data, typically of smaller dimensions, enabling them to generate new data points that resemble the original dataset. The framework of VAEs provides a robust approach to unsupervised learning, making them suitable for a wide range of applications, including image generation, anomaly detection, and more.

Unlike traditional autoencoders that directly learn the encoding and decoding mappings, VAEs introduce a probabilistic twist: they model the encoding as a distribution over the latent space. This approach allows VAEs to capture the underlying probability distribution of the data, and thus to sample new data.

A VAE consists of two main components: the encoder (or recognition model) and the decoder (or generative model), as shown in Fig. 1.4. The encoder maps input data x to a latent representation z through a distribution $q_\phi(z|x)$, where ϕ denotes the parameters of the encoder. The decoder then maps this latent representation back to the data space, attempting to reconstruct the input from the latent code via the distribution $p_\theta(x|z)$, with θ representing the parameters of the decoder.

The objective function of a VAE, derived from the variational lower bound (or evidence lower bound, ELBO), is designed to maximize the likelihood of the data while regularizing the latent space. The ELBO [20] can be formulated as:

$$\mathcal{L}(\phi, \theta; x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x) \| p(z)),$$

where D_{KL} denotes the Kullback-Leibler divergence, a measure of how one probability distribution diverges from a second, expected distribution. The first term of

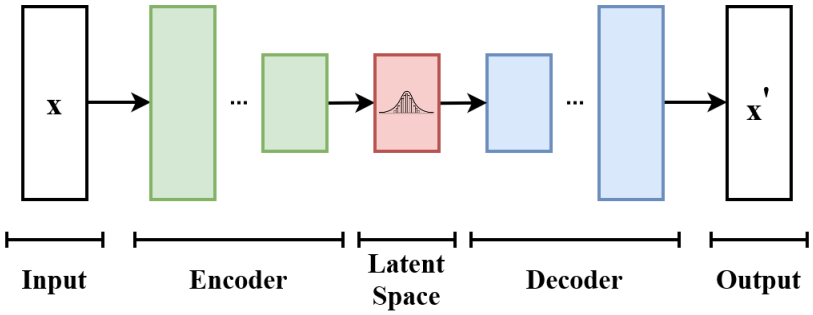


Figure 1.4: Variational Auto Encoder map the input data to a probability distribution in the latent space and then try to reconstruct the original datapoint. Figure from https://en.wikipedia.org/wiki/Variational_autoencoder

the ELBO encourages the reconstructed data to be as close as possible to the original data, while the second term regularizes the encoder by penalizing deviations of the latent code distribution from a prior distribution $p(z)$, typically assumed to be a standard factorized Gaussian distribution.

Training a VAE involves optimizing the ELBO with respect to the parameters of the encoder and decoder, usually achieved via stochastic gradient descent. The optimization seeks to find a balance between fidelity in data reconstruction and regularization of the latent space. An important aspect of VAE training is the "reparameterization trick", which allows gradients to flow through the stochastic sampling of the latent variables, enabling end-to-end training of the model. This trick involves sampling ϵ from a standard normal distribution and computing $z = \mu + \sigma \odot \epsilon$, where μ and σ are the mean and standard deviation of the latent distribution $q_\phi(z|x)$, making the sampling process differentiable.

Once trained, a VAE can generate new data samples by drawing samples from the prior distribution over the latent space $p(z)$ and passing these samples through the decoder. This is enabled by the second term in the ELBO, which forces during training the encoder to encode the data points in a Gaussian latent space. This process leverages the generative capacity of the model to produce data points that mimic the distribution of the original dataset. The quality and diversity of the generated samples are directly influenced by how well the VAE has learned the underlying data distribution and the regularity of the latent space.

1.2.3 Diffusion Models

Alternatives like score matching and denoising diffusion probabilistic models have been developed to bypass the slow training times of traditional EBMs. Score matching, for instance, minimizes the difference between the gradients of the data and the model's log-density functions [21]. This score-matching strategy has inspired denoising models that have shown impressive results lately. Denoising diffusion

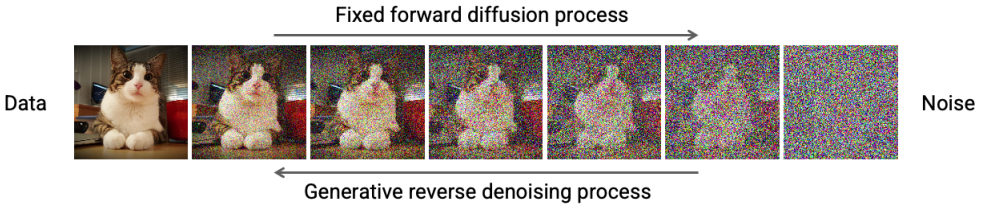


Figure 1.5: Noising and denoising process at the core of Diffusion models. Images from `https://cvpr2022-tutorial-diffusion-models.github.io/`

models incrementally add and then reverse noise to data (see Fig. 1.9), simulating a generative process that can be optimized for better sample generation [22, 23].

Diffusion models are a class of generative models that simulate the gradual process of adding noise to data, followed by a learned reverse process that reconstructs the original data from the noisy version. This process can be described by a Markov chain that incrementally adds Gaussian noise over a series of steps, thereby transforming the original data distribution into a Gaussian distribution. The forward diffusion process is defined as:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I}),$$

where x_t represents the data at time step t, β_t is a variance schedule controlling the noise level, and $\mathcal{N}$ denotes the normal distribution. The reverse process aims to learn the distribution of the original data x_0 given the noisy data x_t, modeled by $p_\theta(x_{t-1}|x_t)$, and is trained by minimizing a variant of the ELBO. This training encourages the model to accurately reconstruct the original data from its noisy versions.

A crucial insight into diffusion models is their ability to transform complex, multimodal data distributions into a simple, tractable Gaussian distribution, from which it becomes easier to sample. The reverse process effectively learns to denoise, capturing the intricate structure of the data distribution as it incrementally removes noise. This method allows diffusion models to generate high-quality samples that closely resemble the original data distribution, making them highly effective for tasks such as image generation [24].

1.2.4 Language Models and Autoregressive modeling

Language models and autoregressive models represent two pivotal facets of modern generative models, especially in the domain of natural language processing (NLP). These models have revolutionized the way machines understand, generate, and interact with human language, offering unprecedented capabilities in text generation, translation, summarization, and more.

Language Modeling Task The core objective of a language model is to predict the probability of a sequence of words (called tokens) in a language. Formally, given a sequence of words $w_1, w_2, ..., w_n$, the language model aims to estimate the probability distribution $P(w_1, w_2, ..., w_n)$. This task is fundamental to various applications in NLP, as it encapsulates the essence of understanding and generating human-like text.

An autoregressive language model decomposes the joint probability of a word sequence into the product of conditional probabilities, sequentially predicting the next word given all the previous ones. The mathematical representation of this decomposition is:

$$p(w_1, w_2, ..., w_n) = \prod_{i=1}^{n} p(w_i | w_1, w_2, ..., w_{i-1}).$$

This autoregressive structure combines two main advantages. Firstly it enables the direct computation of the likelihood of the sequence, as a simple product of token probability. Secondly, it permits fast and efficient sampling by iteratively sampling the next token following their natural ordering.

Transformer Architecture and Attention Mechanism The Transformer architecture [25] has become the standard of modern language models. This architecture eschews recurrence and convolution in favor of attention mechanisms, which dynamically weigh the importance of different words in a sequence when predicting the next word. Unlike traditional approaches that process input sequentially (one element at a time), self-attention enables the model to process all elements of the input sequence in parallel. To be more precise it can directly compute $P(w_i | w_1, w_2, ..., w_{i-1})$ without having to compute $P(w_{i-1} | w_1, w_2, ..., w_{i-2})$ before. This parallel processing allows the model to directly capture relationships between any two elements in the sequence, regardless of their distance from each other.

Self-attention operates on the principle that each element in the input sequence can be a "query" that seeks to find out how much it should attend to other elements, which are "keys". This relationship is quantified by a weight between each position in the sentence. These weights are used to combine the value of each position accordingly.

1. Queries (Q): A representation of an element that is used to query the other elements to determine how much attention to pay to them.
2. Keys (K): Representations of all elements that are used to respond to the queries.
3. Values (V): The actual content of the elements that the queries will focus on, weighted by the attention scores.

The attention mechanism computes a score that reflects how much focus each element (query) should put on every other element (key). This score is calculated using

the dot product of the query with all keys, followed by a scaling factor of $1/\sqrt{d_k}$, where d_k is the dimensionality of the keys. This scaling helps in stabilizing the gradients during training. The scores are then passed through a softmax function to convert them into probabilities (the attention weights).

The core equation of the attention mechanism is given by:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

These attention weights are then used to create a weighted combination of the values, which gives the output of the self-attention mechanism for each element in the sequence. This output is a new representation of the sequence where each element is informed by the entire sequence.

The concept of "heads" in multi-head self-attention refers to the parallel application of the self-attention mechanism, each with different, learnable linear transformations for Q, K, and V. This design allows the model to capture different types of relationships between elements in the sequence simultaneously. Each "head" can be thought of as an independent feature extractor, focusing on different aspects of the information contained in the sequence. The outputs of all heads are then concatenated and linearly transformed to produce the final output. This multiplicity in the attention mechanism significantly enhances the model's ability to understand and generate text, as it can pay attention to multiple facets of the input data at once.

In summary, the Transformer employs a multi-head self-attention mechanism to process sequences in parallel, capturing intricate relationships within the data. This approach enables the model to efficiently handle long sequences with complex dependencies, leading to improved performance in tasks like translation, text generation, and more.

1.3 Artificial Intelligence for Protein Design

1.3.1 Evolution of AI in Protein Engineering

Protein design aims to uncover new protein sequences with a targeted function. This task requires navigating a vast design space, where the functional sequences are a tiny fraction.

Machine learning in protein design focuses on three main objectives [26]: improving existing proteins, creating new functions, or designing new proteins from scratch.

Redesign to Enhance Existing Function The objective is to improve the properties of a protein that already performs a desired function. This includes increasing its main function (e.g., catalytic activity [27], binding affinity), enhancing other attributes (e.g., thermostability [28]), or reducing undesirable interactions (e.g., immunogenicity [29]).

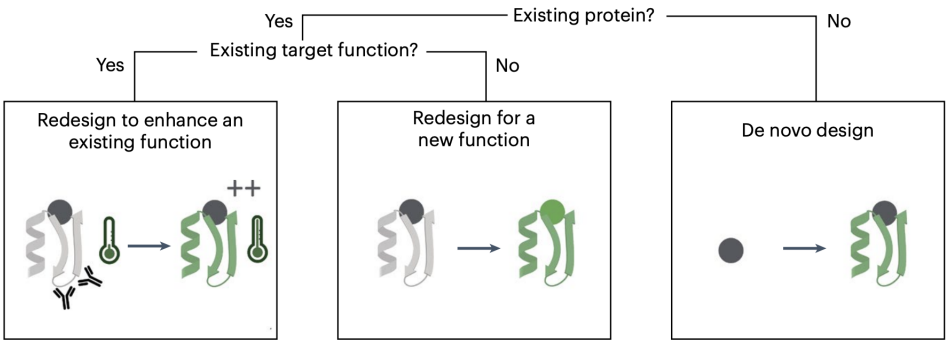


Figure 1.6: There exist three main types of protein design tasks: Redesign to enhance a function, redesign to change the function, and create a completely de novo protein. This figure comes from the review [26]

Redesign for a New Function The objective is to create a protein with a novel function from an existing protein with a related function (for example, shifting a binder or enzyme to act on a new target [30]). This necessitates a comprehensive understanding of the function mechanism or significant data linking sequence to the new function.

De Novo Design The objective is to design protein sequences with entirely new folds. It needs to combine the creation of a new structure and its sequence. It can be used for various purposes like metal-binding and protein-protein interactions or enzymes [31].

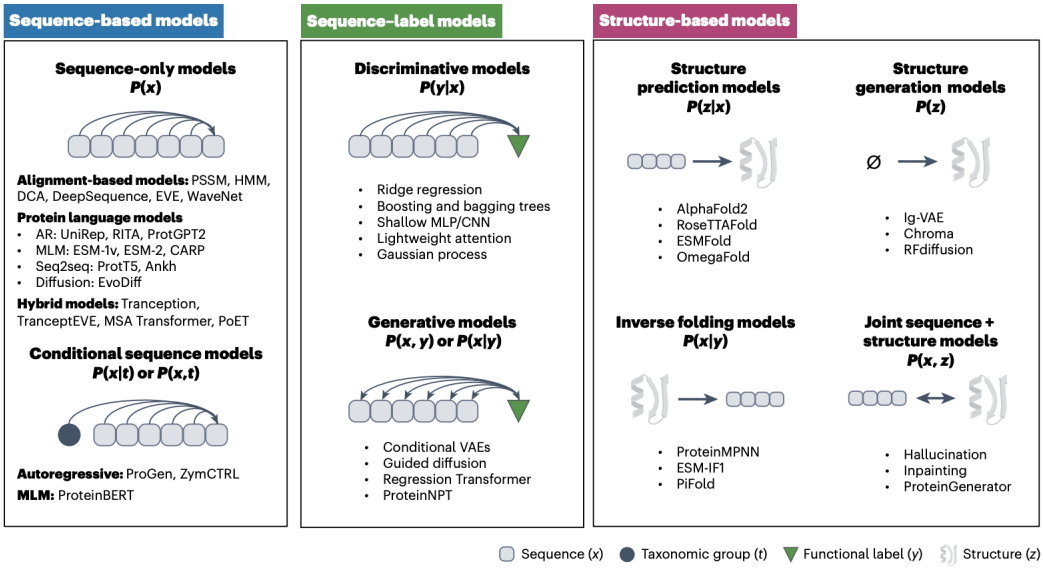


Figure 1.7: Schematic categorization of the different types of Generative models for proteins [26]

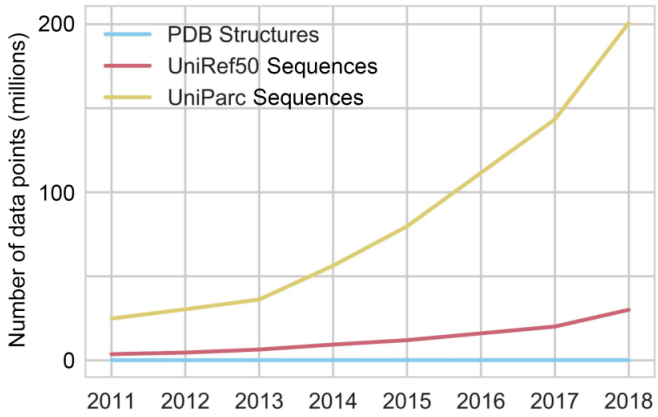


Figure 1.8: Evolution of available number of protein sequences and structures. To estimate this we only compare the most common database Uniparc, and Uniref [32] for sequences and RCSB [33] for protein structures. Image coming from `https://moalquraishi.wordpress.com/2019/04/01/the-future-of-protein-science-will-not-be-supervised/`

1.3.2 Diversity of Proteins Data

The current data landscape for protein modeling is characterized by a significant imbalance: the number of available protein sequences far exceeds that of protein structures stored in formats such as the Protein Data Bank (PDB). We account in 2024 for 250 million sequences on Uniprot [32] against 220 thousand structures on RSCB[33]. This disparity is growing, as highlighted in Fig. 1.8, which contrasts the rapid accumulation of sequence data against the more gradual increase in structural data. Although protein structures provide a more detailed understanding of protein functions, the scarcity of such data has led to the development of both sequence-based and structure-based models for protein design. This situation is exacerbated by the fact that only a small fraction of sequences have experimental annotations.

In response to these data availability trends, machine learning models for protein design have been categorized into three types: sequence-only models that utilize the extensive sequence databases; conditional sequence models that incorporate additional annotations to refine predictions; and structure-based models that focus on protein structures. This classification aids in comparing different methodologies within a probabilistic framework.

The prevailing gap between sequence and structural data has historically directed the field toward distinct modeling approaches. However, advancements in structure prediction algorithms, such as Alphafold 2 [34], are poised to bridge this divide. As these algorithms become more accurate, the distinction between the wealth of sequence data and the paucity of structural data is expected to diminish, facilitating a more unified approach to protein modeling. This development suggests a move towards a more integrated understanding of proteins, leveraging both sequence and structure data for enhanced protein design and research in computational biology, as in [35].

1.3.3 Sequence-based Models

Sequence-only models in protein design aim at understanding and generating the primary structure of proteins. Early iterations of these models were largely family-specific, relying on alignment-based methods to train on multiple sequence alignments (MSAs) of homologous sequences. Such models evolved from making simple position-specific predictions, like those found in Position-Specific Scoring Matrices (PSSMs) [36], to more sophisticated approaches that considered interactions between pairs of residues. This category has seen significant advancements with the introduction of generative models capable of simulating entire protein sequences, marking a pivotal development from their simpler predecessors. Among the notable models that have emerged are Hidden Markov Models (HMMs) [37], which are widely used for homology detection and family alignment, or Variational Autoencoders (VAEs) [38], which leverage deep learning to generate new sequences by learning a latent space representation of protein data.

This progress marks a departure from earlier, simpler prediction models. The introduction of family-agnostic protein language models, trained on unaligned sequences across various protein families, further extends the capabilities of sequence modeling. These models, leveraging autoregressive and masked-language modeling techniques from natural language processing, offer a comprehensive approach to protein sequence generation. Autoregressive models predict residues sequentially, based on preceding sequence information. It is a very effective strategy for generating de novo sequences [39, 40]. On the other side, masked-language models predict masked residues from all the others. This strategy is very efficient in extracting higher features and creates a meaningful latent representation of the sequence [41]. However, the predictive accuracy of family-agnostic models in fitness-prediction tasks sometimes does not match that of the more specialized family-specific models. This discrepancy has led to the development of hybrid models, such as the MSA Transformer [42], which combines the strengths of both approaches by utilizing MSAs and being trained across all protein families.

1.3.4 Direct Coupling Analysis: An important example of Sequence-Based Model

Direct Coupling Analysis (DCA) is a specific example within the sequence-only model category, characterized as a type of Potts model and it will be widely used in this thesis. Mainly DCA is a Boltzmann Machine cf 1.2.1 (therefore a special type of Energy-Based-Model) where all variables are visible. More precisely each variable represents the amino acid at a specific position.

DCA [43] is based on the principle of inferring maximum entropy models that are consistent with the observed second-order correlations in multiple sequence alignments (MSAs). The maximum entropy principle suggests constructing the simplest possible model that reproduces the observed data, in this case, the pairwise correlations between amino acid positions in protein sequences or nucleotide positions in

RNA sequences. The goal is to find a statistical model that maximizes entropy, subject to the constraint that the model reproduces the empirical correlations observed in the MSA.

The energy equation for DCA, is therefore very close to the one of Boltzmann Machine, with the specificity to have only visible variables. The energy of a sequence $\vec{\sigma}$, under this model, is given by:

$$E(\sigma) = -\sum_{i<j} J_{ij}(\sigma_i, \sigma_j) - \sum_i h_i(\sigma_i)$$

Where:

- $E(\sigma)$ is the energy of a sequence $\vec{\sigma}$, which determines the probability of observing that sequence under the model.
- σ_i and σ_j are the states (amino acids or nucleotides) at positions i and j, respectively.
- $J_{ij}(\sigma_i, \sigma_j)$ are the coupling between positions i and j, reflecting how the presence of specific amino acids or nucleotides at these positions co-evolve.
- $h_i(\sigma_i)$ represents the field or bias for a particular amino acid or nucleotide to occur at position i, capturing the effect of single positions independent of others.

This model aims to capture the most relevant statistical features of the MSA, specifically the first and second-order correlations, with the maximum entropy approach ensuring that no unjustified assumptions about the data are made beyond these correlations. By fitting the model parameters J_{ij} and h_i to the empirical data from an MSA, DCA identifies direct couplings between positions that are likely to be functionally or structurally important.

DCA was originally utilized for unsupervised contact prediction [43], identifying amino acids that directly interact within a protein's three-dimensional structure by analyzing co-evolutionary patterns in MSAs. This approach relies on the identification of co-evolved residues, suggesting functional or structural interactions. Beyond contact prediction, DCA also plays a crucial role in sampling de novo members of protein families [44], facilitating the generation of novel protein sequences. DCA's utility in protein design lies in its capacity to offer insights for the rational design of proteins with desired functionalities, ensuring structural integrity and stability. In this thesis, DCA will be frequently discussed as a principal example of sequence-only models, highlighting its importance and widespread application in the field of protein design.

Training a DCA model using the pseudolikelihood method [45] is an efficient alternative to directly maximizing the likelihood of the entire sequence alignment, which is computationally intensive due to the need to estimate the derivative of the partition function Z. The pseudolikelihood method replaces the likelihood of the whole sequence by the product of the conditional likelihoods of observing each

amino acid given the rest of the sequence. This approach significantly reduces computational complexity and allows for a direct computation of the gradient.

The pseudolikelihood of a sequence $\vec{\sigma}$ in a multiple sequence alignment (MSA) is defined as:

$$\mathcal{L}(\vec{\theta}) = \sum_{m=1}^{M} \sum_{i=1}^{L} \log P(\sigma_i^m | \vec{\sigma}_{\setminus i}^m; \vec{\theta})$$

Where:

- M is the number of sequences in the MSA.
- L is the length of the sequences.
- σ_i^m denotes the amino acid (or nucleotide) at position i in sequence m of the MSA.
- $\vec{\sigma}_{\setminus i}^m$ represents the sequence m with the amino acid at position i removed.
- $\vec{\theta}$ represents the parameters of the model, including both the coupling coefficients J_{ij} between positions and the field parameters h_i for individual positions.
- $P(\sigma_i^m | \vec{\sigma}_{\setminus i}^m; \vec{\theta})$ is the conditional probability of observing the amino acid at position i, given the rest of the sequence, under the model parameterized by $\vec{\theta}$.

This approach allows for efficient estimation of the coupling strengths and fields and is efficient for contact prediction, even though the resulting model has poor a poor generative power.

1.3.5 Extending Sequence-Based Model with labels

Conditional Sequence Models refine the generative process by incorporating conditions like taxonomic classifications or gene annotations, offering enhanced control over the generated sequences characteristics through various modeling techniques, like conditional VAE [46], Transformer based [47]. This needs a significant amount of data available for a specific property, these models learn the link between a functional label and an input sequence. This also highlights the need for semi-supervised approaches, able to benefit from the labeled and unlabeled data. Moreover, this approach only considers a categorical variable or a scalar value as a conditioning label. There remains a need for conditioning on more complex types of data such as complete sequences.

1.3.6 Structure-Based Models

In protein design, structure-based models play a crucial role by bridging the intricate relationship between a protein's linear sequence and its three-dimensional con-

formation. These models are divided into several distinct categories, each addressing different aspects of protein structure and design from unique computational perspectives.

The advancements in structure prediction models, especially highlighted by the breakthrough of AlphaFold [48], have significantly influenced computational biology. These models, by accurately forecasting protein tertiary structures from amino acid sequences, provide essential insights into protein functions and lay the groundwork for innovative protein design. This leap in predictive accuracy has spurred the development of new AI-driven approaches in structure modeling, rapidly expanding our capabilities in protein engineering. The progress made in structure prediction has not only enhanced our understanding of proteins but also accelerated the emergence of advanced models for protein design, marking a significant stride in the field.

Protein structure is a powerful tool for understanding protein function. Since structure dictates function, it's a logical approach to directly generate protein structures for design purposes. This is where structure generation models come in. These models, employing techniques like Generative Adversarial Networks (GANs)[49] and Variational Autoencoders (VAEs)[50], and diffusion models [51, 52], bypass the traditional sequence-based approach and attempt to directly create protein structures in 3D.

This focus on structure is crucial. By capturing the intricate relationships within the protein's spatial organization, these models can generate entirely new conformations. This ability to explore the vast universe of potential protein structures, many yet to be found in nature, opens doors for protein design.

Recent advancements have particularly highlighted the effectiveness of diffusion models in this context. Although diffusion models have achieved significant success in fields like image and language generation, their application to protein design has been challenging due to the intricate geometry of protein backbones and the complex sequence-structure relationships. However, the introduction of RoseTTAFold diffusion (RFdiffusion [51]) represents a major stride forward. By reusing the RoseTTAFold structure prediction network for protein structure denoising tasks, RFdiffusion has emerged as a generative model capable of remarkable achievements in various protein design challenges. It is trained by slowly adding noise to the structure and training it to recover the original structure as shown in 1.9.

Inverse folding models serve a critical function in protein design by identifying amino acid sequences predicted to fold into specified 3D structures [53, 54]. This methodology has several practical usages. It addresses practical needs within the protein design pipeline. For instance, in the RFdiffusion process, where only the protein backbone is generated, inverse folding models are essential for determining the corresponding amino acid sequence. Beyond this specific pipeline, these models have a proven utility in enhancing protein stability [28]. By starting with a natural protein's structure, inverse folding can propose alternative sequences that maintain

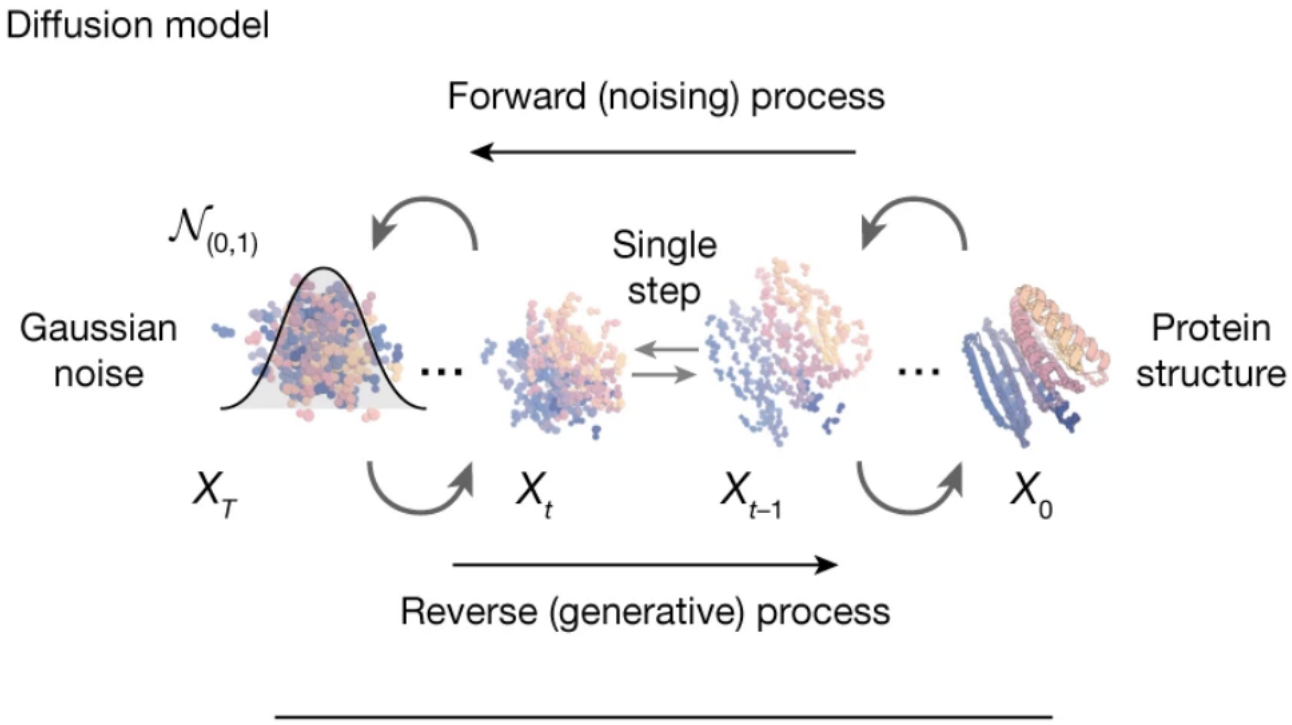


Figure 1.9: Denoising procedure of diffusion models for proteins. This illustration come from [51]

the original fold but with improved stability. Additionally, this thesis introduces a novel application of inverse folding models: utilizing them to explore the diversity of sequences that can adopt a given fold. This approach aims to map out the largest spectrum of potential candidates, varying in properties yet conserving the structural fold, thereby broadening the scope for discovering proteins with desired features and functionalities.

1.4 T-cell Receptor Epitope Binding Prediction

1.4.1 Introduction to T-cell Receptors (TCRs)

T-cell receptors (TCRs) are molecules found on the surface of T-cells, a type of white blood cell that plays a critical role in the immune system. TCRs are responsible for recognizing fragments of antigens as peptides bound to major histocompatibility complex (MHC) molecules on the surface of other cells. This recognition is fundamental to the adaptive immune response, enabling T-cells to identify and eliminate infected or cancerous cells [55]. The specificity of TCRs to their cognate antigen epitopes—short sequences of amino acids that are part of the antigen recognized by the immune system—is a cornerstone of the immune system's ability to detect a vast array of pathogens.

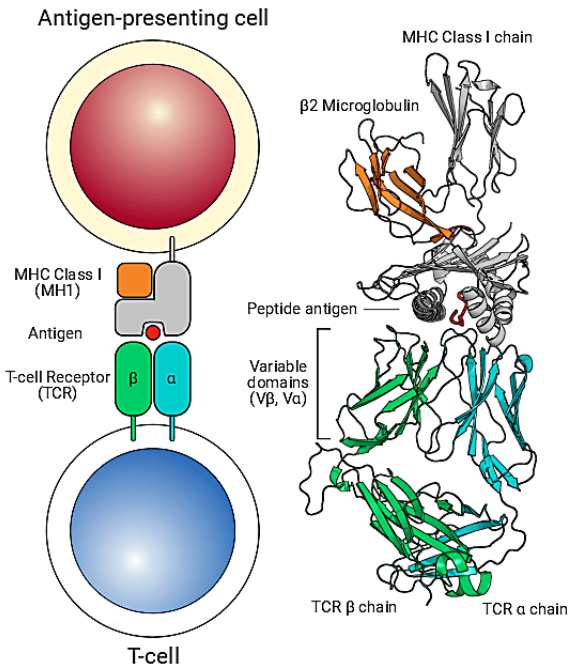


Figure 1.10: T cells rely on their surface receptors to recognize and bind with foreign and disease-associated antigens—in the form of peptide molecules—presented by innate immune cells, such as dendritic cells or other antigen-presenting cells (APCs). Figure taken from [56]

The interaction between a TCR and its epitope is specific, yet this specificity arises from a complex repertoire of TCR sequences generated through a random process of gene rearrangement. This diversity allows the immune system to respond to a wide variety of antigens but also presents a significant challenge in understanding and predicting TCR-epitope interactions.

The remarkable specificity of TCRs arises from their structure. TCRs are heterodimeric proteins, consisting of two separate polypeptide chains: α and β. Each chain is formed by the rearrangement of gene segments during T-cell development. The alpha chain utilizes variable (V) and joining (J) segments, while the beta chain incorporates an additional diversity (D) segment. This rearrangement process generates a vast library of unique TCR sequences, contributing to the immune system's diverse antigen recognition capabilities, as illustrated in 1.10

Crucially, the recognition of specific antigen epitopes relies on hypervariable regions called Complementarity Determining Regions (CDRs) present on both the alpha and beta chains. These CDRs form loops that directly interact with the antigen peptide bound to the MHC molecule as shown in Fig. 1.11. The amino acid composition within these CDRs determines the binding affinity of the TCR for a particular epitope. Ultimately, the precise fit between the CDRs and the epitope dictates T-cell activation and the elimination of target cells.

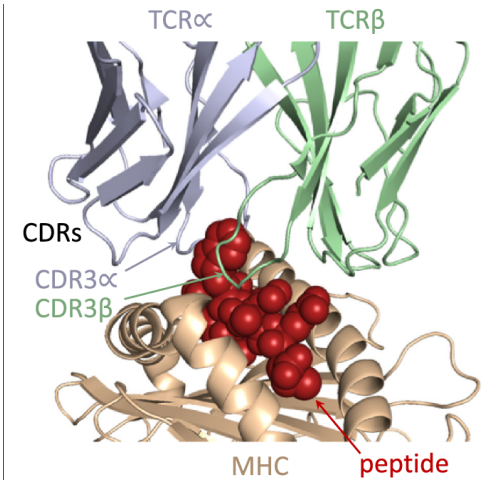


Figure 1.11: Closer view to the interaction zone between TCR and the pMHC. The CDR loops are the part involved in the binding. Figure taken from [57]

1.4.2 Challenges in TCR-Epitope Binding Prediction

Predicting the interaction between TCRs and epitopes is a daunting task due to several factors. Firstly, the immense diversity of TCR sequences, combined with the variability of epitopes, creates a vast interaction space. Each T-cell has a unique TCR, and the human body can hypothetically produce an almost infinite variety of TCRs ($> 10^{60}$ [58]), making it difficult to predict which TCR will interact with which epitope.

Secondly, the lack of comprehensive, high-quality data on TCR-epitope interactions limits the development of predictive models. The specificity of TCR-epitope binding implies that only a small fraction of all possible interactions are biologically relevant and thus observable. Most datasets are sparse, containing only a small subset of all potential interactions, and are often biased toward certain types of interactions or diseases.

Additionally, the binding affinity of a TCR for an epitope is influenced by many factors, including the structure of the TCR and the peptide-MHC complex, the flexibility of the peptide, and the chemical properties of the interacting surfaces. Modeling these interactions requires a detailed understanding of molecular biology that is difficult to capture with predictive models.

1.4.3 AI Approaches for TCR-Epitope Binding Prediction

The landscape of TCR-peptide interaction prediction has been enriched by various machine learning (ML) models, each bringing a unique approach to understanding the specificity of TCRs towards different peptides. Among these models, convolutional neural network (CNN) models like ImRex [59], TCRAI [60], and NetTCR 2.1 [61], have been prominent, utilizing deep learning techniques to analyze TCR sequences. Auto-encoder-based models such as DeepTCR [62] and decision-tree mod-

els like SETE [63] have also contributed significantly to the field, offering different perspectives on the prediction task. Gaussian process models, exemplified by TCRGP [64].

In addition to supervised ML approaches, unsupervised similarity-based methods such as TCRdist3 [65] have been developed. These methods leverage the inherent similarities between TCR sequences to infer potential interactions without the need for labeled training data.

Historically, the majority of TCR specificity prediction models relied on single chain data, predominantly CDR3β, due to its relative abundance. However, the advent of single-cell sequencing technologies has led to an increase in available paired data, comprising both the α- and β-chains. Recent works indicate that models trained on both chains outperform those trained on single-chain data [66].

While similarity-based methods can rival ML-based models under conditions of high training and evaluation data similarity and the presence of numerous positive TCR observations for a given peptide, ML methods generally surpass similarity-based approaches when these conditions are not met.

A notable challenge in the field is the limited extrapolative power of current models for predicting specificity towards previously unseen peptides, especially those not closely related to peptides in the training set. This challenge is predominantly due to the scarcity of training data, particularly for paired-chain data, and the imbalanced nature of the available data, which tends to favor a few peptides.

The pan-specific model approach, training on data covering a wide range of peptides including the peptide sequence as input, has shown success in the MHC system but has yet to achieve substantial performance in predicting binding for unseen peptides in the context of TCRs.

Chapter 2

Evaluating Generative Models of Protein Family

2.1 Introduction

Incorporating the need for a deeper understanding and evaluation of deep learning (DL) methods, especially in the context of their generative capabilities, is crucial. The rapid evolution of computational biology has been significantly propelled by the development of deep learning (DL) techniques, which hold the promise of unlocking new insights into biological phenomena like protein folding and interactions. As these methods become increasingly complex, the need for a critical evaluation of their generative capabilities and underlying mechanisms is paramount. This necessity forms the starting point of this work. We now have many deep learning methods available, which presents both an opportunity and a challenge. There is a need for a detailed framework to evaluate these methods, focusing not only on their effectiveness and efficiency but also on their interpretability.

The concept of second-order interactions has been seen as a good indicator of model quality. However, the question remains: do more profound, more complex models align with this perspective, or do they tap into higher-order interactions that significantly contribute to their generative capabilities?

2.2 Order Interactions estimation through distillation

This study employs model distillation to transfer knowledge from complex, deep neural network models to simpler Potts models. The objective is to critically assess the extent to which second-order interactions are responsible for the predictive capabilities of these sophisticated models. The Potts model, characterized by its emphasis on pairwise interactions, serves as an ideal candidate for evaluating the necessity and efficiency of complex model architectures in capturing the underlying biological phenomena.

The methodology for implementing model distillation consists of several key

steps. Initially, the neural network (teacher model) is trained on a dataset of protein sequences to learn a representation that captures both the explicit sequence characteristics and the implicit interaction patterns. Subsequently, the Potts model (student model) is trained not just on the raw sequence data but is also guided to mimic the output probabilities of the neural network, effectively capturing the learned interactions with a focus on second-order dependencies.

By quantifying the performance of the distilled Potts model against the original neural network, the study evaluates the contribution of second-order interactions to the overall predictive success of complex models.

2.3 Knowledge Distillation

The distillation process can be formalized as follows: Let $f_{\text{teacher}}(\mathbf{x};\Theta_{\text{teacher}})$ represent the output of the teacher model, where $\mathbf{x}$ is an input protein sequence and Θ_{teacher} denotes the parameters of the teacher model, a deep neural network trained to model protein sequences. The goal of the teacher model is to maximize the likelihood (or minimize the energy) of the given input sequence.

The student model, represented by $g_{\text{student}}(\mathbf{x};\Theta_{\text{student}})$, aims to replicate the behavior of the teacher model, where Θ_{student} are the parameters of the Potts model, significantly fewer in number compared to Θ_{teacher}. The distillation process involves training the student model to approximate the teacher model's output distribution, effectively learning $f_{\text{teacher}}(\mathbf{x};\Theta_{\text{teacher}})$ through the minimization of a divergence metric, typically the Kullback-Leibler divergence (KL divergence), between the output distributions of the teacher and student models:

$$\mathcal{L}_{\text{distill}} = D_{KL}\left(f_{\text{teacher}}(\mathbf{x};\Theta_{\text{teacher}}) \parallel g_{\text{student}}(\mathbf{x};\Theta_{\text{student}})\right)$$

In a standard distillation, the student model is exposed to both the original training data and the output probabilities generated by the teacher model. This dual input mechanism allows the student model not only to learn the direct mapping from sequence to structure/function but also to internalize the nuanced decision-making process of the teacher model.

In some works, the standard loss on the training set is added, such as a traditional cross-entropy loss against the true labels, or the negative loglikelihood (NLL) on the training data.

$$\mathcal{L}_{\text{total}} = \lambda\mathcal{L}_{\text{distill}} + (1-\lambda)\mathcal{L}_{\text{NLL}}(g_{\text{student}}(\mathbf{x};\Theta_{\text{student}}))$$

where λ is a hyperparameter that balances the contribution of the distillation loss and the standard training loss.

In the case of generative models, we can highly enrich the set of sequences used, by sampling new sequences with the teacher model. In such a situation, we can entirely focus on the distillation task ($\lambda = 1$).

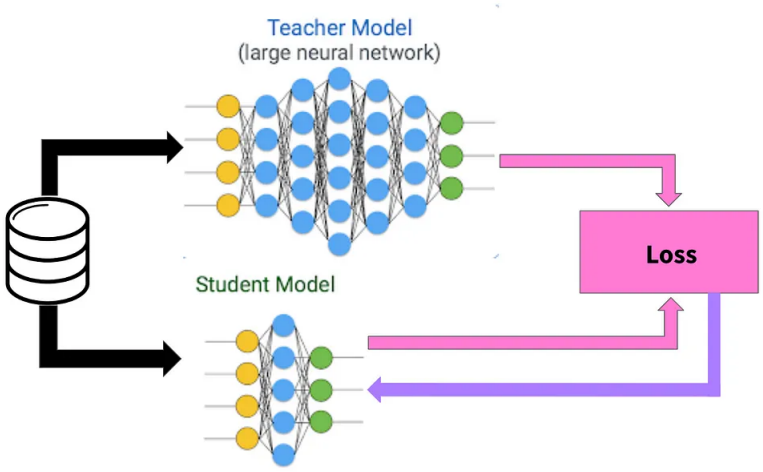


Figure 2.1: Representation of the distillation process: A first Deep neural network, called teacher is trained on a specific dataset. Then a smaller, simpler model, called student is trained to reproduce the same output than the teacher. In some settings, the student can also be trained on the original task of the Teacher, in addition to being trained to copy the teacher's output. Figure from `https://towardsdatascience.com/knowledge-distillation-simplified-dd4973dbc764`

2.4 Paper: Interpretable pairwise distillations for generative protein sequence models

In our study, we looked at neural network models designed for analyzing protein sequences, which are used for tasks like predicting mutations, designing proteins, and understanding protein structures. Although these complex neural networks are known for their effectiveness, we compared them to simpler models based on pairwise interactions.

We developed a method to simplify these complex models into ones that only consider pairwise interactions using an energy-based framework. Our results showed that these simplified models could almost match the performance of the complex neural networks in predicting the effects of mutations. This highlights the central importance of second-order interaction in protein modeling.

PLOS COMPUTATIONAL BIOLOGY

RESEARCH ARTICLE

Interpretable pairwise distillations for generative protein sequence models

Christoph Feinauer[1,2]*, **Barthelemy Meynard-Piganeau**[3,4], **Carlo Lucibello**[1,2]

1 Department of Computing Sciences, Bocconi University, Milan, Italy, **2** Bocconi Institute for Data Science and Analytics (BIDSA), Milan, Italy, **3** Laboratory of Computational and Quantitative Biology (LCQB) UMR 7238 CNRS, Sorbonne Université, Paris, France, **4** Department of Applied Science and Technologies (DISAT), Politecnico di Torino, Turin, Italy

* christoph.feinauer@unibocconi.it

Abstract

Many different types of generative models for protein sequences have been proposed in literature. Their uses include the prediction of mutational effects, protein design and the prediction of structural properties. Neural network (NN) architectures have shown great performances, commonly attributed to the capacity to extract non-trivial higher-order interactions from the data. In this work, we analyze two different NN models and assess how close they are to simple pairwise distributions, which have been used in the past for similar problems. We present an approach for extracting pairwise models from more complex ones using an energy-based modeling framework. We show that for the tested models the extracted pairwise models can replicate the energies of the original models and are also close in performance in tasks like mutational effect prediction. In addition, we show that even simpler, factorized models often come close in performance to the original models.

Author summary

Complex neural networks trained on large biological datasets have recently shown powerful capabilites in tasks like the prediction of protein structure, assessing the effect of mutations on the fitness of proteins and even designing completely novel proteins with desired characteristics. The enthralling prospect of leveraging these advances in fields like medicine and synthetic biology has created a large amount of interest in academic research and industry. The connected question of what biological insights these methods actually gain during training has, however, received less attention. In this work, we systematically investigate in how far neural networks capture information that could not be captured by simpler models. To this end, we develop a method to train simpler models to imitate more complex models, and compare their performance to the original neural network models. Surprisingly, we find that the simpler models thus trained often perform on par with the neural networks, while having a considerably easier structure. This highlights the importance of finding ways to interpret the predictions of neural networks in these fields, which could inform the creation of better models, improve methods for their assessment and ultimately also increase our understanding of the underlying biology.

Citation: Feinauer C, Meynard-Piganeau B, Lucibello C (2022) Interpretable pairwise distillations for generative protein sequence models. PLoS Comput Biol 18(6): e1010219. https://doi.org/10.1371/journal.pcbi.1010219

Editor: Joanna Slusky, University of Kansas, UNITED STATES

Received: October 13, 2021

Accepted: May 17, 2022

Published: June 23, 2022

Data Availability Statement: Data and code can be found at https://github.com/christophfeinauer/PairwiseDistillations.

Funding: The author(s) received no specific funding for this work.

Competing interests: The authors have declared that no competing interests exist.

1 Introduction

Many different types of generative models for protein sequences have been explored, from pairwise models inspired by statistical physics [1–4] to more complex architectures based on neural networks like variational autoencoders [5–7], generative adversarial networks [8], autoregressive architectures [9, 10] and models based on self-attention [11]. While such models promise a rich field of applications in biology and medicine [12], the question of what information they extract from the sequence data has received less attention. This is, however, a very interesting field of research since especially the more complex models might extract non-trivial higher-order dependencies between residues. This in turn might reveal interesting biological insights.

Some recent works address this interpretability issue. In Ref. [13], the authors introduce the notion of *pairwise saliency* and use it to quantify the degree to which more complex models learn structural information and how this relates to the performance in the prediction of mutational effects. Ref. [14] instead constructs pairwise approximations to classifiers trained on categorical data and, among other results, show an example using protein sequence data.

We observe that the performance of many different models on tasks like the prediction of mutational effects is often similar even when using very different architectures and, in addition, is close to what simple, pairwise models achieve (see e.g. [9]). It appears natural to ask then how much of the predictive performance of the more complex models like variational autoencoders is due to higher-order interactions which are inaccessible to more simple models.

We therefore ask in this work how close trained neural network (NN) based models are to the manifold of pairwise distributions. To this end, we train two different architectures on protein sequence data. Interpreting these models as energy-based models [15], we present a simple way to extract pairwise models from them and analyze errors in energy between extracted and original models. We show that the subtle question of gauge invariance is important for this purpose and address this invariance ambiguity using different objective functions for the extraction. In addition, we show that even simpler models for which the probability distribution factorizes over positions in the protein can often come close to the performance of the original models after extraction.

Our work embeds itself into the field of *knowledge distillation* [16], where the goal is to extract simpler models from more complex models, improving the computational performance of the models and also increasing interpretability [17]. We also note, however, that higher-order interactions are not necessarily connected to a lack of interpretability: Restricted-Boltzmann Machines, for example, provide a way to model higher-order interactions in protein sequences that are still relatable to biological properties [18].

The main contributions of this paper are 1) we introduce a method for extracting independent and pairwise models for arbitrarily complex generative models for protein sequences, 2) we connect this task to the properties of gauge transformations and show that one can focus the extraction on different parts of the sequence space, 3) we show that pairwise models (and sometimes even independent models) are often good approximations for the original models in terms of the reconstruction of energies and also for tasks like mutational effect prediction.

The code and data for reproducing the experiments in this work are available at https://github.com/christophfeinauer/PairwiseDistillations.

2 Methods and data

While we present a more detailed description in the following sections, we describe in this section the general pipeline we use throughout the paper. Our goal is to analyse generative models for protein sequences trained on a multiple sequence alignment of homologous protein sequences. Denoting by N the length of the aligned sequences, these models define a

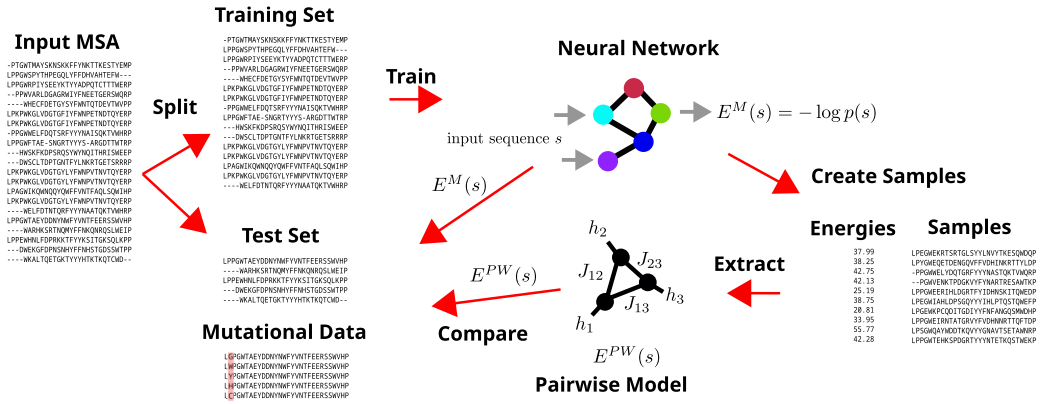


Fig 1. Pipeline. An overview of our train and test pipeline. Given an input MSA, we remove duplicated sequences and split into training and test set with a ratio of 9:1. We then train neural network models on the training set. After creating samples (either from the distribution induced by the neural network itself or from another distribution), we calculate the energy (negative log probability) of these samples from the neural network. We then use the samples and energies to extract a pairwise model (and also an independent model) and compare the original neural network model with the extracted in terms of their energies and their performance in mutational effect prediction.

https://doi.org/10.1371/journal.pcbi.1010219.g001

probability distribution $p(s)$, where s is an arbitrary sequence of amino acids of fixed length. One goal of generative modelling is to arrive at a $p(s)$ which reflects faithfully the evolutionary constraints acting on the family, assigning high probability to sequences with high fitness and low probability to sequences with low fitness. A common benchmark for such models is the prediction of experimentally measured fitness values of mutated sequences. A successful model can then for example be applied when screening pathological mutations involved in human disease [2, 19].

The arguably simplest generative model is an *independent* model, where the probability factorizes over the positions in the protein, which is equivalent to saying that the log probability is a sum over independent terms including only a single position. Slightly more complex models are *pairwise* models, where the log probability is a function including terms depending on up to two positions. A central claim in recent literature is that models based on neural networks outperform simpler models because they can capture more complex constraints from the data, which would correspond to higher-order interactions (terms including more than two positions) in the log probability.

In order to test and quantify this claim, we employ a simple pipeline (see Fig 1): We train neural network based generative models and test how well we can reproduce their distribution with independent and pairwise models. We do this by sampling sequences from either the uniform distribution or from the distribution induced by the original generative model itself, calculating the energies (negative log probabilities) of these sequences in the original generative models and then training a pairwise or independent model to reproduce these energies on the same sequences using a simple mean squared error loss. We then analyse how well these simpler models reproduce the probabilities of sequences on a test set and how well they perform in the task of the prediction of mutational effects when compared to the original models.

We provide a list of abbreviations for the various distributions and models we use in S1 Appendix.

2.1 Data and preprocessing

The training data for all models are aligned, homologous sequences of protein domains gathered in a multiple sequence alignment (MSA), where every row corresponds to a sequence of amino acids and every column to a position with homologous residues [20].

We denote a single amino acid sequence of length N as $s = (s_1, \ldots, s_N)$, where we identify every possible amino acid with a number between 1 and q, with q being the number of possible symbols (we use 20 amino acids and 1 alignment gap symbol, so $q = 21$). While we use this representation for the mathematical description and analysis, the implementations of the models use a one-hot encoding of the amino acids as inputs, where every amino acid is replaced by a vector of size q which contains zeros except at the position indicated by the integer corresponding to the amino acid. The input size for the models is thus a vector of size Nq.

The datasets we use are taken from [6]. Since some steps in our pipeline are computationally heavy, we selected 5 of the smaller of the 41 datasets used there. The datasets show a range of different properties with respect to the performances of different types of models (see Section 3.1). Every dataset consists of experimental fitness values for mutations in a target sequence and an MSA containing homologous sequences of the same target sequence. We use the datasets for the *BRCA1* tumor suppressor gene [21], the *GAL4* transcription factor [22], the small ubiquitin-like modifier *SUMO1* [23], the ubiquitination factor *UBE4B* [24] and the yes-associated protein *YAP1* [25]. When the dataset reported more than on experimental measurement we chose the same one as used in [9], see Table A2 in S1 Appendix for a list of experimental measurements used and the number of mutants in each dataset.

As the first step of preprocessing of the input MSAs, we replaced non-standard amino acids with a gap and removed all duplicated sequences from the datasets. We then partitioned the remaining sequences randomly into train and test sets, comprising 90% and 10% of the sequences respectively (see Table A1 in S1 Appendix for the number of sequences in the train and test sets).

While there are no identical sequences in the train and test sets due to the prior removal of duplicated sequences, for most families there is a fraction of the test sequences that have a Hamming distance of 1 to some sequence in the training set, see Fig 2. We note, however, that this should not pose a problem in the evaluation phase for the extracted models, since we do not extract models on the training sequences, but either from samples generated by the original models or from uniformly sampled sequences. In order to still control for possible effects on how well the extracted models reproduce the distribution of sequences in the test set with

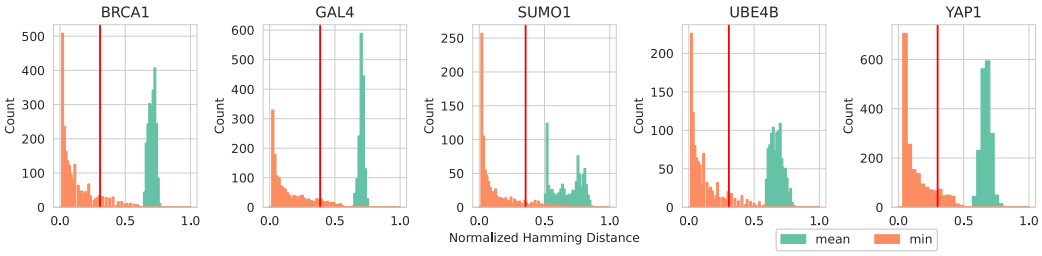


Fig 2. Normalized Hamming distance from test to train datasets. Shown are mean (green) and minimum (orange) distances from every test sequence to the sequences in the training set. The normalized Hamming Distance is the number of differing amino acids between two sequences, divided by the sequence length. The red bar indicates the distance at which the test set is divided in a ratio 9:1 with respect to the minimum distance to the training set.

https://doi.org/10.1371/journal.pcbi.1010219.g002

respect to the distance from the training set, we partition the test set further into two sub-datasets: One which contains the 10% of test sequences that are farthest from the training set in terms of the minimum Hamming distance and another one with the other 90% of sequences. In Fig 2 this cutoff is indicated by a red bar.

2.2 Original models

We train two different types of models on the datasets described in the last section: The autoregressive architecture presented in [9] (ArDCA), and a variational autoencoder (VAE) using the architecture from [5]. We refer to these models as the *original models* in order to distinguish them from the extracted models that are introduced later. For a given family, all models (including EVMutation, which we discuss in Sec.3.1) are trained on the same training MSA. All models apply a reweighting scheme, which aims at removing phylogenetic bias by giving a smaller weight in the objective function to sequences that have many similar sequences in the training set. We refer to the original publications for details on these.

ArDCA uses an autoregressive decomposition of the probability,

$$p(s) = \prod_{i=1}^{N} p(s_i|s_{<i}), \tag{1}$$

where

$$p(s_i|s_{<i}) = \frac{\exp(h_i(s_i) + \sum_{j=1}^{i-1} J_{ij}(s_i, s_j))}{z_i(s_{<i})}. \tag{2}$$

Here, the h are parameters depending only on a single position and the J are parameters depending on pairs of positions. The $z_i(s_{<i})$ is a normalization depending on the part of the sequence preceding position i, which we denote by $s_{<i}$. While the form is reminiscent of a pairwise model (see Sec. 2.3), the log probability $\log p(s)$ cannot be written as a sum of terms including only up to two positions, and therefore the model can in principle also include higher-order interactions. The model can be interpreted as a repeated application of the *softmax* operation common in classification using neural networks, predicting the next amino acid in the sequence given the previous ones.

Due to the autoregressive architecture, the probability $p(s)$ and consequently the likelihood function can be calculated directly using Eq 2, where both the nominator and denominator are tractable. The training is done using the L-BFGS method [26], with an additional $L2$ regularization, see [9] for further details. We use the code provided by the authors for training.

The VAE we use follows the architecture in [5]. The decoder defines a probability distribution $q_\psi(z|s)$ over the latent representation z given the one-hot encoded input sequence s by using a multivariate Gaussian distribution with means $\mu = W^\mu h^{enc} + b^\mu$, log variances $\log \sigma^2 = W^\sigma h^{enc} + b^\sigma$, and zero off-diagonal correlations. These are expressed in terms of the output of single layer: $h^{enc} = \tanh(W^{enc} s + b^{enc})$. The weight matrices W^{enc}, W^μ and W^σ together with the biases b^{enc}, b^μ, b^σ form ψ. The decoder $p_\Theta(s|z)$ defines a factorized probability distribution over s given z given by $p_\Theta(s|z) = softmax(W^s h^{dec} + b^s)$, with $h^{dec} = \tanh(W^{dec} z + b^{dec})$. Here the weight matrices W^{dec}, W^s and the bias vectors b^{dec}, b^s form the set of parameters Θ. The prior over z is a standard Gaussian.

The VAE can be trained using the ELBO objective function, see [5] for details. We use the code of the authors of [5] to train the VAE models, which uses full-batch gradient descent on the ELBO objective and a weight decay regularization.

Since neither mutational effect prediction nor contact prediction is the central aim in [5], we test the VAE for a range of hyperparameters on all five datasets and assess the resulting

performance on the task of mutational effect prediction, see Section 3.1 for details. The hyperparameters we test are the number of units in the hidden layers, the dimension of the latent representation and the weight decay setting. We then select a subset of these models for the application of the rest of the pipeline in Fig 1.

An assessment of the performance of the original models is done in Sec. 3.1.

2.3 Energy-based models

Using a framework based on Energy-based models (EBMs) [15], we can define a probability distribution $p(s)$ over protein sequences by specifying the energy $E_\theta(s)$, which is equal to the negative log probability up to a constant. The energy can be implemented for example by a neural network with weights and biases represented by θ. While the calculation of the exact probability

$$p(s) = \frac{e^{-E_\theta(s)}}{Z_\theta} \quad \text{with} \quad Z_\theta = \sum_{s'} e^{-E_\theta(s')} \tag{3}$$

is intractable since the normalization constant Z_θ is a sum over q^N terms, numerous ways of training such models have been developed.

In this work, we use the fact that *any* probability distribution $p(s)$ can be thought of as an EBM by defining $E(s) \coloneqq -\log p(s)$. We will use the term *energy* for both cases: when derived from a distribution $p(s)$, and when given by an explicit energy function. While this formulation could be extended to models for sequences of varying length, we restrict ourselves in this work to sequences of fixed length.

2.4 Energy expansions and gauge freedom

We call $I = \{1, \ldots, N\}$ the set of all positions in the sequence s and s_L the subsequence consisting of amino acids at positions in $L \subseteq I$. Then, we can expand any energy $E(s)$ in the form

$$E(s) = \sum_{L \subseteq I} f_L(s_L), \tag{4}$$

where f_L is a function depending only on the amino acids at positions at L, and the sum is over all subsets of I. We will use f for denoting the set of all f_L in the expansion. Models for which $f_L = 0$ for $|L| > 2$ are called *pairwise models* (or *Potts models*) and their energy can be written as a special case of Eq 4 as

$$E^{pw}(s) = -\sum_{i=1}^{N} \sum_{j=i+1}^{N} J_{ij}(s_i, s_j) - \sum_{i=1}^{N} h_i(s_i) - C, \tag{5}$$

with J being commonly called couplings and h the fields [27]. Note that while the terminology is the same as for the parameters in ArDCA (see previous Sec.2.2), the parameters of ArDCA cannot be easily mapped to the parameters of a pairwise model. The constant C is typically not added to the model definition since it does not change the corresponding probabilities, but we keep it in order to be consistent with the generic expansion in Eq 4.

Models for which $f_L = 0$ for $|L| > 1$ are called *independent* or *profile* models [9]. Their energy function can be written as

$$E^{ind}(s) = -\sum_{i=1}^{N} h_i(s_i) - C, \tag{6}$$

which results in a factorized version of the probability distribution in Eq 3.

The expansion in Eq 4 is not unique, which means that given an energy $E(s)$ it is possible to find different expansion parameters f for which Eq 4 holds. Therefore additional constraints must be imposed to fix the expansion coefficients (gauge fixing). It is for example trivial to rewrite the *pairwise* model in Eq 5 as a model with interactions only of order N by defining $f_I(s) = E(s)$ and $f_L = 0$ for $|L|<N$.

A common route is to impose the so-called *zero-sum* gauge [28], also called the *Ising* gauge, which aims to shift as much of the coefficient mass to lower orders as possible (see, e.g., Ref [14], where the authors use the term 'Ising gauge' and Section D.2 in S1 Appendix for details). This is intuitively sensible, since explaining as much of the variance as possible with low order coefficients seems to be a key element when trying to understand how complex the model is. However, we will show in the next section that the problem of gauge invariance is more subtle and important for understanding the structure of the fitness landscape induced by NN models.

2.5 MSE formulation for extraction

We formulate the problem of extracting pairwise and independent models from more general models by using a loss function $\mathcal{L}$ that measures the average mean squared error (MSE) in energies with respect to a distribution D over sequences. We call $E^M(s)$ the energy of the original model that we want to project onto the space of pairwise or independent models. We define the loss function over the parameters J, h and C on which the pairwise energy $E^{pw}(s)$ of Eq 5 implicitly depends as

$$\mathcal{L}(J,h,C) = \mathbb{E}_{s\sim D}\left[\left(E^M(s) - E^{pw}(s)\right)^2\right]. \tag{7}$$

We minimize the loss function with respect to J, h and C and use the resulting pairwise model E^{pw} as an approximation to E^M. For the independent model $E^{ind}(s)$ as defined in Eq 6, we use the same loss function but fix J to 0. Since independent models can be seen as special cases of pairwise models, we restrict the following discussion to pairwise models.

The distribution D is central in this formulation of the problem and is closely related to the question of gauge invariance. It can be shown that if D is the uniform distribution U over sequences, the minimizer of $\mathcal{L}(J,h,C)$ is equivalent to the pairwise part of E^M in the zero-sum gauge (see Section D in S1 Appendix for a proof). This means in reverse, that extracting the pairwise model using the zero-sum gauge is equivalent to minimizing the MSE in energy when giving all possible sequences equal weight. However, generative models trained on protein families are used only on a small region of the sequence space. By changing D it is possible to give more weight to these regions and construct a pairwise model that might be worse in replicating E^M globally, but better in regions of interest. This is equivalent to extracting the pairwise interactions in a different gauge of E^M.

A natural candidate for D is the distribution induced by E^M, leading to extracted models that aim to reproduce the original distribution well on typical sequences of that distribution. We denote this distribution by M. With this choice, the loss corresponds to an f-divergence ($f(t) = log^2(t)$) in the unnormalized distribution space [29]. Notice also that for a trained model E^M, one would expect this distribution to be close to the distribution of sequences in the training data.

One possible interpretation is that using different distributions D, one obtains extracted models corresponding to different 'views' of the original model: Using the uniform distribution U, we obtain an expansion that explains the original model distribution over the whole sequence space with minimal higher-order interactions. The extracted pairwise model then corresponds to the pairwise part in this expansion. Using the model distribution M itself, on

the other hand, we obtain an expansion where the pairwise part explains as much of the energy variation on typical sequences from the model distribution itself, and the extracted pairwise model corresponds to this pairwise part. We show below that using the model distribution M has advantages when using the extracted pairwise model for reproducing the energies on the test sequences and also in many cases for mutational effect prediction.

Note that if the original model is in fact a pairwise model, then for any D with a sufficiently large support the minimizer of Eq 7 should correspond to the original model (up to a gauge transformation).

The method we use for minimizing the loss in Eq 7 depends on the distribution used for creating the samples. For the uniform distribution U, the exact minimizer of the loss can be calculated by taking averages over the energies of the original model $E^M(s)$ with s sampled from a uniform distribution, (see Section D.2 in S1 Appendix for a derivation). This leads to the sampling estimators

$$\begin{aligned} C &= -\mathbb{E}_{s\sim U}[E^M(s)] \\ h_i(a) &= -\mathbb{E}_{s\sim U}[E^M(s)|s_i = a] + C \\ J_{ij}(a,b) &= -\mathbb{E}_{s\sim U}[E^M(s)|s_i = a, s_j = b] + h_i(a) + h_j(b) + C, \end{aligned}$$

where $\mathbb{E}_{s\sim U}[E^M(s)]$ is the expectation of $E^M(s)$ when sampling s from the uniform distribution, $\mathbb{E}_{s\sim U}[E^M(s)|s_i = a]$ is the expectation of $E^M(s)$ when keeping s_i fixed to a and sampling the amino acids at the other positions from the uniform distribution and $\mathbb{E}_{s\sim U}[E^M(s)|s_i = a, s_j = b]$ is the expectation of $E^M(s)$ when fixing s_i to a, s_j to b and sampling the amino acids at the other positions from the uniform distribution. For independent models, only the first two equations are used.

For distributions D different from U, there is in general no simple sampling estimator for the parameters of the pairwise model E^{pw}. Therefore, in the case where we set $D = M$, the distribution induced by $E^M(s)$, we resort to gradient descent on the loss in Eq 7. After preparing 10^7 samples from the M distribution, we minimize the loss using the Adam optimizer [30] with a batch size of 10000. The samples in a batch are sampled individually from all available samples for every gradient descent step. Since not all amino acids might be observed in all positions in the samples from the model distribution M, we replace every sample in the batch individually with a sample from the U distribution with a probability of 1%. All parameters are initialized to 0. We keep an exponential moving average of the batch losses with a smoothing factor of 0.9 and stop optimizing if this average has not reached a new minimum within 1000 gradient descent steps. We chose 10^7 samples based on the observation that minimizing the loss in Eq 7 can be formulated as solving a set of linear equations in the parameters of the extracted model (see Eq. 4 in S1 Appendix). While we cannot solve this set of equations directly for arbitrary distributions D, we aim to use a number of samples at least as large as the number of parameters we want to fit. The largest sequence length we have in our datasets is 77, which corresponds to about 1.3×10^6 parameters. For independent models, we use the same method but discard the gradient of J.

2.6 Sampling from the original models

For both original models we create samples from the uniform distribution by sampling every amino acid (including the gap) at every position with an equal probability $1/q$. For ArDCA, we obtain samples from the model distribution M by sampling amino acids sequentially using the expression in Eq 2. For the VAE models, we sample the latent factors z from a standard normal

distribution and sample amino acids at every position given the probabilities as returned by the decoder.

For calculating the energies for sequences in ArDCA, we can again use the autoregressive decomposition in Eq 2. For the VAE models, we use importance sampling [31] with 5000 samples.

3 Results

3.1 Performance of original models

We train ArDCA and VAE models on the five different datasets described in Section 2.1. We assess their performance in terms of mutational effect prediction using the experimental values provided in the datasets, see Appendix Table A2 in S1 Appendix for a summary of experimental measurements used. These experimental values consist of quantitative fitness measures of sequences that contain mutations with respect to a wild-type sequence. Since these measures are different for the different datasets, we use the Spearman rank correlation between the energies of the mutants in the original models and the experimental values as an indicator of performance.

We also compare the performance of EVMutation [3], which is a method based on pairwise models, i.e., training an energy function as defined in Eq 5 directly on the training data. We retrain EVMutation using the code provided by the authors, where the training is based on the method of pseudolikelihoods (see [3] and [32] for details).

For ArDCA, we use the code and hyperparameters optimized for generative modeling provided by the authors, using an $L2$ regularization of strength 0.01 for J and 0.0001 for h (values communicated by the authors of [9]). The performance in terms of the Spearman correlation is close to the ones reported in [9]: We report the Spearman correlation for the original ArDCA models in Fig 3 (the green bars). On four out of five datasets, ArDCA outperforms EVMutation and is roughly equal in performance on one dataset (GAL4).

For the VAE, we do not have hyperparmeters optimized for generative modeling. We therefore keep the general architecture of a single layer of hidden units in the decoder and encoder (same number of hidden units in encoder and decoder) and train for a series of different numbers of hidden units (40, 80, 100, 120, 140 and 160), different latent dimensions (5, 10, 20, 40, 80 and 120) and different settings for the weight decay strength (0.1, 0.05, 0.01, 0.005 and 0.001), using the code provided by the authors of [5]. We then assess the resulting 180 models

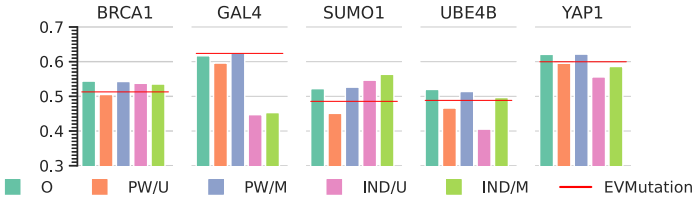


Fig 3. Spearman correlation with experimental data of original (O) and extracted models (PW/U, PW/M, IND/U, IND/M) for ArDCA. Shown is the Spearman rank correlation between the experimental data and the energies of the original model (O), the pairwise and independent models extracted using samples from a uniform distribution (PW/U and IND/U) and for the pairwise and independent models extracted using samples from the original model distribution (PW/M and IND/M).

https://doi.org/10.1371/journal.pcbi.1010219.g003

for every dataset (so 900 models in total) in terms of their mutational effect prediction performance.

In Fig B1 in S1 Appendix we display the Spearman correlation values for all 900 models. From these results, it appears that the number of hidden units and the size of the latent dimension have minor effects on the performance. The weight decay, on the other hand has a more pronounced effect for the range we tested. Interestingly, this effect has opposite directions for different datasets: For example, a lower weight decay strength leads to generally better results for GAL4, while the best results for SUMO1 are obtained when using the strongest weight decay. This seems to correlate with the performance of independent models as reported in [9] (Fig 3 there). There the authors report that for example for SUMO1, an independent model trained directly on the training data performs *better* than EVMutation, which is based on a pairwise model. For GAL4 (called GAL1 in [9]), on the other hand, an independent model performs significantly worse than EVMutation. It is tempting to speculate that the stronger weight decay suppresses pairwise and higher-order interactions in the VAE, which in the case of SUMO1 seems to improve the performance and in the case of GAL4 decreases the performance. This would indicate that for SUMO1, some of the patterns the models capture in the data are not in agreement with the experimentally measured fitness values. We corroborate this finding in later sections.

We note that for all datasets, we can find a VAE model that outperforms EVMutation in terms of mutational effect prediction. However, since we use the same experimental measurements to assess the performance of extracted models in later sections, choosing the best model based on these results might introduce biases. Since the number of hidden units and the size of the latent dimension seem to be less important, we fix these to 40 and 5 respectively, and run our pipeline for all five weight decay values. Within this subset of the trained models, there is always a model that performs as well or better than EVMutation (see green bars in Fig 4), except for UBE4B. On the same time, the performance is often very poor for specific weight decay settings: Setting for example strong weight decay of 0.1 for GAL4 reduces the Spearman correlation with the experimental data to well below 0.5, while EVMutation gives a correlation of more than 0.6.

3.2 Energy errors

In Fig 5 we show the error in the energies of extracted pairwise and independent models with respect to the energies in the original models for ArDCA. As described in Section 2, we use two different distributions D in Eq (7) for sampling the sequences used for the extraction of the pairwise models: The uniform distribution (U) and the distribution of the original generative model (M). We evaluate the error on the 10% of test sequences that are farthest from the training sequences in terms of the minimum Hamming distance (called 'Test Distant' in Fig 5), on the 90% remaining sequences (called 'Test Close' in Fig 5) and on the sequences from the mutational datasets.

The error in the plot is the root mean squared error, normalized by the range, i.e.,

$$\text{Normalized RMSE} = \frac{\sqrt{\frac{1}{M}\sum_{m=1}^{M}\left(E^M(s_m) - E^{pw}(s_m)\right)^2}}{\max_m E^M(s_m) - \min_m E^M(s_m)}, \tag{8}$$

where $\{s_m\}_{m=1}^M$ is the set of sequences on which we calculate the error, E^M is the energy of the original model, E^{pw} the energy of the extracted pairwise model and $\max_m E^M(s_m)$ and $\min_m E^M(s_m)$ are the maximum and minimum energies of the original model on the dataset.

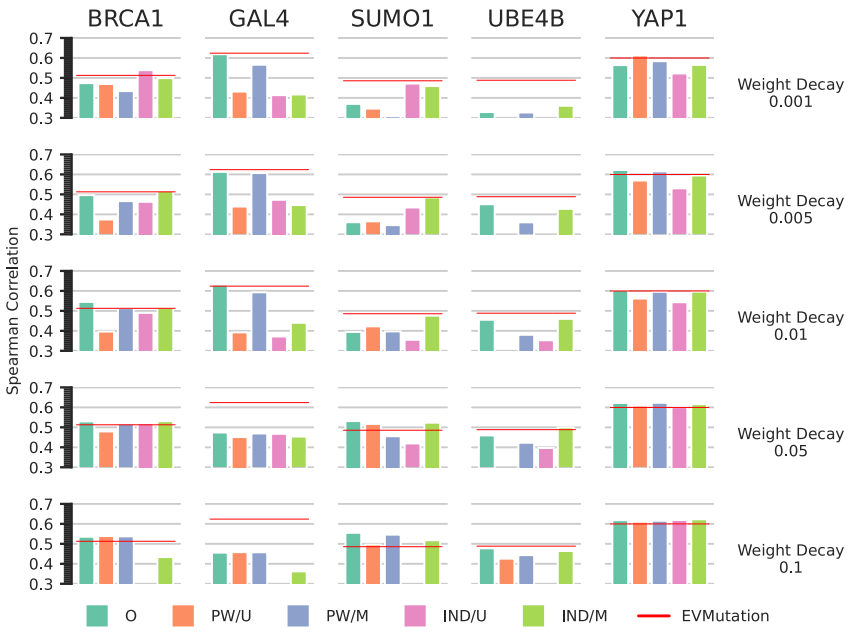


Fig 4. Spearman correlation with experimental data of original (O) and extracted models (PW/U, PW/M, IND/U, IND/M) for VAE models. Shown is the Spearman rank correlation between the experimental data and the energies of the original model (O), the energies of the pairwise and independent models extracted using samples from a uniform distribution (PW/U and IND/U), and the energies of the pairwise and independent models extracted using sequences sampled from the original model distribution (PW/M and IND/M). The rows correspond to different weight decay settings in the original model, as indicated on the right.

https://doi.org/10.1371/journal.pcbi.1010219.g004

The error for pairwise models is considerably smaller than for independent models, which is evidence that the original ArDCA models are indeed including at least pairwise interactions. In most cases, the error drops significantly when using the model distribution M for extraction instead of the uniform distribution U. This can be taken as evidence that the original models indeed include higher-order interactions and corroborates the idea that focusing on a specific part of the sequence space improves the quality of the extracted model on there. On the same time, the errors as percentages are relatively low for the extracted pairwise models: The errors are between 1% and 10% when using uniform samples and around 1% or below when using samples from M for extraction. This indicates that the ArDCA models are relatively close to pairwise models in the space around the training and test sequences.

Interestingly, the error on the test sequences distant from the training set is similar or smaller than the error on the test sequences closer to the training set. The largest error is on the sequences from the mutational dataset, which are very close to the training set. One possible explanation for this is found in Fig 6, where we plot for the test sequences of the BRCA1 dataset i) the energies of the test sequences in the original model; ii) their standard deviation at a given Hamming distance; iii) the absolute error on single sequence when comparing the energies from a model extracted with sequences sampled from the original model distribution;

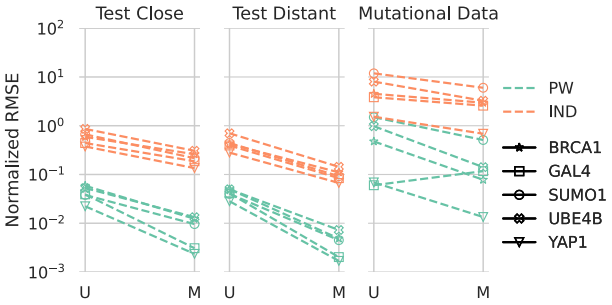


Fig 5. Errors in energies of the extracted pairwise and independent models with respect to the original ArDCA models. The three columns correspond to three different datasets: "Test Distant" corresponds to the 10% of sequences in the test set that have the largest distance to the training set in terms of minimum Hamming distance, "Test Close" to the remaining sequences. The right-most column corresponds to the sequences in the mutational datasets. The colors indicate whether the extracted model is an independent model (orange) or a pairwise model (green). Within every column, the left bar (U) corresponds to models extracted with samples from the uniform distribution, the right bar (M) to models extracted with samples from the distribution of the original models. The error shown is the normalized root-mean squared error (see Eq 8. Note the logarithmic scale.

https://doi.org/10.1371/journal.pcbi.1010219.g005

iv) the root-mean squared error, all in dependence of the normalized Hamming distance to the closest sequence in the training set. As can be seen in the upper right panel, there is an *inverse* relation between the standard deviation of energies in the original ArDCA model and the distance from the training set. The root-mean-squared error of the energies (the error defined in 8 without the denominator), shown in the lower right panel, closely follows this relation, meaning that the root-mean-squared error decreases on test sequences farther from the training set. We therefore speculate that the original model is more discriminative on sequences close to the training set, making it harder for the extracted pairwise model to reproduce the energy fluctuations there.

In Fig B3 in S1 Appendix we also show a scatter plot of the test sequences and the sequences of the mutational training set.

We show the errors for the VAE models with different weight decay settings in Fig 7. The general trend is very similar to the one described for ArDCA above, albeit with larger errors, indicating that the VAE models are less well described with pairwise models than ArDCA models. However, using sequences from the model distribution for extraction the error for pairwise models is mostly well below 10% on test sequences and, depending on the weight decay setting, often close to 1%. This indicates that when focusing on the part of the sequence space close to training and test sequences, the models can still be well approximated with pairwise models.

The performance of extracted pairwise and independent model seems to be closer together, which can be taken as evidence that the VAE models rely less on pairwise interactions. Also, the difference between extracted pairwise and independent models seems to increase when switching from using uniformly sampled sequences for extraction to sequences sampled from the original model distribution.

3.3 Comparing extracted couplings to EVMutation

Given that EVMutation is based on a pairwise model, we can directly compare the couplings from the extracted models to the ones obtained from EVMutation. In Fig 8 we plot couplings

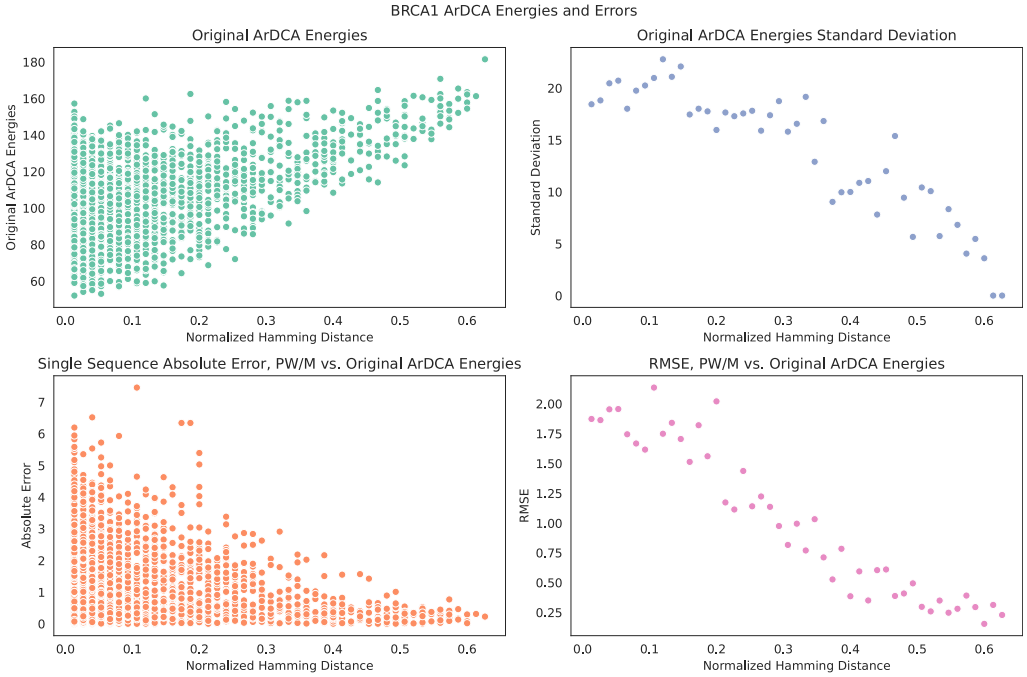


Fig 6. Energies and their standard deviations on BRCA1 test sequences. All plots share the abscissa, which shows the normalized Hamming distance of the test sequences to the closest sequence in the training set. *Upper left:* Energies of individual test sequences in the original ArDCA model. *Lower left:* Absolute errors on individual test sequences of a pairwise model extracted with sequences sampled from the original model distribution M with respect to the original energies. *Upper right:* Standard deviation of the energies in the original model at a given distance. *Lower right:* Root-mean-squared error on all test sequences at a given distance for a pairwise model extracted with sequences sampled from the original model distribution M with respect to the original energies.

https://doi.org/10.1371/journal.pcbi.1010219.g006

of pairwise models extracted with sequences sampled from the original model distribution against EVMutation couplings. For the VAE models, we chose original models with weight decay setting 0.01. As can be seen, the extracted ArDCA couplings follow the EVMutation couplings more closely than the extracted VAE couplings, although both are correlated. This can be seen as evidence that ArDCA models are closer to pairwise models directly trained on the input data. We note that this correspondence could likely be more pronounced for ArDCA by coordinating regularization strengths in EVMutation and ArDCA. This also suggests the possibility of using ArDCA with a subsequent extraction step as a training method for pairwise models, which can in general only be trained approximately for realistic sequence lengths. We leave this, however, for future research.

3.4 Mutational effect prediction using extracted models

The prediction of mutational effects is a typical field of application for the type of models analyzed in this work.

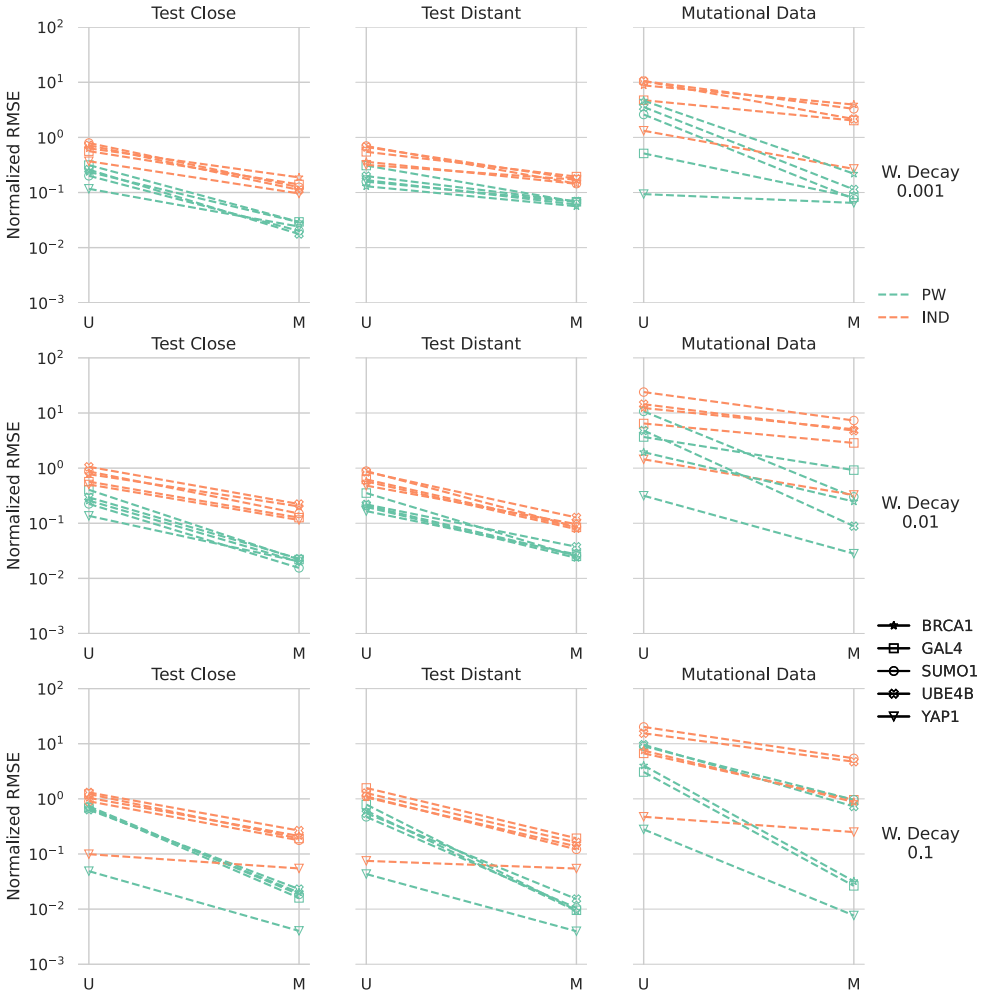


Fig 7. Errors in energies of the extracted pairwise and independent models with respect to the original VAE models. The three columns correspond to three different datasets: "Test Distant" corresponds to the 10% of sequences in the test set that have the largest distance to the training set in terms of minimum Hamming distance, "Test Close" to the remaining sequences. The right-most column corresponds to the sequences in the mutational datasets. The colors indicate whether the extracted model is an independent model (orange) or a pairwise model (green). Within every column, the left bar (U) corresponds to models extracted with samples from the uniform distribution, the right bar (M) to models extracted with samples from the distribution of the original models. The three rows correspond to different settings for weight decay during training. The error shown is the normalized root-mean squared error (see Eq 8. Note the logarithmic scale.

https://doi.org/10.1371/journal.pcbi.1010219.g007

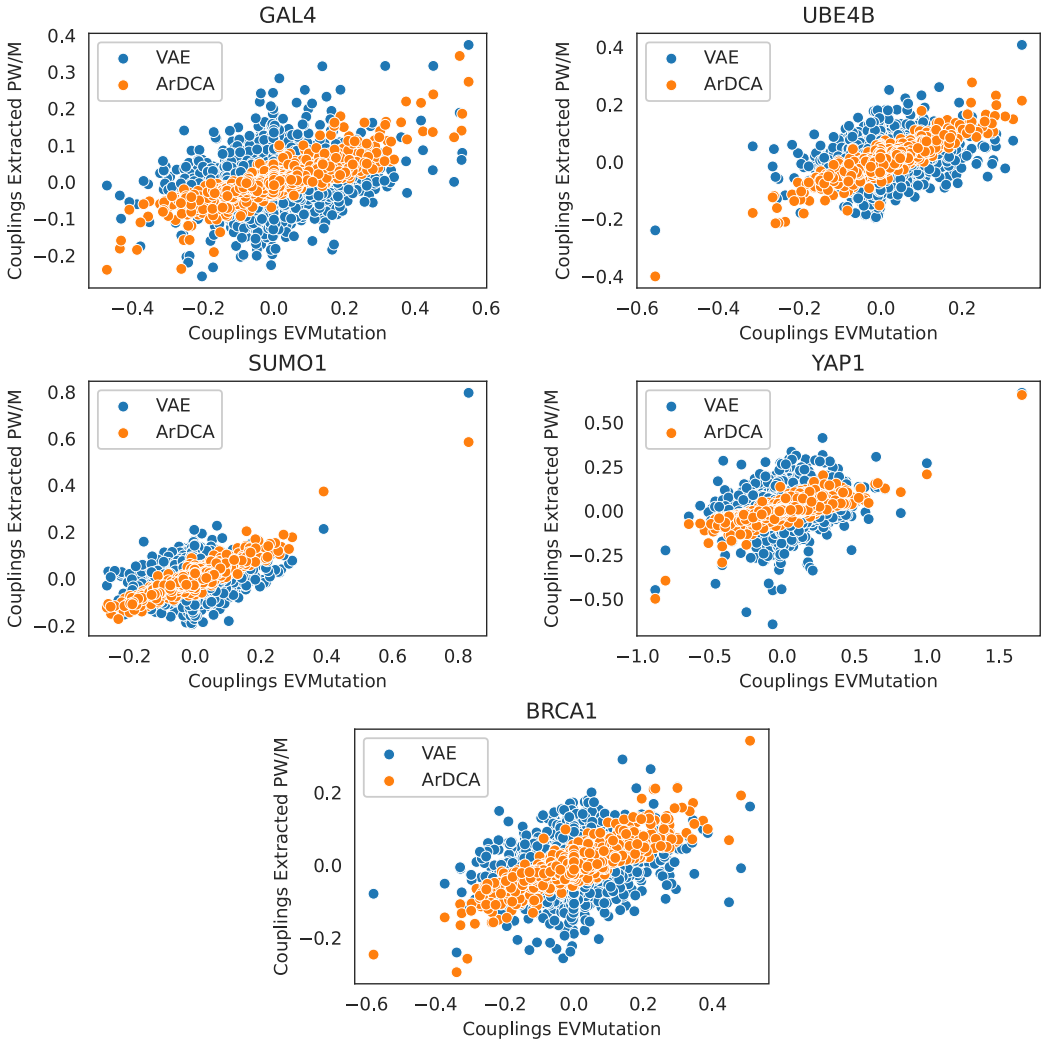


Fig 8. EVMutation couplings versus Couplings from PW/M Models extracted from ArDCA and VAE. Shown are scatterplots for all datasets of couplings as obtained from EVMutation after training versus couplings from pairwise models extracted using sequences sampled from the original model distribution, for ArDCA and VAE (weight decay setting 0.01). Given that the number of couplings is very large, we plot in all cases a subset of 5000 randomly chosen coupling pairs. Before plotting, all couplings were transformed to the zero-sum gauge.

https://doi.org/10.1371/journal.pcbi.1010219.g008

In Fig 3 we show the Spearman correlations between the experimental data and the energies in the original ArDCA models (O), the energies of pairwise and independent models extracted using samples from a uniform distribution (PW/U and IND/U) and the energies for models extracted using sequences sampled from the original model distribution (PW/M and IND/M). The red line indicates the performance of EVMutation, which is a pairwise model directly trained on the training data. The pairwise models extracted using sequences sampled from the original distribution reproduce the performance of ArDCA very closely, while pairwise models extracted using uniformly sampled sequences show a drop in performance. This corroborates the idea that while ArDCA models are not pairwise models in general, their characteristics on the part of the sequence space where they are typically used can be reproduced by a pairwise model.

Interestingly, for SUMO1 independent models extracted from the original models *outperform* the original models as well as extracted pairwise models. Since independent models are a special case of pairwise models (see Eq 5), this means that additional variability captured in the pairwise and original models hurts the performance in this case. For a possible explanation one can note that the available experimental fitness values are for sequences close to a wild-type sequence, and that the fitness landscape in this specific region might exhibit idiosyncrasies that are not mirrored in the fitness landscape at a larger scale or are even contradictory to it. Another possible explanation is experimental bias, which might systematically generate values for fitness proxies that are contradictory to the evolutionary patterns in the datasets used for training the models.

Another interesting case is BRCA1, where the original ArDCA models and the extracted independent and pairwise model perform similarly, and both outperform EVMutation. This suggests that one has to be careful when interpreting the relative performance of different model types: Considering only the values for the original ArDCA and EVMutation models, one might be tempted to conclude that ArDCA improves the predictions with respect to a pairwise model due to the capture of higher-order constraints. However, our results suggest that even an independent model can reach similar results. It is likely, therefore, that the differential performances in this case are less due to the intrinsic complexity of the model architecture but to specific choices of regularization and other training settings.

In order to test the robustness of these results, we run the pipeline a second time for the ArDCA models. Since the parameters of the original models and the fields and couplings of the extracted models are initialized to 0, the important factors in terms of stochasticity are the random seeds used for the splitting of the training and test sets, the random seeds used when sampling from the original models and the random seeds used for the stochastic gradient descent using the Adam optimizer when extracting the models. In Fig B2 in S1 Appendix we show the same plot as in Fig 3, but with all of these seeds set to a different value. The results show only minor variations.

In Fig 4 we show the results on mutation effect prediction for the VAE models for the different weight decay settings we tested. In most cases, the pairwise models extracted using sequences sampled from the original model distribution follow the performance of the original model more closely than pairwise models extracted using samples from the uniform distribution.

Interestingly, there is always a weight decay setting for the training of the original VAE model for which the independent model extracted using sequences sampled from the original model distribution performs as well or better than EVMutation, with the exception of GAL4. For SUMO1, the results corroborate the findings when using ArDCA: The original model performs better when a strong weight decay is used, and the extracted pairwise models follow this tendency. For low weight decay values, however, the independent model extracted using

sequences sampled from the original model distribution performs significantly better than the original and the extracted pairwise models. As in the case of ArDCA, this can be taken as evidence that any interaction that the original model might capture in the data is incompatible with the experimental fitness values from the mutational dataset. We plot the fitness values against the model energies and the respective ranks in Figures B4–B8 in S1 Appendix.

3.5 Contact prediction

Given that we have the explicit couplings for the extracted pairwise models, we can use standard methods from this field to predict structural contacts [27, 32] (see Section C in S1 Appendix for the contact prediction pipeline and the PDBs used). We show the results in Fig 9.

For ArDCA, the contact predictions for the extracted pairwise models are largely the same, irrespective of which distribution the sequences used for extraction come from, and also very similar to the predictions from the original method. We note that the overall performance is not particularly good. This can be explained, however, by the fact that we did not use hyperparameters optimized for contact prediction for ArDCA, but hyperparameters optimized for generative modelling (see [9]).

The results for the pairwise model extracted from the original VAE models (trained with weight decay set to 0.01) are similar, although the overall performance for contact prediction is worse. This is in line with other recent results [13], where the authors show that VAE models can learn to predict mutational effects well but structural characteristics poorly.

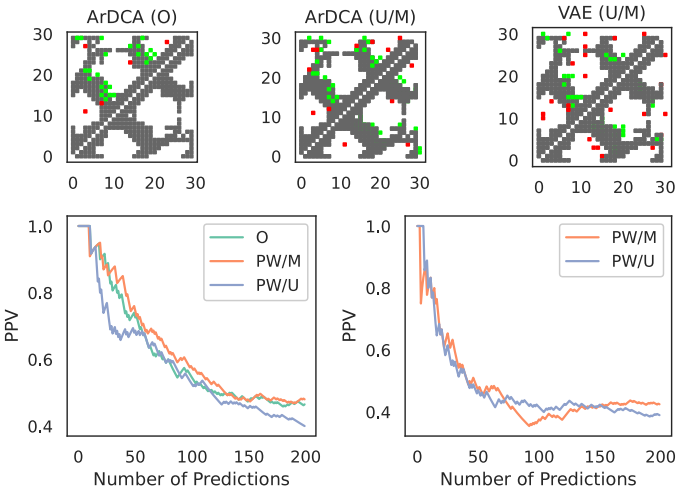


Fig 9. Contact prediction on YAP1 using extracted models. *Top Row*: Contact predictions vs. ground truth for the top $N = 30$ predicted contacts for models extracted from ArDCA and the VAE (weight decay setting 0.01). Horizontal and vertical axes show positions. True contacts are grey, true positives are green, and false positives are red. The upper parts show the contacts for models extracted with the uniform distribution, the lower parts show the same for models extracted with the original model distribution. The left-most plot shows the contact predictions for ArDCA from the original method in [9]. *Bottom Row:* PPV plots for the top 200 predictions for the original ArDCA model and the extracted pairwise model for ArDCA (left) and the VAE (right). The plots shows the fraction of true positives in dependence of the number of top predictions taken into account.

https://doi.org/10.1371/journal.pcbi.1010219.g009

4 Discussion

In this work, we provide evidence that the neural network based generative models for protein sequences analyzed by us can be approximated well by pairwise distributions in the part of the sequence space close to natural sequences, and in many cases even by the factorized distributions of independent models. The autoregressive architecture on which ArDCA is based seems to be closest to a pairwise model after training. For the VAE, the results seem to at least indicate that their pairwise projection is a very close approximation in the part of the sequence space in which they are typically used, close to the data manifold.

We cannot of course exclude that the neural network models tested by us do extract some meaningful higher-order interactions from the data, but the results seem to indicate that their effect is rather subtle. This suggests that the general strategy outlined in [33], where the pairwise part of the model is kept explicitly and an universal approximator is used for extracting higher-order interactions, might be promising. However, the current work also highlights that one has to be careful when ascribing improved performance of complex models to higher-order interactions. Apart from the fact that it is not trivial to define unambiguously what constitutes a higher-order interaction in the space of parameters due to gauge invariances, one also has to show that it is indeed the higher-order interactions that lead to the improved performance. Several works have highlighted that pairwise models, which are trained to reproduce the covariance in the data, are capable of reproducing data characteristics that are not used during training, for example three-point correlations [34]. At the same time, there are known cases where relatively clear higher-order interactions can be captured from the data and which can be included in the model, for example related to stretches of alignment gaps [35]. For the idea of combining a pairwise model with an universal approximator, this suggests that a promising approach is to regularize the combined model in way to give more prior importance to the pairwise part, and possibly to restrict the neural network to model few higher-order interactions.

Several interesting further lines of research suggest themselves. While the general idea of approximating a pairwise distribution over fixed-length sequences to models trained on unaligned data (like recent very large attention-based models [36]) seems to be ill-defined, the approach of extracting a pairwise model for a small part of the sequence space as highlighted in this work might still be feasible. Another interesting question is whether sparse higher-order interactions can be efficiently extracted from neural network based models. It is for example possible that methods like the Goldreich-Levin algorithm [37] might be adapted for pseudo-boolean functions based on generative models for protein sequence data.

Supporting information

S1 Appendix. Additional figures, mathematical derivations and a list of abbreviations.
(PDF)

Author Contributions

Conceptualization: Christoph Feinauer, Barthelemy Meynard-Piganeau, Carlo Lucibello.

Data curation: Christoph Feinauer.

Formal analysis: Christoph Feinauer, Barthelemy Meynard-Piganeau.

Investigation: Christoph Feinauer.

Methodology: Christoph Feinauer, Barthelemy Meynard-Piganeau, Carlo Lucibello.

Project administration: Christoph Feinauer.

Resources: Christoph Feinauer.

Software: Christoph Feinauer.

Validation: Christoph Feinauer.

Visualization: Christoph Feinauer.

Writing – original draft: Christoph Feinauer.

Writing – review & editing: Christoph Feinauer, Carlo Lucibello.

References

1. Balakrishnan S, Kamisetty H, Carbonell JG, Lee SI, Langmead CJ. Learning generative models for protein fold families. Proteins: Structure, Function, and Bioinformatics. 2011; 79(4):1061–1078. https://doi.org/10.1002/prot.22934 PMID: 21268112
2. Feinauer C, Weigt M. Context-aware prediction of pathogenicity of missense mutations involved in human disease. arXiv preprint arXiv:170107246. 2017;.
3. Hopf TA, Ingraham JB, Poelwijk FJ, Schärfe CP, Springer M, Sander C, et al. Mutation effects predicted from sequence co-variation. Nature biotechnology. 2017; 35(2):128–135. https://doi.org/10.1038/nbt.3769 PMID: 28092658
4. Russ WP, Figliuzzi M, Stocker C, Barrat-Charlaix P, Socolich M, Kast P, et al. An evolution-based model for designing chorismate mutase enzymes. Science. 2020; 369(6502):440–445. https://doi.org/10.1126/science.aba3304 PMID: 32703877
5. Ding X, Zou Z, Brooks CL III. Deciphering protein evolution and fitness landscapes with latent space models. Nature communications. 2019; 10(1):1–13. https://doi.org/10.1038/s41467-019-13633-0 PMID: 31822668
6. Riesselman AJ, Ingraham JB, Marks DS. Deep generative models of genetic variation capture the effects of mutations. Nature methods. 2018; 15(10):816–822. https://doi.org/10.1038/s41592-018-0138-4 PMID: 30250057
7. Hawkins-Hooker A, Depardieu F, Baur S, Couairon G, Chen A, Bikard D. Generating functional protein variants with variational autoencoders. PLoS computational biology. 2021; 17(2):e1008736. https://doi.org/10.1371/journal.pcbi.1008736 PMID: 33635868
8. Repecka D, Jauniskis V, Karpus L, Rembeza E, Rokaitis I, Zrimec J, et al. Expanding functional protein sequence spaces using generative adversarial networks. Nature Machine Intelligence. 2021; 3(4):324–333. https://doi.org/10.1038/s42256-021-00310-5
9. Trinquier J, Uguzzoni G, Pagnani A, Zamponi F, Weigt M. Efficient generative modeling of protein sequences using simple autoregressive models. Nature communications. 2021; 12(1):1–11. https://doi.org/10.1038/s41467-021-25756-4 PMID: 34608136
10. Shin JE, Riesselman AJ, Kollasch AW, McMahon C, Simon E, Sander C, et al. Protein design and variant prediction using autoregressive generative models. Nature communications. 2021; 12(1):1–11. https://doi.org/10.1038/s41467-021-22732-w PMID: 33893299
11. Madani A, McCann B, Naik N, Keskar NS, Anand N, Eguchi RR, et al. Progen: Language modeling for protein generation. arXiv preprint arXiv:200403497. 2020;.
12. Wu Z, Johnston KE, Arnold FH, Yang KK. Protein sequence design with deep generative models. Current Opinion in Chemical Biology. 2021; 65:18–27. https://doi.org/10.1016/j.cbpa.2021.04.004 PMID: 34051682
13. Marshall D, Wang H, Stiffler M, Dauparas J, Koo P, Ovchinnikov S. The structure-fitness landscape of pairwise relations in generative sequence models. bioRxiv. 2020;.
14. Zamuner S, Rios PDL. Interpretable Neural Networks based classifiers for categorical inputs. arXiv preprint arXiv:210203202. 2021;.
15. LeCun Y, Chopra S, Hadsell R, Ranzato M, Huang F. A tutorial on energy-based learning. Predicting structured data. 2006; 1(0).
16. Hinton G, Vinyals O, Dean J, et al. Distilling the knowledge in a neural network. arXiv preprint arXiv:150302531. 2015; 2(7).
17. Liu X, Wang X, Matwin S. Improving the interpretability of deep neural networks with knowledge distillation. In: 2018 IEEE International Conference on Data Mining Workshops (ICDMW). IEEE; 2018. p. 905–912.

18. Tubiana J, Cocco S, Monasson R. Learning protein constitutive motifs from sequence data. Elife. 2019; 8:e39397. https://doi.org/10.7554/eLife.39397 PMID: 30857591
19. Frazer J, Notin P, Dias M, Gomez A, Min JK, Brock K, et al. Disease variant prediction with deep generative models of evolutionary data. Nature. 2021; 599(7883):91–95. https://doi.org/10.1038/s41586-021-04043-8 PMID: 34707284
20. Durbin R, Eddy SR, Krogh A, Mitchison G. Biological sequence analysis: probabilistic models of proteins and nucleic acids. Cambridge university press; 1998.
21. Starita LM, Young DL, Islam M, Kitzman JO, Gullingsrud J, Hause RJ, et al. Massively parallel functional analysis of BRCA1 RING domain variants. Genetics. 2015; 200(2):413–422. https://doi.org/10.1534/genetics.115.175802 PMID: 25823446
22. Kitzman JO, Starita LM, Lo RS, Fields S, Shendure J. Massively parallel single-amino-acid mutagenesis. Nature methods. 2015; 12(3):203–206. https://doi.org/10.1038/nmeth.3223 PMID: 25559584
23. Weile J, Sun S, Cote AG, Knapp J, Verby M, Mellor JC, et al. A framework for exhaustively mapping functional missense variants. Molecular systems biology. 2017; 13(12):957. https://doi.org/10.15252/msb.20177908 PMID: 29269382
24. Starita LM, Pruneda JN, Lo RS, Fowler DM, Kim HJ, Hiatt JB, et al. Activity-enhancing mutations in an E3 ubiquitin ligase identified by high-throughput mutagenesis. Proceedings of the National Academy of Sciences. 2013; 110(14):E1263–E1272. https://doi.org/10.1073/pnas.1303309110
25. Araya CL, Fowler DM, Chen W, Muniez I, Kelly JW, Fields S. A fundamental protein property, thermodynamic stability, revealed solely from large-scale measurements of protein function. Proceedings of the National Academy of Sciences. 2012; 109(42):16858–16863. https://doi.org/10.1073/pnas.1209751109 PMID: 23035249
26. Liu DC, Nocedal J. On the limited memory BFGS method for large scale optimization. Mathematical programming. 1989; 45(1):503–528. https://doi.org/10.1007/BF01589116
27. Morcos F, Pagnani A, Lunt B, Bertolino A, Marks DS, Sander C, et al. Direct-coupling analysis of residue coevolution captures native contacts across many protein families. Proceedings of the National Academy of Sciences. 2011; 108(49):E1293–E1301. https://doi.org/10.1073/pnas.1111471108 PMID: 22106262
28. Stein RR, Marks DS, Sander C. Inferring pairwise interactions from biological data using maximum-entropy probability models. PLoS computational biology. 2015; 11(7):e1004182. https://doi.org/10.1371/journal.pcbi.1004182 PMID: 26225866
29. Csiszár I. Information-type measures of difference of probability distributions and indirect observation. Studia Scientiarum Mathematicarum Hungarica. 1967; 2:229–318.
30. Kingma DP, Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv:14126980. 2014;.
31. Burda Y, Grosse R, Salakhutdinov R. Importance weighted autoencoders. arXiv preprint arXiv:150900519. 2015;.
32. Ekeberg M, Lövkvist C, Lan Y, Weigt M, Aurell E. Improved contact prediction in proteins: using pseudolikelihoods to infer Potts models. Physical Review E. 2013; 87(1):012707. https://doi.org/10.1103/PhysRevE.87.012707 PMID: 23410359
33. Feinauer C, Lucibello C. Reconstruction of pairwise interactions using energy-based models. Journal of Statistical Mechanics: Theory and Experiment. 2021; 2021(12):124007. https://doi.org/10.1088/1742-5468/ac3a7f
34. Figliuzzi M, Barrat-Charlaix P, Weigt M. How pairwise coevolutionary models capture the collective residue variability in proteins? Molecular biology and evolution. 2018; 35(4):1018–1027. https://doi.org/10.1093/molbev/msy007 PMID: 29351669
35. Feinauer C, Skwark MJ, Pagnani A, Aurell E. Improving contact prediction along three dimensions. PLoS computational biology. 2014; 10(10):e1003847. https://doi.org/10.1371/journal.pcbi.1003847 PMID: 25299132
36. Meier J, Rao R, Verkuil R, Liu J, Sercu T, Rives A. Language models enable zero-shot prediction of the effects of mutations on protein function. bioRxiv. 2021;.
37. O'Donnell R. Analysis of boolean functions. Cambridge University Press; 2014.

Chapter 3

Conditioning on Structural Information

3.1 Introduction to InvMSAFold: Expanding the Horizon of Protein Sequence Diversity

The effort to understand and design proteins has greatly advanced with the help of deep learning models. These models are very good at predicting protein structures and interactions, which has opened up new ways to design proteins. After learning that focusing on second-order interactions is crucial to understanding protein behavior and distribution, our latest research introduces an advanced development called InvMSAFold.

Building upon our exploration of generative models for protein sequences, we advance our research by integrating additional guidance into the generation process. Our objective is to incorporate external information into our generative models. In this study, we achieve this by conditioning the generative process on the structural information of proteins. This method leverages the foundational understanding of protein sequence and structure relationships, aiming to generate a diverse set of protein sequences that fold into a specified structure.

3.1.1 From Model Distillation to Sequence Diversity

The preceding work on model distillation highlighted the significance of second-order correlations in understanding the generative capabilities of DL models for protein sequences. Building on this foundation, InvMSAFold seeks to extend the application of these insights, by applying the idea of second-order interaction and reconstruction to the field of inverse folding. We remind that inverse folding represents the task of generating the sequence for a fixed fold. For example, proteinMPNN [53] is known to produce sequences that fold extremely close to the actual protein structures, often resembling them more closely than the variations found within a protein family. This can lead to the fitting of structural details, where the model might be focused on mimicking specific structures rather than capturing broader functional

properties. We transition from the recovery of single sequences to the generation of entire datasets, aiming at capturing the entire diversity of sequences in this family.

3.1.2 Innovations and Contributions of InvMSAFold

InvMSAFold introduces a methodological advancement in the world of inverse folding by focusing on defining a probability distribution over the entire space of sequences that can fold into a given structure. This distribution is designed to capture the second-order correlations observed in MSA of homologous proteins, facilitating the generation of protein sequences that are not only structurally consistent with the target fold but also exhibit a high degree of sequence diversity.

Natural homologs typically fold into nearly identical structures, and InvMSAFold aims to replicate this by generating sequences that not only fold into almost the same structure but also encompass the diversity of sequences possible for that structure. This approach mirrors the variation seen in a protein family, where despite minor differences, the sequences are structurally compatible. Therefore it varies from traditional inverse folding algorithm that aims at recovering a specific sequence and consequently generates much less diversity. The real challenge here is to explore

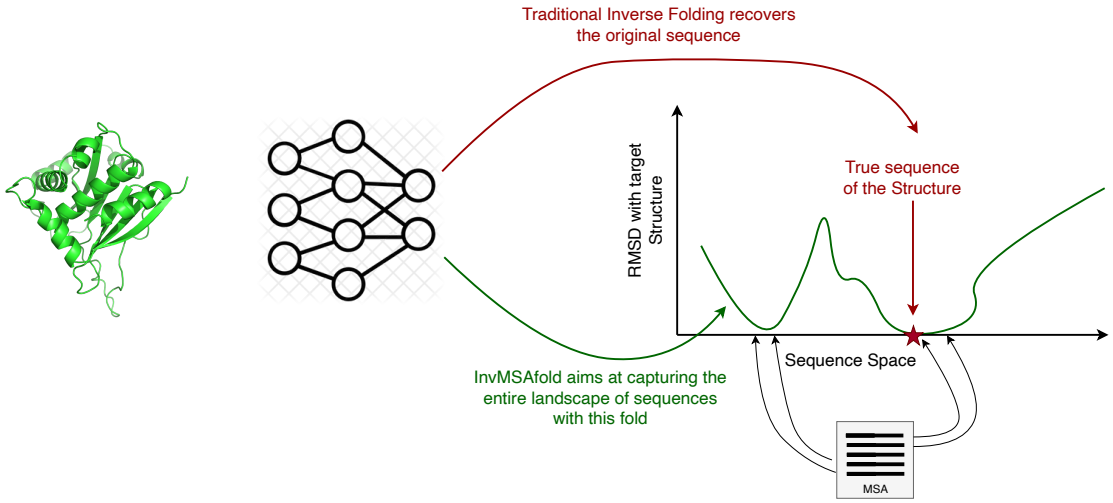


Figure 3.1: This figure summarizes the concept behind this project. From an input backbone, traditional inverse folding uses a neural network to predict the original sequence of the backbone. However, the space of sequences able to fold (or nearly fold) into a predefined shape is vast. It is shown schematically on the right where we map each sequence to the RMSD of its fold with respect to the input structure. Of course, the sequence has an RMSD of 0, but the space of low RMSD is much larger. The green arrow shows the aim of InvMSAfold to recover the complete landscape and retrieve the full width of sequences with an input fold.

and describe the sequence space that is compatible with a given structure.

Our empirical evaluations demonstrate InvMSAFold's high performance in preserving structural integrity, as evidenced by comparative analyses with the ESM-IF1 model. Metrics such as RMSD score indicate a more faithful retention of the native structure, even for sequences significantly divergent from the original. This

achievement underscores the method's effectiveness in balancing sequence diversity with structural fidelity, offering valuable insights into protein adaptation and engineering.

3.2 Paper: InvMSAFold

Uncovering sequence diversity from a known protein structure

Luca Alessandro Silva[1], Barthelemy Meynard-Piganeau[1,2,3], Carlo Lucibello[1], and Christoph Feinauer[1]

[1]Department of Computing Sciences, Bocconi University, Milan, Italy
[2]Politecnico di Torino, Milan, Italy
[3]Sorbonne Université, Institut de Biologie Paris Seine, Biologie Computationnelle et Quantitative LCQB, Paris, France

Abstract

We present InvMSAFold, a method for generating a diverse set of protein sequences folding into a single structure. For a given structure it defines a probability distribution over the space of sequences. This distribution captures second-order correlations observed in Multiple Sequence Alignments (MSA) of homologous proteins. Our innovation lies in generating highly diverse protein sequences while preserving structural and functional integrity. This approach offers exciting prospects, particularly in directed evolution, by providing diverse starting points for protein design.

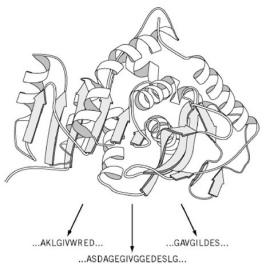
1 Introduction

Inverse folding aims to predict amino acid sequences that fold into a given protein structure and plays a fundamental role for example in the protein design pipeline of RFDiffusion [1]. Recent deep learning approaches such as ESM-1F [2] or ProteinMPNN [3] achieve remarkable accuracies in this task. However, instead of predicting a single ground truth sequence, it is often desirable to have a model that is able to generate a variety of different sequences, for example starting from a source sequence [4, 5] and taking different molecular environments into consideration [6]. Such approaches allow to expand the sequence design space while preserving structural consistency, allowing for a larger pool of sequence when selecting for additional properties like thermostability, solubility or toxicity.

In this work, we present a method that is able to generate diverse protein sequences given a structure, including sequences far away from the natural sequence. Our method is potentially applicable in various domains. In drug discovery, for example, it would allow for the generation of large amount of diverse candidates, enabling further selection optimized for properties like bioavailability. Similarly, in biotechnology and enzyme engineering, it could facilitate the creation of enzymes with tailored properties, such as improved stability and activity under varying conditions.

Recent architectures for inverse folding are based on encoder-decoder architectures, where a structure is encoded and a sequence decoded. Such models then generally take into account only the native sequence of a given structure, trying to maximise its output probability. While such an approach maximises native sequence discovery, it might fail to properly model the sequence diversity known to exist for a given structure(EXPAND).

37th Conference on Neural Information Processing Systems (NeurIPS 2023).



In our approach, we veer off from this paradigm, and instead use the decoder to generate the parameters of a pairwise probability distribution which, rather than modelling solely the native sequence, models the sequence diversity of the *multiple sequence alignment*(MSA) of the native sequence [7]; i.e. we aim at capturing the broader probability distribution of sequences that correspond to a specific fold, rather than maximising the native sequence output probability. As for the output probability distribution we chose *direct coupling analysis (DCA)* [8], which is a statistical model that has proven to describe well the amino acid statistics of protein families by reproducing the second-order correlation of the Multiple Sequence Alignment (MSA), and has proven to be a good generative model [9]. DCA trains a Potts model, which originates from statistical mechanics and is a generalization of the Ising model for ferromagnetism.

Homologous sequences are known to have very akin folds, hence DCA is well-suited to efficiently capture the diversity we want to uncover. Indeed in [10], the exploration of large scale library of mutants, enabled to estimate the importance of different order for capturing the fitness. It revealed that first and second order are pivotal in this understanding, contrary to the others statistics. While we use MSAs of families of homologous proteins when training, during inference we only use the structure of a protein in order to generate such a pairwise model. The generated pairwise model itself is very light-weight and can be used for rapidly generate a large diversity of sequences for the input structure.

We show that the models we generate are able to create sets of sequences that capture the diversity of the protein family better than other models and are able to find sequences far away from the natural sequence that are predicted to still fold into the same structure.

2 Methods

In this Section, we describe the components of our architecture, InvMSAFold, and its training procedure.

Inverse folding methods typically define a probability distribution $p(\boldsymbol{s}|\boldsymbol{X})$ for a sequence $\boldsymbol{s}$ given backbone coordinates $\boldsymbol{X}$. Most deep learning based methods structure this distribution auto-regressively, using $p(\boldsymbol{\sigma}|\boldsymbol{X}) = \prod_{i=1}^{N} p(\sigma_i|\sigma_{i-1}, \ldots, \sigma_1, \boldsymbol{X})$, where N is the sequence length, and train by minimizing the loss of the true sequence for the backbone coordinates $\boldsymbol{X}$.

Both of these choices have drawbacks: Sampling amino acids auto-regressively requires a full forward pass through the neural network for every generated token, making it very expensive to use the models in a virtual-screening like setting, where a large number of sequences are scanned for properties beyond folding into structure $\boldsymbol{X}$. Secondly, minimizing the loss on the true sequence ignores the fact that there are many different sequences that might fold into the same structure. Even if the training is successful, one would expect the resulting distribution to be peaked around very few sequences and not capturing other parts of sequence space that might be interesting for the problem one tries to solve.

The InvMSAFold architecture. In this paper, we propose InvMSAFold, which deviates from the type of methods described above in two important ways: Instead of directly returning the probability distribution over amino acids, we use the neural network to generate a set of parameters for a secondary probability distribution that is then used for sampling amino acids, i.e. we define $p(\boldsymbol{\sigma}|\boldsymbol{X}) = p(\boldsymbol{\sigma}|\boldsymbol{\theta}(\boldsymbol{X}))$, where $\boldsymbol{\theta}(\boldsymbol{X}))$ are parameters of a light-weight probability distribution generated by a neural network that is given the backbone coordinates $\boldsymbol{X}$ as input.

We explore different choices for parameterizing the distribution. In order to go beyond distributions that treat different positions in the protein as independent and ensure sufficient expressivity, we focus on pairwise models [7], which have an experimentally validated ability capture structure-sequence relationships [9]. We therefore train the neural network to learn on the mapping $\boldsymbol{X} \rightarrow \Theta = \boldsymbol{J}, \boldsymbol{h}$,

where $\boldsymbol{J}$ and $\boldsymbol{h}$ are parameter tensors of size $N \times q \times N \times q$ resp. $N \times q$. As is common in pairwise models, we call the quantities $h_i(a)$ the *fields*, indexed by position i and amino acid type a the *fields*, and the quantities $J_{ij}(a, b)$, indexed by a pair of positions and a pair of amino acid types, the *couplings*. The former describe the propensity of an amino acid to appear at a given position, while the latter describe the propensity of pairs of amino acids appear at pairs of positions. We show below how to define a distribution over sequences of amino acids given these parameters.

The complete InvMSAFold architecture neural network composed of two parts, the structure encoder and a decoder which outputs the fields and couplings, see Fig. 1.

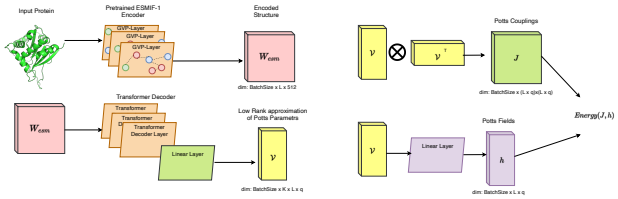


Figure 1: Left: Decoder architecture up to the outputting of the low-rank tensor $\mathcal{V}$. Right: how $\mathcal{V}$ is used to produce couplings and fields for the Potts model.

For the encoder, we use pre-trained encoder ESM-IF1 model [2] which follows the GVP-GNN architecture of [11]. This encoder gives a rotationally invariant representation of the input $\boldsymbol{X}$.

The decoder takes as input batches of encoded structures, padded to a common length L. The sequence is embedded to a $L \times D$ tensor ($D = 512$ in our experiments) and passed through 6 transformer layers with 8 attention heads. The output is projected to a $L \times K \times q$ tensor V, with $K < L$. Using V, we obtain a low-rank coupling matrix as follows:

$$J_{ij}(a, b) = \sum_{k=1}^{K} v_i^k(a) v_j^k(b). \tag{1}$$

We note that in other settings, low-rank decompositons of J (1) have been shown to be as effective as the full-rank counterparts [12].

The $\mathcal{O}(L)$ values for the fields $\boldsymbol{h}$ are predicted directly as part of the network output.

Pairwise and Autoregressive Pairwise Distributions Given the parameters J and h we explore two different approaches to defining probability distributions over amino sequences. In all cases, the distribution is fully determined by J and h and we do not need an MSA in inference.

The first distribution, a standard pairwise distribution [7], defines the probability p^{pw} as

$$\log p^{pw}(\boldsymbol{\sigma}|\boldsymbol{h}, \boldsymbol{J}) \sim \sum_{i=1}^{N} \sum_{j=i+1}^{N} J_{ij}(\sigma_i, \sigma_j) + \sum_i h_i(\sigma_i). \tag{2}$$

While this distribution has been explored extensively for proteins [12], it has the disadvantage that the normalization factor for Eq. 2 is intractable, and therefore also the likelihood. We therefore use pseudo-likelihoods for inference [13] and resort to MCMC for sampling.

As an alternative, we also consider an efficient auto-regressive variant of pairwise models [14], called ArDCA, which defines the autoregressive distribution p^{ar} over amino acids in a sequence as

$$\log p^{ar}(\sigma_i|\sigma_1, \ldots, \sigma_{i-1}, \boldsymbol{h}, \boldsymbol{J}) \sim h_i(\sigma_i) + \sum_{j=1}^{i-1} J_{ij}(\sigma_i, \sigma_j). \tag{3}$$

This parameterization has the advantage that the normalization factor is tractable. This allows for calculating the likelihood explicitly in training and inference and also for efficient sampling.

Since the number of parts in the sum in Eq. 3 depends on the position i, we rescale the couplings as

$$J_{ij} \leftarrow \frac{J_{ij}}{max(i,j)}, \tag{4}$$

see [15]. We found this to be beneficial for training and note that without this scaling the neural network would have to generate couplings of significantly different magnitudes for different sites, something that is not a problem if we were to optimize the couplings directly, as in previous research [14].

Training on Homologous Sequences In most other works [2], inverse folding models are training to predict the ground truth sequences only. In this work, we aim to generate a distribution that captures the complete sequence space that is compatible with the input structure $\boldsymbol{X}$. To this end, we use the ground truth sequence corresponding to a structure $\boldsymbol{X}$ and extract an MSA M_X from sequence database (see next section for details). We then use the pairs $(\boldsymbol{X}, M_X)$ for training by taking the mean negative pseudo log-likelihood of a random subsample of sequences in M_X given the parameters generated by the network.

For both model specifications (2), (3) we then add a regularizing L_2 term for both the fields and the couplings; this should concurrently solve the gauge-invariance inherent in the models, while more importantly oppose overfitting.

For the first standard pairwise distribution (2) we select a batch size of 1, a subsample size for M_X of 64, rank K of 48, a learning rate of 10^{-4} and equal penalties $\lambda_h = \lambda_J = 10^{-4}$ for fields and couplings, while for (3) we select those parameters trough hypertuning. We refer the reader to the dedicated appendix subsection A.3 for the details. Both models are trained with AdamW optimizer for a total of 94 epochs.

3 Results

3.1 Data Collection and Preprocessing

To train and evaluate models we rely on the curated CATH [16] 4.2 40% non-redundant data set, in which structures are grouped by their CATH (class, architecture, topology/fold and homologous superfamily) classification. We leverage this dataset to evaluate the generalization performance of our models in three different settings. To this end, we split the CATH dataset into a training set and three testing sets, which we label respectively *sequence, structure and superfamily*. Sequences in the *sequence* testing set have homologous sequences in the training set. Sequences [continue here]

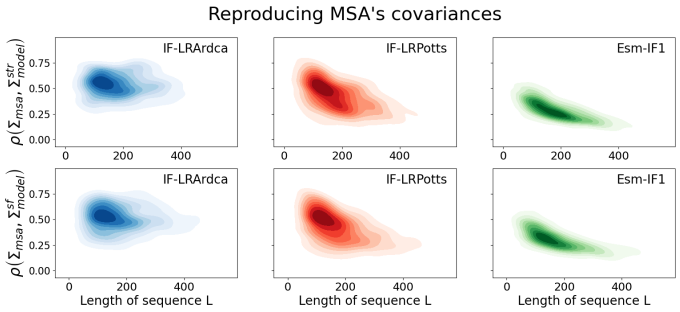
80% of the sequences are divided into train and sequence testing data set: the latter contains those structures of which we have seen an homologous during the training. The structure test data set, which contains 10% of sequences, has structures not seen during training, but of which we have seen a structure belonging to the same superfamily. Finally, in the superfamily test dataset we have sequences of which we have not even seen an element belonging to the same superfamily during training: these sequences hence have very different folds from the one seen during training, and hence test our models capacity to generalize to unseen structures.

Once we have divided the structures of CATH as detailed above, for each structure we generate a MSA searching for homologous sequences inside the comprehensive Uniprot50 data set. To obtain the MSAs we leverage the MMseqs2 (Many-against-Many sequence searching) software, which allows for accurate and fast generation of many MSAs.

3.2 Second Order Reconstruction

In this paper section, we highlight the importance of looking at pairs of amino acids, known as second-order reconstruction, in machine learning for protein domains. Taking Direct Coupling Analysis (DCA) as an example, we move beyond studying individual amino acids and focus on understanding the relationships between pairs of residues in protein sequences. DCA, recognized for

its ability to capture co-evolutionary patterns, serves as a key method in reproducing essential second-order information. This approach is particularly crucial for grasping the complex protein landscape, especially within protein domains where local interactions play a significant role. Prioritizing the analysis of these pairs enhances the accuracy of predicting protein structures, providing deeper insights into how amino acids collaborate. The ability of a generative model to reproduce second order has been proven to be good metric for measuring how well the protein landscape is captured. Reproducing well this pattern enables efficient sampling of functional protein [9]. We compared the three different models in their ability to reproduce the covariance matrix of the MSA for structures belonging to the structure and super-family test data sets. For details on the experimental procedure, we refer to the Appendix.



Quartiles	Ardca		Potts		ESM	
	Superfamily	Structure	Superfamily	Structure	Superfamily	Structure
First quartile	**0.43**	**0.45**	**0.50**	**0.53**	**0.60**	**0.60**
Median	0.31	0.31	0.43	0.42	0.53	0.53
Third quartile	0.23	0.21	0.29	0.28	0.37	0.35

Figure 2: Comparing the correlations between generated and true samples for ArDCA(blue), potts(red) and ESM-IF1(green). On the top we have the results for the sequence in the structure test dataset, on the center of the superfamily test dataset and on the bottom we have a table giving a more quantitative analysis of the above plots

As we can see from Figure **??**, ArDCA and Potts do both significantly better than ESM-IF1. ArDCA seems also to dominate potts; especially its performance does not seem to deteriorate with the length of the input sequence, while especially ESM-IF1, and to a lesser extent Potts, exhibit this feature. For a more quantitative analysis, we can look at Table 2; as we can see the results is robust to the two test dataset under consideraion.

Such a result signal that both Ardca and Potts are able to better learn the diversity within the MSA, and as a result then are better suited to generate sequences that better explore the complexity of the landscape of a MSA. On the other hand, esm1F, by only focusing on the native sequence, is not able to capture it.

3.3 Sequence Space Exploration

In this subsection, we delve into an experimental exploration aimed at comparing the efficacy of different sampling methods in navigating the sequence space, utilizing a natural sequence multiple sequence alignment (MSA) as a reference benchmark. Our investigation centers on the principal component analysis (PCA) of the natural sequence MSA, serving as a foundational framework for subsequent analysis.

We initiated our investigation by computing the PCA component of the natural sequence MSA, providing a robust foundation for comparative analysis. Subsequently, we conducted sampling

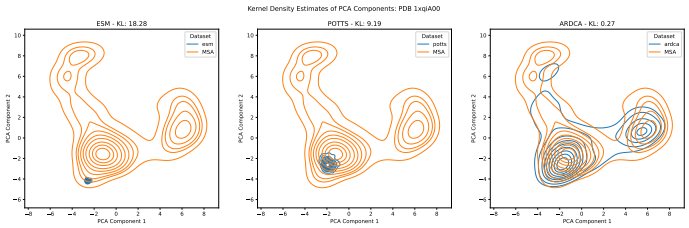


Figure 3: This plot demonstrates the capacity of InvMSAfold ardca to explore the space. First, the two main principal components of the natural were extracted. We then used esm and our two versions of InvMSAfold to generate 2000 sequences conditioned on the pdb 1xqiA00. We projected all these sequences on the two main components computed before and applied a kernel density estimation. The three plots represent the diversity of the three models compared with the diversity of the natural sequences. We also used this density estimate to compute the Kullbach-Leiber (KL) divergence between the density of the natural data and the density of the sampled data. The values of these KLs are written in the title of each subplots.

	esm	potts	ardca
Structure	15.8	4.6	**0.49**
Superfamily	11.9	11.2	**0.67**

Table 1: Kullbach-Lieber Divervence between between Kernel Density of the natural sequence and sampled sequence. More precisely, for each method (esm, ports, ardca we sampled 2000 sequences for each backbone. We then project these sequences on the two main PCA components of the natural MSA. In this 2D space, we apply a Gaussian kernel density estimator of kernel size 1.0. We then use these densities to compute the KL divergence between the space of natural sequences and the one generated by the inverse folding approach.

experiments from three distinct models: ESMLfold, INVMSAfold-Potts, and INVMSAfold-ArDCA. A total of 2000 sequences were sampled from each model to capture a diverse representation of the sequence space.

Plotting the projection onto the first two PCA components facilitated a comparative examination of the exploration patterns exhibited by the sampling methods. Notably, the distribution of sequences sampled from ESMLfold revealed a strikingly narrow focus around a single point in the sequence space. This observation suggests a limited exploration capability inherent in the ESMLfold model, potentially constraining its utility in capturing diverse sequence variants.

In contrast, sequences sampled from INVMSAfold-Potts displayed broader exploration across the sequence space. However, a predominant tendency towards unimodal distributions hinted at a propensity for the model to converge towards specific sequence motifs or modes. While INVMSAfold-Potts exhibited a broader exploration scope compared to ESMLfold, its exploration remained somewhat constrained within individual modes.

Remarkably, sequences sampled from INVMSAfold-ArDCA showcased a distinctly different exploration pattern characterized by comprehensive exploration across the sequence space. Unlike its counterparts, INVMSAfold-ArDCA demonstrated an ability to detect and explore multiple modes within the sequence landscape. This enhanced versatility and sensitivity highlight the potential of INVMSAfold-ArDCA as a powerful tool for navigating complex sequence spaces and capturing diverse sequence variants.

Our comparative analysis sheds light on the differential exploration capabilities of sampling methods within the context of sequence space. While ESMLfold and INVMSAfold-Potts exhibit limited exploration and propensity towards unimodal distributions, INVMSAfold-ArDCA stands out for its comprehensive exploration and detection of multiple modes. These findings underscore the importance of employing robust sampling methods, such as INVMSAfold-ArDCA, for effective navigation

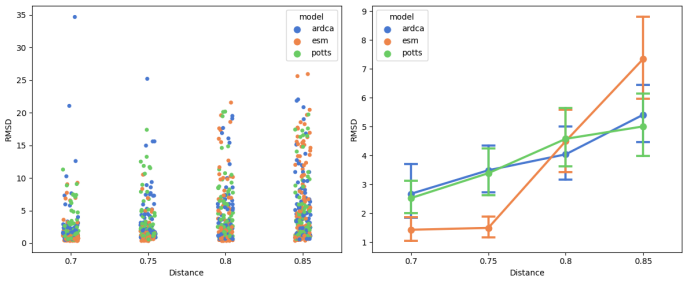


Figure 4: comparison of the quality of the generated sequence under the constraint of increasing hamming distance (normalized by protein length) from the native sequence. We started aggregated the score of 14 pdbs from the superfamily set, and sampled from ESM-IF1 and our methods. We then refold the sequence with alphafold and compare the refolded structure with the original one using the RMSD. We observe the ESM-IF1 is not able to generate good sequences far away from the native one. List of pdb used: 1otjA00 2ytyA00 1p9hA00 2o0bA02 1b34B00 1f6mA02 2hs5A01 2bh8A02 2de6A03 1ia6A00 4gc1A01 3sobB02 1xqiA00 4yt9A01

of sequence space and the discovery of diverse sequence variants with potential implications across various domains of biological research and protein engineering.

3.4 Iso-Structure Exploration

In our pursuit of understanding the relationship between protein sequence variations and structural deformations, we designed a controlled experiment that systematically explores the tolerance of a protein structure to increasing sequence dissimilarity. This experiment provides a unique perspective on protein structural robustness and represents a novel application of state-of-the-art computational tools.

For a given protein structure of interest, we initiated the experiment by generating a spectrum of sequence variants. We systematically increased the hamming distance of these variants from the original native sequence. This divergence was achieved by progressively altering amino acids, thereby introducing increasing levels of sequence dissimilarity while preserving the protein's overall fold.

To assess the impact of these sequence variations on the protein structure, we employed a temperature-based exploration approach. Starting from sequences with minimal dissimilarity from the native sequence, we gradually elevated the temperature. This step allowed us to reach sequences that were further and further removed from the native sequence, effectively increasing the structural diversity under consideration.

Following the generation of these sequence variants, we utilized the AlphaFold protein folding model [17, 18] to refold the sequences. AlphaFold has demonstrated exceptional capabilities in predicting protein structures accurately. We leveraged its predictive power to refold the diverse set of sequences generated at different levels of sequence divergence.

To evaluate the structural fidelity of the refolded sequences, we conducted a comparative analysis with the ESM-IF1 model, a prominent computational tool in structural bioinformatics. Specifically, we measured the structural similarity between the refolded sequences and the native structure using the RMSD 5, 4. Alphafold was used with no templates, and mmseq2 was used for the msa.

4 Melting temperature prediction

INVMSAfold is a valuable tool in molecular biology and bioengineering as it diverges from conventional sequence-structure prediction models by prioritizing the generation of a diverse set of sequences that adhere to a fixed structural fold. This unique approach has applications in protein engineering, drug design, where the exploration of sequence space within a specified structural motif is crucial for

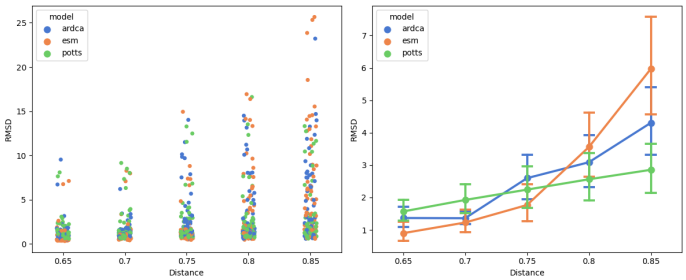


Figure 5: comparison of the quality of the generated sequence under the constraint of increasing hamming distance (normalized by protein length) from the native sequence. We started aggregated the score of 14 pdbs from the structure set, and sampled from ESM-IF1 and our methods. We then refold the sequence with alphafold and compare the refolded structure with the original one using the RMSD. We observe the ESM-IF1 is not able to generate good sequence far away from the native one. List of pdb used: 5i0qB02 4iulB00 2wfhA00 2egcA00 3oz6B02 3m8bA01 2pz0B00 2cnxA00 4k7zA02 1dl2A00 1vjkA00 1udkA00 3bs9A00

discovering novel functional variants, designing targeted drug candidates. The model's applicative strength lies in its ability to tailor sequences, enhancing for example thermal stability, mitigating susceptibility to proteolytic degradation, or reducing potential cytotoxicity, all while upholding the foundational protein structure and function. Anchored in this principled approach, INVMSAfold aims to contribute to the systematic enhancement of lead proteins.

Protein stability refers to the ability of a protein to maintain its structural and functional integrity under various environmental conditions. While different environmental factors can affect protein stability, thermal stability is an important property of proteins, as many biological processes occur at specific temperatures. Proteins that are less thermally stable are more prone to aggregate at physiological temperatures leading to loss of activity, dysfunction or even the formation of toxic protein aggregates.The thermal stability of a protein can be measured by its denaturation or melting temperature (T_m), which is the temperature at which 50% of the protein loses its native structure and activity or alternatively defined as the area under the melting curve.

In the past, measuring the stability of proteins required extensive work and resulted in limited data; while the advancement of mass spectrometry-based thermal proteome profiling (TPP) has allowed the development of an atlas of the thermal stability for roughly $50k$ proteins across different species, the technical limitations, cost, and the labor-intensive nature of the experimental approach limit the proteome and species amenable to thermal profiling. As a result, predicting protein thermal stability has become a modern solution when experimental data is incomplete or when assessing thermal stability is not easy. Recently a novel machine learning architecture, DeepSTABp [19], displayed reliable and SOTA performance in regression-based prediction of cellular thermal protein stability. Leveraging DeepStab, we want to assess the predicted T_m of the proteins coming from the different models.

To do so, we follow a procedure analogous to that one of the iso-structure exploration; We generate proteins from the different models from both the structure and superfamily test dataset under different experimental conditions, and compare the predicted T_m, including also a sub-sample from the MSA. To avoid clutter we avoid reporting details of the experiment here and refer the reader to the appendix;

As we can see by Figure 6 and Figure 7, both Potts and Ardca outperform consistently esm in predicted T_m across different *optimal growth temperatures* and CATH superfamilies. This suggests that both Potts and arDCA,by leveraging the MSA's diversity during training, are able to generate explore better the MSA space of a given structure, and generate more thermostable proteins at higher

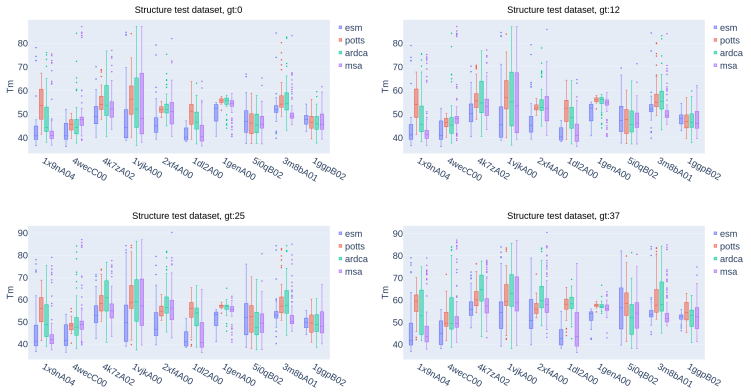


Figure 6: Predicted melting temperatures for different domains inside the structure test dataset at different optimal growth temperatures and lysate TPP enviroment. Box plots are created from 50 synthetic sequences from every model at different hamming distance from the native one. For the MSA we randomly select 100 sequences from the MSA.

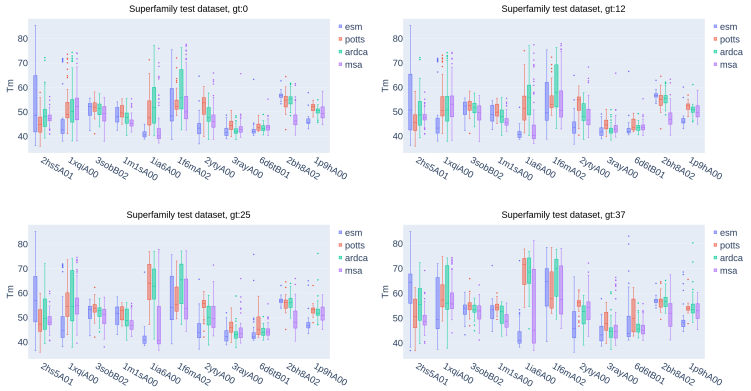


Figure 7: Predicted melting temperatures for different domains inside the superfamily test dataset at different optimal growth temperatures and lysate TPP enviroment. Box plots are created from 50 synthetic sequences from every model at different hamming distance from the native one. For the MSA we randomly select 100 sequences from the MSA.

hamming distance than esm both for homeotherminc and heterothermic organisms. Such a feature could be very valuable in a protein design feature, where it would allow to subsample proposed proteins for thermostability ..... (extend here)

5 Discussion

Our findings reveal that our approach consistently outperforms the ESM-IF1 model in preserving the native structure of the protein for sequences that are far removed from the original native sequence. The RMSD score and pLDDT metrics consistently demonstrate a higher level of structural fidelity in the refolded sequences generated using our approach.

This experimental setup and comparative analysis underscore the effectiveness of our methodology in exploring the structural consequences of sequence diversity, emphasizing its potential in applications ranging from protein engineering to understanding the adaptation of proteins in diverse environments.

Our study marks a significant shift in the domain of inverse folding, where the conventional focus was on decoding the original sequence from a known structure. In contrast, our research addresses a more realistic scenario where we possess the native sequence but seek to generate alternative sequences. This novel approach, aimed at expanding sequence diversity while preserving the fold, carries profound implications for a range of applications. By systematically generating alternative sequences for a given protein structure, we open new avenues for filtering and selecting sequences based on desired properties, such as improved thermostability, altered substrate specificity, or reduced toxicity.

References

[1] 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, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, 2023.

[2] Chloe Hsu, Robert Verkuil, Jason Liu, Zeming Lin, Brian Hie, Tom Sercu, Adam Lerer, and Alexander Rives. Learning inverse folding from millions of predicted structures. *bioRxiv*, 2022.

[3] Justas Dauparas, Ivan Anishchenko, Nathaniel Bennett, Hua Bai, Robert J Ragotte, Lukas F Milles, Basile IM Wicky, Alexis Courbet, Rob J de Haas, Neville Bethel, et al. Robust deep learning–based protein sequence design using proteinmpnn. *Science*, 378(6615):49–56, 2022.

[4] Pascal Sturmfels, Roshan Rao, Robert Verkuil, Zeming Lin, Ori Kabeli, Tom Sercu, Adam Lerer, and Alexander Rives. Seq2msa: A language model for protein sequence diversification.

[5] Drew H Bryant, Ali Bashir, Sam Sinai, Nina K Jain, Pierce J Ogden, Patrick F Riley, George M Church, Lucy J Colwell, and Eric D Kelsic. Deep diversification of an aav capsid protein by machine learning. *Nature Biotechnology*, 39(6):691–696, 2021.

[6] Lucien Krapp, Fernado Meireles, Luciano Abriata, and Matteo Dal Peraro. Context-aware geometric deep learning for protein sequence design. *bioRxiv*, pages 2023–06, 2023.

[7] Simona Cocco, Christoph Feinauer, Matteo Figliuzzi, Rémi Monasson, and Martin Weigt. Inverse statistical physics of protein sequences: a key issues review. *Reports on Progress in Physics*, 81(3):032601, 2018.

[8] Matteo Figliuzzi, Pierre Barrat-Charlaix, and Martin Weigt. How Pairwise Coevolutionary Models Capture the Collective Residue Variability in Proteins? *Molecular Biology and Evolution*, 35(4):1018–1027, 01 2018.

[9] William P Russ, Matteo Figliuzzi, Christian Stocker, Pierre Barrat-Charlaix, Michael Socolich, Peter Kast, Donald Hilvert, Remi Monasson, Simona Cocco, Martin Weigt, et al. An evolution-based model for designing chorismate mutase enzymes. *Science*, 369(6502):440–445, 2020.

[10] FJ Poelwijk, M Socolich, and R Ranganathan. Learning the pattern of epistasis linking genotype and phenotype in a protein. biorxiv: 213835, 2017.

[11] Bowen Jing, Stephan Eismann, Patricia Suriana, Raphael JL Townshend, and Ron Dror. Learning from protein structure with geometric vector perceptrons. *arXiv preprint arXiv:2009.01411*, 2020.

[12] Simona Cocco, Rémi Monasson, and Martin Weigt. Inference of hopfield-potts patterns from covariation in protein families: calculation and statistical error bars. In *Journal of Physics: Conference Series*, volume 473, page 012010. IOP Publishing, 2013.

[13] Magnus Ekeberg, Cecilia Lövkvist, Yueheng Lan, Martin Weigt, and Erik Aurell. Improved contact prediction in proteins: using pseudolikelihoods to infer potts models. *Physical Review E*, 87(1):012707, 2013.

[14] Jeanne Trinquier, Guido Uguzzoni, Andrea Pagnani, Francesco Zamponi, and Martin Weigt. Efficient generative modeling of protein sequences using simple autoregressive models. *Nature communications*, 12(1):1–11, 2021.

[15] Simone Ciarella, Jeanne Trinquier, Martin Weigt, and Francesco Zamponi. Machine-learning-assisted monte carlo fails at sampling computationally hard problems. *Machine Learning: Science and Technology*, 4(1):010501, 2023.

[16] Ian Sillitoe, Nicola Bordin, Natalie Dawson, Vaishali P Waman, Paul Ashford, Harry M Scholes, Camilla SM Pang, Laurel Woodridge, Clemens Rauer, Neeladri Sen, et al. Cath: increased structural coverage of functional space. *Nucleic acids research*, 49(D1):D266–D273, 2021.

[17] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[18] Milot Mirdita, Konstantin Schütze, Yoshitaka Moriwaki, Lim Heo, Sergey Ovchinnikov, and Martin Steinegger. Colabfold: making protein folding accessible to all. *Nature methods*, 19(6):679–682, 2022.

[19] Felix Jung, Kevin Frey, David Zimmer, and Timo Mühlhaus. Deepstabp: A deep learning approach for the prediction of thermal protein stability. *International Journal of Molecular Sciences*, 24(8), 2023.

Chapter 4

Conditioning Generation on interacting sequence

Advancing computational protein design to include interaction partners as a conditioning factor is a promising perspective in the field. This progression aims to design proteins that precisely interact with designated protein or domains. This research phase is composed of two papers and focuses on encoding interacting sequences and decoding optimized protein designs for these interactions.

The initial paper sets the stage by presenting a novel methodology for generating sequences of protein domains designed to interact with specific partner domains. This approach treats the generation problem as one of domain-to-domain translation, utilizing unsupervised learning to map the relationship between interacting protein domains. The project expands the MSA-based, family-specific models, by enabling family-specific models informed by the rest of the protein, informed by the context in which a specific domain is inserted. By focusing on the translation between given interactor domains to new ones, this paper highlights the model's ability to understand and predict the complex patterns of protein interactions, laying the groundwork for subsequent research in conditional protein design.

Building on the concepts introduced in the first paper, the second study, TULIP, applies and extends the unsupervised learning framework to the specific context of T-cell receptor (TCR) and epitope interactions. This paper addresses the challenges of data scarcity and bias by leveraging the transformer architecture, demonstrating the model's effectiveness in predicting bindings between TCRs and epitopes. TULIP's success in navigating the complexities of the immune response exemplifies the potential of unsupervised generative models to generalize across different types of protein interactions.

4.1 Advancing Protein Design through Domain-to-Domain Translation

The field of computational biology faces the intricate challenge of designing proteins for specific functionalities within the complex framework of multi-domain proteins and protein-protein interactions (PPIs). Generative models, heavily inspired by advancements in natural language processing (NLP), have been used for generating amino acid sequences [40, 67]. However, these models frequently overlook the nuances of functional specificity and contextual interactions crucial for effective protein design. This study introduces a domain-to-domain translation approach to address these limitations, aiming to generate protein sequences that are specifically designed to interact with designated partner domains. This method not only marks a shift towards incorporating context in protein design but also demonstrates its applicability to PPIs, as shown in the generation of binding partners in the Histidine Kinase Response Regulator interaction.

This approach underscores the importance of accounting for the complex interplay between protein domains within the same molecule, which is essential for the functional complexity and specificity of cellular mechanisms. Utilizing data from natural multi-domain proteins, the research redefines the protein design process as a translation task, focusing on the creation of sequences that are both structurally and functionally aligned with their interaction targets. This strategy diverges from conventional practices by emphasizing the generation of interaction-specific sequences, establishing a more nuanced and context-aware framework for computational protein design.

4.1.1 Context-Aware Modeling in Protein Sequence Design

Moving beyond the conventional modeling paradigms, our approach adopts a context-aware perspective, acknowledging that proteins exist within specific organisms and are subject to unique evolutionary pressures. This recognition of the nuanced fitness landscape necessitates a modeling strategy that can incorporate elements of the protein's context into the design process.

Interacting protein families serve as a significant component of this context, influencing the functional and structural attributes of protein domains. Our method maintains the strengths of highly specialized MSA-based, family-specific models while integrating additional contextual information to inform the design with interacting domains. This balance enables the creation of domain sequences that are not only good representing of their family but also get specifically fit to be inserted into their specific biological context.

Moreover, this study has explored the task of matching pairs of interacting domains. It consists of predicting which sequence in an MSA is binding with another sequence from a second MSA. The goal was to do it in a completely unsupervised manner, leveraging only conditional likelihood. Indeed intuitively, we understand

that a specific sequence will likely have a higher probability when conditioned with the right interacting partner. The ability to match pairs efficiently without supervision, therefore without having to generate negative examples of non-interacting partners will be the starting point of the next work.

4.1.2 Innovations and Future Directions

Employing Transformers for this domain translation task allows us to harness the powerful sequence-to-sequence prediction capabilities demonstrated in language translation. This architectural choice facilitates direct application to the translation between protein domains, showing promise over traditional shallow autoregressive models. Our methodology stands at the crossroads of generative modeling for domain-specific applications and the broader potential of Transformer architectures to capture complex data patterns.

One limitation of the current approach is its focus on single-domain contexts when predicting the sequence of an interacting domain. Future research could expand this context to include multiple domains or other relevant biological information, enriching the model's predictive capacity. Future work could investigate the potential of applying transfer learning from extensive domain-domain interaction datasets to specific PPI tasks. Having demonstrated effectiveness in PPI through the HK-RR example, it would be intriguing to assess if pretraining on a broad set of domain-domain interactions enhances performance when subsequently finetuned to a particular PPI task.

4.1.3 Domain-to-Domain Translation Paper

Bioinformatics, 2023, **39(7)**, btad401
https://doi.org/10.1093/bioinformatics/btad401
Advance access publication 3 July 2023
Original Paper

OXFORD

Sequence analysis

Generating interacting protein sequences using domain-to-domain translation

Barthelemy Meynard-Piganeau[1,2,*], Caterina Fabbri[2], Martin Weigt [1], Andrea Pagnani [3,4], Christoph Feinauer [2]

[1]Computational and Quantitative Biology, LCQB UMR 7238, Institut de Biologie Paris Seine, CNRS, Sorbonne Université, Paris 75005, France
[2]Department of Computing Sciences, Bocconi University, Milan 20100, Italy
[3]Politecnico di Torino, Duca degli Abruzzi, 24, Turin 10129, Italy
[4]Italian Institute for Genomic Medicine, Strada Provinciale, 142, Candiolo 10060, Italy

*Corresponding author. Computational and Quantitative Biology, LCQB UMR 7238, Institut de Biologie Paris Seine, CNRS, Sorbonne Université, Paris 75005, France. E-mail: barthelemy.meynard@polytechnique.edu (B.M.-P.)

Associate Editor: Lenore Cowen

Abstract

Motivation: Being able to artificially design novel proteins of desired function is pivotal in many biological and biomedical applications. Generative statistical modeling has recently emerged as a new paradigm for designing amino acid sequences, including in particular models and embedding methods borrowed from natural language processing (NLP). However, most approaches target single proteins or protein domains, and do not take into account any functional specificity or interaction with the context. To extend beyond current computational strategies, we develop a method for generating protein domain sequences intended to interact with another protein domain. Using data from natural multidomain proteins, we cast the problem as a translation problem from a given interactor domain to the new domain to be generated, i.e. we generate artificial partner sequences conditional on an input sequence. We also show in an example that the same procedure can be applied to interactions between distinct proteins.

Results: Evaluating our model's quality using diverse metrics, in part related to distinct biological questions, we show that our method outperforms state-of-the-art shallow autoregressive strategies. We also explore the possibility of fine-tuning pretrained large language models for the same task and of using Alphafold 2 for assessing the quality of sampled sequences.

Availability and implementation: Data and code on https://github.com/barthelemymp/Domain2DomainProteinTranslation.

1 Introduction

Generating novel protein sequences with desired properties is one of the key challenges of computational biology. It is likely that machine learning methods will play an important role in this task, being already used for the generation of new enzymes, biological sensors, and drug molecules (Wu *et al.* 2021). A promising approach is to leverage deep generative models, which use neural networks for learning probability distributions from known, naturally occurring protein sequences (Alley *et al.* 2019, Madani *et al.* 2020, Hawkins-Hooker *et al.* 2021, Shin *et al.* 2021, Repecka *et al.* 2021). Apart from other uses, like the prediction of mutational effects (Riesselman *et al.* 2018), these models can be used for protein design by selecting high-probability sequences (possibly under constraints) from the learned distribution.

Naturally occurring protein sequences are often comprised of several domains, and domains can be classified into different families (Alberts 2008). Models that work on the domain level usually use as training data a single multiple sequence alignment (MSA) (Durbin *et al.* 1998), containing sequences from the same domain family after aligning them, and make the assumption that each sequence is constrained by the same fitness landscape. This modeling paradigm neglects the dependence of the sequence constraints on the specific context corresponding to each organism, including other proteins interacting with the sequence or other domains on the same protein. Together with the fact that most of the crystallographic structures deposited in the PDB database (Burley *et al.* 2017) are resolved only at the single domain level (Zhou *et al.* 2022), this poses interesting questions about the limitations of current approaches, e.g. when predicting the relative orientation of multidomain proteins (Wu *et al.* 2021). Another field where this issue arises is immunology, where monoclonal antibody experiments are typically performed on mouse models and only later tested in humans. This is related to the so-called humanization problem, i.e. how to graft a promising variable receptor region (CDR) from a murine to a human context (Clavero-Álvarez *et al.* 2018). For protein design, this approach may be especially relevant. When redesigning a protein in order to increase its fitness, one usually only has to redesign a specific active domain inside the protein (Cheng *et al.* 2014, Reimer *et al.* 2019, Marchand *et al.* 2022). Being able to condition this process on the context (like, e.g. interacting domain inside the protein, or interacting domain of another protein) could potentially improve the precision of the design.

Known families of interacting domains can be organized in a paired MSA (pMSA), where the aligned interaction partners

Received: October 27, 2022. **Revised:** June 1, 2023. **Editorial Decision:** June 10, 2023. **Accepted:** June 30, 2023

are concatenated (Muscat *et al.* 2020). Given the evolutionary pressure for maintaining functional interactions between proteins, amino acid substitutions at interaction surfaces are not independent between the interaction partners. The interacting sequence therefore can be used as additional information when generating a novel sequence. The current work addresses the task of generating domain sequences given an interacting domain sequence. Given that this task is similar to translation tasks in natural language processing (NLP), we explore the use of Transformers in this context. While there is some recent work using Transformers for translating between protein sequences (Wu *et al.* 2020) for specific applications, there is, to the best of our knowledge, no systematic exploration of this idea on the level of protein domain families on a diverse dataset. We explore different architectural choices, and regularization schemes and compare our results with a recently published shallow autoregressive method (Trinquier *et al.* 2021), which we use as a baseline. We also compare on a smaller scale to fine-tuned large protein language models, using Rita (Hesslow *et al.* 2022), and explore how structural predictions from Alphafold 2 correlate with our results.

The general idea of this work is summarized in Fig. 1. Consider a protein with at least two interacting domains, where interaction is defined as having a pair of amino acids at a distance of less than 8 angstrom. We then search a database of proteins for other sequences where these domains co-occur in the same protein and assemble the pMSA and use it for training the Transformer to translate from one domain to the other. The decoder being a causal language model, we can efficiently calculate the probability of a target sequence given the input sequence. This probability enables us to evaluate the compatibility of domains, which can be used for matching a domain to an interacting partner among several possible partners. The model is generative in that it can be used for generating a novel target sequence given the input sequence. Given a context, we can generate a new "translation" or target sequence and evaluate the new *de novo* proteins. In this setting, we highlight that one model per pair of domains is trained. We intend this article to fit into the line of work of domain-specific models, like in Potts Models, Variational Autoencoders (VAEs), and Restricted Boltzmann Machines (RBMs) (Tubiana *et al.* 2019, Russ *et al.* 2020, Hawkins-Hooker *et al.* 2021). We intend to provide a method for the task of redesigning a specific domain/protein, e.g. to increase its specific fitness for a specific task. We also explore the possibility of training one large Transformer for all the pairs, without observing any clear transfer learning advantage cf. Supplementary Appendix Section B.4.

2 Related literature

Generative modeling for protein design has a wide range of applications and a considerable number of different models have been proposed in the literature, recently especially

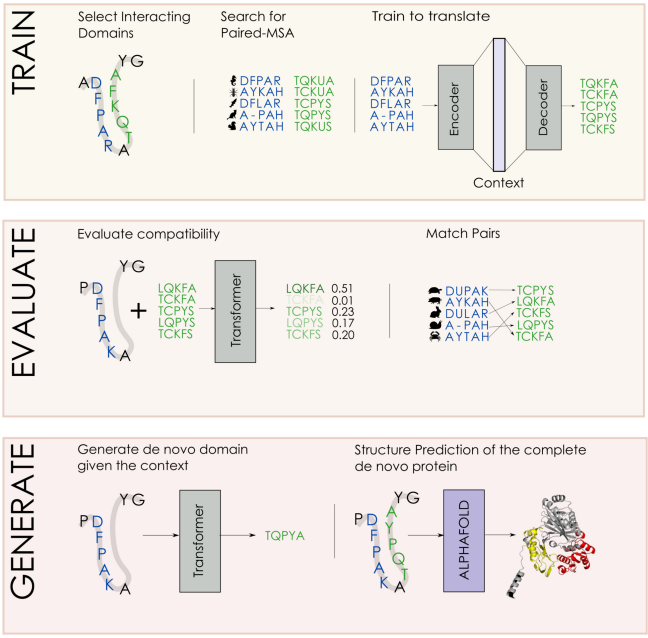


Figure 1. Summary of the work presented in this article. In the first box (Train), we extract interacting domains from known structures. We then build a pMSA based on homologous sequences of these domains and train the Transformer to translate between them. In the second box (Evaluate), we use the probabilities of the trained Transformers to match the source and target domains and assess the resulting accuracy. In the third box (Generate), we sample novel target domains and use them for replacing the original target domain. We then use Alphafold to predict the structure of the modified sequence and analyze the difference to the original structure.

deep neural network models (Wu *et al.* 2021). These include autoregressive models based on convolutional architectures (Shin *et al.* 2021), generative adversarial networks (Repecka *et al.* 2021), variational autoencoders (Hawkins-Hooker *et al.* 2021), LSTM-based architectures (Alley *et al.* 2019), and self-attention-based architectures (Madani *et al.* 2020). The latter work allows for sequence generation conditioned on tags corresponding to molecular function or taxonomic information. Similar to results in NLP, scaling protein language models to very large sizes seems promising for protein sequences (Hesslow *et al.* 2022).

Transformer-based architectures (Vaswani *et al.* 2017), which we use in the present work for sequence-to-sequence prediction, have also been used, e.g. for creating generic embeddings trained on almost all known protein sequences (Rives *et al.* 2021), the prediction of mutational effects (Meier *et al.* 2021), protein interaction prediction and protein family classification (Nambiar *et al.* 2020), MSA-based language modeling (Rao *et al.* 2021), protein contact prediction (Zhang *et al.* 2021), inverse folding (Hsu *et al.* 2022, McPartlon *et al.* 2022), and have been at the core of recent breakthroughs in protein structure prediction (Jumper *et al.* 2021).

Recently, specific tasks have been cast as sequence-to-sequence translation problems using Transformers, similar to our approach. This includes, e.g. the generation of drug molecules given a protein sequence the molecule should interact with (Grechishnikova 2021) and the generation of short signal-peptides guiding the secretion of industrial enzymes, given the amino acid sequence of the enzymes (Wu *et al.* 2020).

Finally, non-neural network models borrowed from statistical mechanics have been extensively used in the context of sequence generation, e.g. generalized Potts models, a particular form of Markov Random Field (Figliuzzi *et al.* 2018). This type of model can be used for generating sequences using MCMC strategies, albeit with a significant computational cost. Relevant approximation strategies are, e.g. the recently introduced autoregressive (shallow) variants (Trinquier *et al.* 2021), which show a similar performance to Potts models but are computationally more efficient.

3 Data and methods

3.1 Dataset

Our data consist of 27 pMSAs containing domain sequence pairs that are part of the same multidomain proteins, taken from Muscat *et al.* (2020). The dataset contains only domain pairs which form a structural contact in at least one resolved PDB structure, making it likely that the two domains coevolve in order to maintain compatibility. We also extended our work to protein–protein interaction (PPI) by analyzing the dataset of histidine kinases and response regulators (HK-RR), which form the core of bacterial two-component signal transduction systems. In the case of PPI, we consider one protein as the context of the other, even if being on different proteins. HK-RR are a good case for our framework, since there are large and well-studied pMSA available (Anishchenko *et al.* 2017). Each dataset is comprised of M rows corresponding to M sequence pairs, where M depends on the dataset and ranges from a few hundred to more than 15 000, see Supplementary Appendix Section A for a summary of the datasets used. The sequences are already aligned using standard bioinformatics tools (Finn *et al.* 2011), which means that sequences belonging to the same domain family have the same length. Each row l in a dataset represents a pair of domain sequences B^l and A^l, which are part of the same protein (or form an interacting pair of proteins). Every sequence consists of symbols denoting either 1 of 20 amino acids or an alignment gap, making the total size of the vocabulary equal to 21.

The first sequence $B^l = (b^l_1, \ldots, b^l_{N_{in}})$ is called the source or input sequence and we denote its length by N_{in}. It is used as an input to predict the second sequence, called the target or output sequence, $A^l = (a^l_1, \ldots, a^l_{N_{out}})$, which is of length N_{out}. All source sequences $\{B^l\}_{l=1}^M$ in the pMSA are members of the same domain family, and all target sequences $\{A^l\}_{l=1}^M$ in the pMSA are members of the same domain family. Each dataset was randomly split into a training set (70%) and a validation set (15%). The last 15% were kept as a testing set in order to be able to optimize hyperparameters for every domain, but we did not use it in the experiments shown in this work. Since sequences in an MSA are related to each other due to phylogeny, the validation set might contain sequences that are nearly identical to some sequences in the training set. We therefore further divided the validation set into two parts, one close to the training set and one far from it. This allows us to control for the effects of phylogeny on the performance metrics. This second splitting was made based on the median of the Hamming distance from the training set. The details for this subpartition are found in Supplementary Appendix Section A.

3.2 Performance metrics

3.2.1 Log-likelihood and perplexity

An interesting property of autoregressive models, such as the Transformer or arDCA, is that they define a tractable probability distribution over the space of sequences. Contrary to, e.g. Potts Models and other energy-based models, we do not have to evaluate a global normalizing constant over the complete space of possible sequences. We can therefore calculate the log-likelihood of a sequence A given B as

$$\log P(A|B) = \sum_{i=1}^{N_{out}} \log\left(P(a_i|B, a_1, \ldots, a_{i-1})\right). \tag{1}$$

This is related to the cross-entropy, which we use as a loss during training,

$$\mathcal{L}(A, B) = -\frac{\log P(A|B)}{N_{out}}, \tag{2}$$

which we average over batches during training.

For assessing one aspect of the quality of our models, we use the closely related perplexity $\mathcal{PP}(A, B)$, which is a common quality metric for protein language models (Armenteros *et al.* 2020), and can be defined as

$$\mathcal{PP}(A, B) = \left(\prod_{i=1}^{N_{out}} P(a_i|B, a_1, \ldots, a_{i-1})\right)^{-1/N_{out}}. \tag{3}$$

Below we show averages of the perplexity over the training and validation sets and use the notation $\mathcal{PP}^{train}$ and $\mathcal{PP}^{val}$ for these.

3.2.2 Accuracy

While we use the perplexity as one metric for the quality of our model, it is not always easy to interpret: A high perplexity can result from a single wrong prediction with a high level of confidence. We therefore also use the accuracy $\mathcal{A}(A,B)$ for assessing our models. This measure takes the same input as the cross-entropy (the conditional probability for every position) and counts the fraction of times where the true amino acid is the one with the highest probability, leading to

$$\mathcal{A}(A,B) = \frac{1}{N_{out}} \sum_{i=1}^{N_{out}} I\Big(a_i = \underset{\hat{a}\in\mathcal{V}}{argmax}[P(\hat{a}|B,a_1,\ldots,a_{i-1})]\Big), \tag{4}$$

where $\mathcal{V}$ is the alphabet of symbols and I is an indicator function that is 1 if its argument is true, and 0 else. We define $\mathcal{A}^{train}$ and $\mathcal{A}^{val}$ as the average of the accuracy on the training and validation set.

3.2.3 Matching specificity

We expect the interaction between two domains to affect the probability distribution of the target sequence only marginally, with much of the variability in the distribution being explainable by constraints internal to the target sequence. As a consequence, a good performance in the quality measures defined above might be due to the decoder being a good language model of the target protein, possibly ignoring the input sequence altogether. We therefore also evaluate the specificity of the predicted target sequence given the source sequence.

Specificity is also related to the task of matching pairs of protein sequences, which is an active domain of research in bioinformatics (Bitbol *et al.* 2016, Gueudré *et al.* 2016, Szurmant and Weigt 2018). We implement this task by separating the source and target sequences in the validation pMSA, resulting in two separate MSAs with the same number of rows, one containing the source sequences and one the target sequences. We then shuffle the rows in the target MSA randomly and attempt to use our models to find the permutation of the target sequences that matches the original order. In order to create a matching based on a model, we calculate the log-likelihood of every combination of source and target sequences in the shuffled validation set and create a matching between source and target sequences based on the Hungarian algorithm (Kuhn 1955).

We then use the fraction of correctly matched pairs as an additional metric for the performance of our model, formally defining it as

$$\mathcal{M}^{val} = \frac{\#\text{ of correctly matched pairs in validation set}}{M_{val}}, \tag{5}$$

where M_{val} is the size of the validation set. Note that the difficulty of this task increases with the size of the validation set, since the expected fraction of correctly matched pairs using a random matching is $1/M_{val}$.

3.3 Transformers and baselines

We mainly used two Transformer models with different sizes, calling one the *shallow* and one the *large* model. The shallow Transformer consists of two layers with a single attention head, has an embedding dimension of $d_{model} = 55$, and a forward dimension of $d_{ff} = 2048$. The large Transformer consists of three layers and has an embedding dimension of $d_{model} = 105$ with the same forward dimension and number of heads as the shallow transformer. Further details on their architectures can be found in Supplementary Appendix Section B.1. Both models are relatively small compared with Transformers trained on large protein sequence databases (cf. e.g. Hesslow *et al.* 2022, Lin *et al.* 2022). This can be explained in two ways. Firstly, when looking at one pair, the task is simpler as we only need to model a small fraction of the protein space where sequences can be aligned. Secondly, the smaller the number of training points gives rise to a complex overfitting problem that we analyze in Section 3.4 and Supplementary Appendix Section C.1. We compare their performance to the recently introduced shallow autoregressive model called arDCA (Trinquier *et al.* 2021) and a fine-tuned version of Rita L (Hesslow *et al.* 2022). While details on these methods can be found in Supplementary Appendix Section B.2, we note here that Rita was pretrained on a large corpus of unaligned, full-length sequences, which is a different setting from the pMSAs that we use for the Transformers. We, therefore, evaluated Rita only on unaligned, full-length sequences. For arDCA, which we train from scratch on pMSAs, there is no such mismatch and we can use it on the same pMSAs as the Transformers. For the datasets used in this work, the training time of the Transformer models ranges from less than an hour to about 1.5 days for the large Transformer on the largest dataset. The training was done using a single Nvidia V100 GPU. When using entropic regularization, which we will introduce in a later section, the training time increases significantly. In addition, we also trained a larger Transformer trained on nearly all the pairs. This Transformer has five heads, four layers, and a $d_{model} = 205$. The goal was to understand whether training on the joined dataset would enable transfer learning, or if, by mixing sequences from different families and alignments, it would make the task harder for the model. To enable a fair comparison we also replaced the <SOS> token with a token indicating the domain pair it was modeling. We need to give this hint to help the model know which family it has to generate. We held out three pairs of domains in order to check whether this joined Transformer was showing some transfer learning between families. We observed a loss of performance of approximately 3% in accuracy, 7% in matching, and 0.7 in perplexity. We concluded that a specialized Transformer for each pair, smaller and trainable in a few hours, is more suitable for the protein domain redesign task of this work. Complete results of the joined Transformer can be found in Supplementary Appendix Section B.4. We provide a table with training times in Supplementary Appendix Section B.5.

3.4 Entropic regularization

When experimenting with the large Transformer, we observed strong overfitting of the perplexity, especially when trained on smaller datasets. While this could be expected, we found that the matching performance was not following the same trend: While the perplexity started to degrade at some point during training, which is indicative of overfitting, the accuracy, and the matching performance were still increasing, see Supplementary Appendix Section C.1. While the shallow Transformer is less prone to overfitting, most likely due to its limited capacity, we found it necessary to introduce regularization for the large Transformer. We experimented with dropout and weight decay with limited success. While both

schemes prevent overfitting in terms of perplexity, the matching performance and the accuracy dropped significantly. We show this effect in Supplementary Appendix Section C.1 for different training set sizes and regularization settings.

In order to find models with a good performance on perplexity, matching, and accuracy at the same time, we explored other regularization approaches.

In this section, we present an approach based on entropic regularization, where we enforce the probability of a target sequence A given a source sequence B to be similar to other sequences sampled from the model conditioned on B. This encourages the model to give similar weights to different possible interaction partners, even if there is only a single one present in the training set.

We therefore add a regularization term $\frac{1}{T}\sum_{l=1}^{T} R_{ent}(A^l, B^l)$ to the loss, where l indexes the input sequence B^l and the target sequence A^l in the batch and T is the batch size. We sample S different target sequences for B^l from the model. We denote the kth sampled sequence conditioned on B^l as $A^{l,k}$. We sample using a Gumbel-Softmax distribution (Jang *et al.* 2016), which enables back-propagation through the sampling step. For computational efficiency, we sample every amino acid in $A^{l,k}$ conditioning on the preceding amino acids of the true A^l. Then we evaluate the log-likelihoods $R^{l,k}$ of the target sequence $A^{l,k}$ given B^l and the log-likelihood of the true pair R^l,

$$\begin{aligned} R^{l,k} &= \log P(A^{l,k}|B^l) \quad \forall k = 1,\ldots,S \\ R^l &= \log P(A^l|B^l). \end{aligned} \tag{6}$$

We then use these quantities as the input for a log-softmax operation, resulting in

$$\begin{aligned} R_{ent}(A^l, B^l) = \log P(A^l|B^l) \\ -\log\left(P(A^l|B^l) + \sum_{k=1}^{S} P(A^{l,k}|B^l)\right). \end{aligned}$$

This term is multiplied by a factor $\alpha > 0$ to regulate its strength and *added* to the loss function, meaning that we aim to minimize it. This enforces similar probabilities for the true target sequence A^l and the sampled target sequences $A^{l,k}$, conditioned on B^l. A diagram summarizing the regularization approach can be found in Fig. 2. A closer look reveals that it is a form of entropic regularization, maximizing the conditional Rényi entropy of order 2, see Supplementary Appendix Section C.2.

4 Results

4.1 Performance gain from context sequence

We first tested whether the input sequences had any effect on the perplexity of the target sequence. As already mentioned before, this is not self-evident, since the Transformer decoder itself could be a good model for the target sequence distribution without taking the input into account. We, therefore, trained two shallow Transformer models, one with the normal training set and one where we randomly shuffled the pairing between input and output sequences. We then evaluated the models on the normal validation sets, without shuffling. We expect that if the model trained on the normal training set exploits the information in the inputs when predicting the output, it should show a considerably lower perplexity than the model trained on a shuffled dataset.

We show the results of these experiments in Fig. 3. As can be seen, the models trained on the normal dataset have a significantly lower perplexity than the models trained on a shuffled dataset. This corroborates and quantifies the idea that domain sequences that appear in the context of a second domain contain information that can be used for modeling the constraints on the sequence of the second domain. We note that the difference in the logarithm of the perplexity, which is equivalent to the cross entropy, can be seen as a rough estimate of the mutual information between the output and the input. When the input sequence is randomly chosen, there is no correlation between the input and the output, and the corresponding probability can be seen as the marginal probability of the output sequence. We can therefore write

$$\begin{aligned} MI &= \sum_{a,b} P(a,b) \log\left(\frac{P(b|a)}{P(b)}\right) \\ &\approx \sum_{a,b\in V} \log\Big(P(b|a)) - \log(P(b)\Big), \end{aligned} \tag{7}$$

where a and b are paired sequences. On the left-hand side of the equation, the sum is on the complete sequence space whereas on the right-hand side, the sum is only over the sequences in the validation set.

4.2 Results on performance metrics of the shallow Transformer

We next compared the shallow Transformer models to the arDCA baseline. Shallow Transformers outperform arDCA on nearly all datasets above a certain training set size in terms of perplexity, accuracy, and matching with a large margin, as

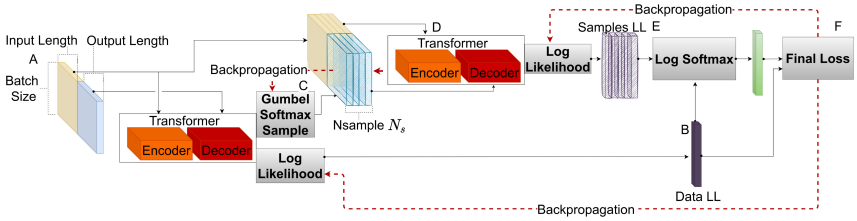


Figure 2. Diagram explaining training with entropic regularization: Panel (A) corresponds to the training batch, with yellow being the input protein sequences and blue the output protein sequences. At (B), the batch is sent to the Transformer and the log-likelihood *Data LL* is computed. At (C), N_s output protein sequences are sampled from the Transformer for every input protein sequence using the Gumbel softmax operation. At (D), we evaluate the log-likelihood of the sequences sampled at (C) and call it *Samples LL*. At (E), we measure how well the loglikelihood separates the training sequences from the sampled sequences using a logarithmic softmax, creating an additional loss term; At (F), the losses calculated at (E) and (B) are combined. Gray boxes correspond to operations that do not include learnable parameters.

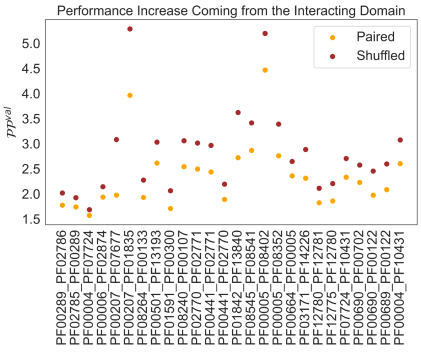


Figure 3. Performance (lower better) increases when taking domain sequence in context into account. We plot the perplexity $\mathcal{PP}^{val}$ for target sequences on the validation set, once for a shallow Transformer trained with the true pairings (Paired) and once for a shallow Transformer trained with shuffled pairs in the training set. ("Paired" is always below "Shuffled" with an average difference of 0.48).

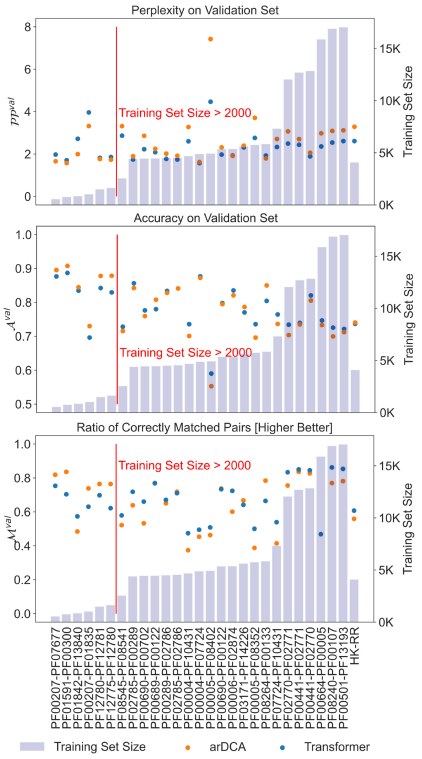


Figure 4. Perplexity $\mathcal{PP}^{val}$, accuracy $\mathcal{A}^{val}$, and the matching performance $\mathcal{M}^{val}$ for shallow Transformers and arDCA on validation set. The families pairs are ordered by training set size, followed by the PPI pair, HK-RR. Perplexity below 2000 examples: arDCA is always below Transformer with an average difference of 0.33. Perplexity above 2000 examples: Transformer is below arDCA in 91% of cases with an average difference of 0.48. Accuracy below 2000 examples: arDCA is always above Transformer with an average difference of 0.02. Accuracy above 2000 examples: Transformer is below arDCA in 81% of cases with an average difference of 0.01. Matching fraction below 2000 examples: arDCA is above Transformer in 83% of cases with an average difference of 0.07. Matching fraction above 2000 examples: Transformer is below arDCA in 77% of cases with an average difference of 0.05.

can be seen in Fig. 4. We note here that the Transformer models, both shallow and large, have *fewer* parameters than arDCA for every family size we tried: The number of parameters in the Transformer models is independent of the length of the input and target sequences, while the number of parameters in the arDCA models scales quadratically with the concatenated input length.

The best performance is achieved for the families with the largest training sets, indicating that the performance of the Transformer might further increase with increasing training set size. Comparing the fraction of correctly matched paired between pairs is not straight forward. The matching task gets harder with the size of the validation set. For each input sequence, there is only one correct partner, which has to be identified in between all other proteins in the validation set. We repeated the calculation of the matching performance on subsampled versions of the validation set in order to obtain a better understanding of the matching performance of the model, see Supplementary Appendix Section F.

We also considered the possibility of using a large language model trained on protein sequences for our task. To this end, we tested and fine-tuned Rita L (Hesslow *et al.* 2022), a 680-M parameters model trained for predicting the next amino acid in a sequence.

Given that Rita is trained on full-length unaligned sequences, we used RITA also on full-length unaligned sequences, comparing the metrics only on match positions as predicted by the Pfam HMM of the corresponding domain family (excluding gaps and inserts).

According to our metrics, a large language model like RITA seems to underperform our family-specific Transformer by a large margin see Fig. 5. We, therefore, fine-tuned RITA for each of the domain–domain pairs. We should also note that RITA is only able to model single, full-length, proteins, meaning that it cannot be applied to the PPI task of HK-RR.

The details of the fine-tuning can be found in Supplementary Appendix Section B.3. The results are comparable with the domain-to-domain Transformer model, see Fig. 6, with the Transformer having a slightly higher accuracy. We note that Rita models are trained on Uniref100 and we suspect that most of the sequences in our validation set are in the training set of Rita, so this comparison is likely biased in favor of Rita.

4.3 Performance of the entropic regularization

We performed a set of experiments on the 27 datasets in order to see if this type of regularization improves the performance. We retrained the large Transformer with and without the entropic regularization. We used $S = 5$ and $\alpha = 0.7$ for the experiments. The results can be seen in Fig. 7, where we plot the performance of the shallow Transformer against the performance of the large Transformer for different regularization schemes and arDCA. The details of the training, models, and of performance for every family can be found in Supplementary Appendix Section C.2.1. The large

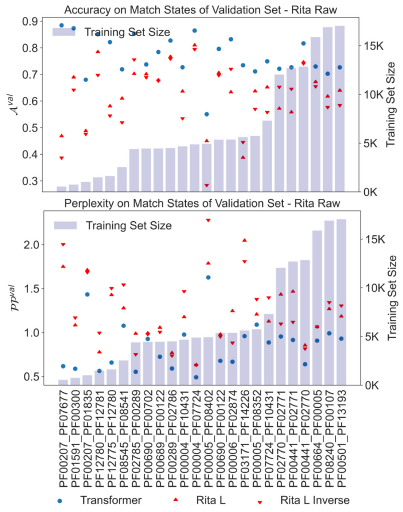


Figure 5. Perplexity $\mathcal{PP}^{val}$ and accuracy $\mathcal{A}^{val}$ for shallow Transformers and RITA L on validation set. The families are ordered by training set size. RITA L inverse refers to RITA when given the inverse sequence as input (RITA is trained on both original and inverse sequences).

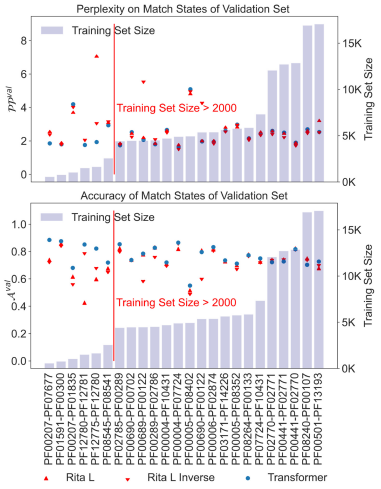


Figure 6. Perplexity $\mathcal{PP}^{val}$, accuracy $\mathcal{A}^{val}$, and the matching performance $\mathcal{M}^{val}$ for shallow Transformers and fine-tuned RITA L on the validation set. The families are ordered by training set size. RITA L inverse refers to Rita when given the inverse sequence as input (Rita is trained on both original and inverse sequences).

Transformer outperforms the shallow Transformer in terms of accuracy and matching both with and without regularization, indicating that the large Transformer extracted more useful information from the training set. However, the large

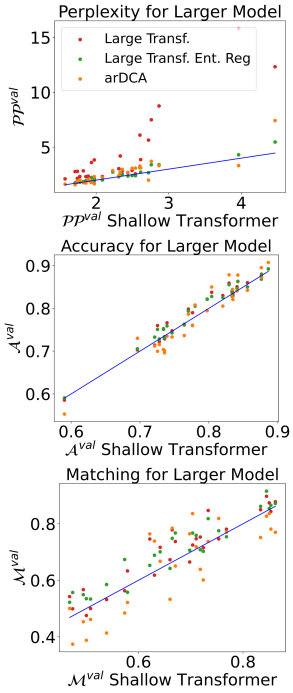


Figure 7. Comparison of the performance of the large Transformer without regularization (red), with entropic regularization (green) and arDCA (orange), and the shallow Transformer. The blue lines have a slope 1.

Transformer without regularization has a significantly higher perplexity on the validation set, indicating overfitting. Adding the entropic regularization leads to a good performance of the large Transformer in all metrics.

We also performed a systematic comparison of the entropic regularization scheme with standard weight decay, testing different weight decay values for the large Transformer. The details of the experiments and the results for every family can be found in Supplementary Appendix Section C.3.

4.4 Generalization and phylogeny

One specific characteristic of protein sequences, compared with data in NLP, is the structure of the data. The sequences in our datasets have a phylogenetic bias, visible as clusters of similar sequences in the data, that are simply explained by a close common ancestor. This bias makes a random split unsuitable since the test set will contain sequences that are very similar to some sequences in the training set. We, therefore, evaluate our model on different subsets of the test set, which are selected based on the similarity to the training set.

We show the perplexity on target sequences in the validation set in dependence of the distance from the training set in Fig. 8, where the distance of a sequence to the training set is the smallest Hamming distance from the sequence to any training sequence. Interestingly, it seems that the advantage in performance of Transformer models over arDCA is mostly

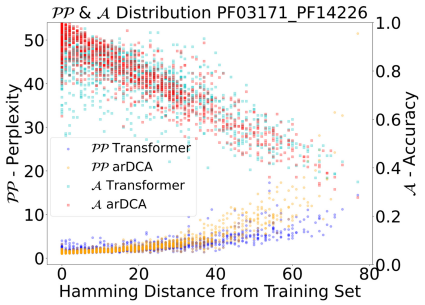


Figure 8. Perplexity (*lower* is better) and accuracy (*higher* is better) for every sequence of the validation set in dependence on the distance of the sequence from the training set. The distance of a sequence to the training set is the Hamming distance to the closest sequence in the training set.

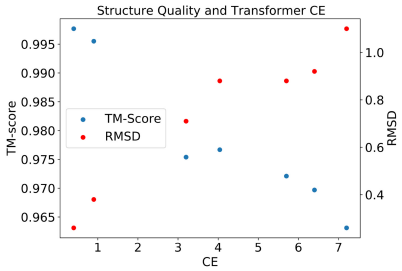


Figure 9. TM-scores and RMSD values when comparing the Alphafold-predicted structures of true sequences with Alphafold-predicted structures of sequences where single domains have been replaced with homologous natural sequences. The results are based on Q1H158, which contains domains PF00289 and PF02785, which are in contact in PDB 5ks8. Homologous PF00289 sequences are sampled from the validation set and inserted into the Q1H158 sequence, measuring the change in structural scores and in cross-entropy in the shallow Transformer model (abscissa).

Figure 10. TM-scores comparing Alphafold structural predictions based on original and modified sequences of the protein Q1H158, which contains domains PF00289 and PF02785. The sequences are modified by resampling PF00289 from the shallow Transformer and Rita.

due to sequences far away from the training set, indicating that Transformers generalize better in regions of sequence space far away from the training set. We also verified that this advantage holds for matching. To do so, we split the test set into the half closer and the half further from the training set. When matching the pairs, we look at the performance on these two subdatasets. The details of these results can be found in Supplementary Appendix Section E.1 for the shallow Transformer and at Supplementary Appendix Section C.3 for the large and regularized Transformer.

4.5 Structural information and generative properties

In this section, we show further results on the performance of the shallow Transformer.

We tested whether the target sequence distribution of trained Transformers is integrating structural information. To do so, we explored the correlation of our metrics with scores related to structure prediction when using Alphafold (Jumper *et al.* 2021). To this end, we selected the protein Q1H158 from the validation set, which contains the Pfam domains PF00289 and PF02785. We then replaced the domain PF00289 with homologous sequences from the validation set, keeping the rest of the Q1H158 sequence unmodified. The resulting sequences contain natural sequences for both domains but in a combination that does not exist in any known protein. We then used Alphafold to predict the structure of the original sequence and the modified sequence, comparing them using the TM-score and the RMSD on the two domains. We found these structural metrics to be well-correlated with the cross-entropy of the resampled PF00289 of the shallow Transformer conditioned on PF02785, see Fig. 9. We stress here that all domain sequences assessed here are natural sequences with presumably a high fitness, which makes it more likely that a higher cross-entropy for a pair is due to a decreased mutual incompatibility, reflected in the structural scores. We present results for more proteins in Supplementary Appendix Section D.1.

We next assessed the generative power of domain-to-domain Transformer models. To this end, we again used the protein Q1H158 as a test. We sampled novel PF00289 domain sequences conditioned on the PF02785 sequence found in Q1H158 using the shallow Transformer model. We then replaced the original domain sequence in Q1H158 with the sampled sequences and compared the structures predicted with Alphafold based on the original and modified sequences. For comparison, we also sampled sequences from Rita using beam search. We note that one reason for choosing Q1H158 is that the domain we want to redesign is at the end of the sequence, enabling a causal language model like RITA to sample the domain conditioned on the rest of the protein. We show the results in Fig. 10, where several sequences sampled with RITA have a significantly lower TM-score than sequences sampled from the Transformer. A closer analysis showed that some of these sequences did not contain a domain recognized by the Pfam HMM for family PF00289, indicating the fine-tuned Rita model did not always complete the sequence with the same domain as is found in the original sequence, as desired. While such alternative completions might very well correspond to a domain organization found in natural sequences, it shows that some care has to be taken when using unconditional language models for redesigning parts of a sequence, even if the model has been fine-tuned only with examples for the desired domain organization. On the other hand, the decoder of the shallow Transformer has been trained only for sampling the desired domain.

Finally, we looked at a method for unsupervised structural prediction called direct coupling analysis (DCA). We sampled for each input protein sequence of the training set eight target sequences, adding all sampled sequences together with the input sequences to a pMSA, which therefore contained natural sequences from the input domain concatenated to artificial

sequences from the target domain. We then attempted to extract structural contacts between the two domains using plmDCA (Ekeberg *et al.* 2013), a popular method for prediction contacts. While the performance in contact prediction is worse than when using the natural target sequences directly, see Supplementary Appendix Section D.2, there is a strong signal with several correctly predicted contacts among the highest scoring residue pairs.

5 Discussion

In this work, we explored the use of Transformers for generating protein domain sequences while taking into account other domain sequences that are part of the same multidomain protein. We cast the problem as a translation task, which allowed us to directly use Transformers developed for translation between natural languages. We showed that this architecture is capable of outperforming state-of-the-art shallow autoregressive models in several metrics and explored a new regularization scheme optimized for our use case. Casting the task as a translation problem allowed us to use metrics like the matching performance for assessing the quality of the generative models.

Our work is placed at the intersection of two streams of research: There is a long history of building domain-specific generative models on aligned sequences for tasks like drug design or mutational effect prediction. More recently, however, large models based on Transformer architectures trained on all or nearly all unaligned protein sequences available have shown remarkable capabilities for capturing complex patterns in the data. Our work, on the other hand, solves a very generic sequence-to-sequence prediction task using smaller Transformer architectures, specialized for a family pair and using aligned sequences, which allows for domain-specific models. One limitation of our work is that we consider only a single domain as the context when predicting the sequence of an interacting domain, disregarding additional domains that might be present in the same protein. Conceptually, it would be interesting to enrich the context to multiple other domains or other biological information such as location or phylogeny.

An interesting question for further research is if we could observe a gain in performance due to transfer learning when training one model on a very large number of pairs. Given the successful extension to HK-RR, it would be interesting to apply this approach to other PPI problems, such as TCR-epitope binding.

Supplementary data

Supplementary data are available at *Bioinformatics* online.

Conflict of interest

None declared.

References

Alberts B. *Molecular Biology of the Cell*, 5th edn. Wiley Online Library, 2008.

Alley EC, Khimulya G, Biswas S *et al.* Unified rational protein engineering with sequence-based deep representation learning. *Nat Methods* 2019;**16**:1315–22.

Anishchenko I, Ovchinnikov S, Kamisetty H *et al.* Origins of coevolution between residues distant in protein 3d structures. *Proc Natl Acad Sci USA* 2017;**114**:9122–7.

Armenteros JJA, Johansen AR, Winther O *et al.* Language modelling for biological sequences–curated datasets and baselines. BioRxiv, 2020.

Bitbol A-F, Dwyer RS, Colwell LJ *et al.* Inferring interaction partners from protein sequences. *Proc Natl Acad Sci USA* 2016;**113**:12180–5.

Burley SK, Berman HM, Kleywegt GJ *et al.* Protein data bank (pdb): the single global macromolecular structure archive. *Protein Crystallogr* 2017;**1607**:627–41.

Cheng RR, Morcos F, Levine H *et al.* Toward rationally redesigning bacterial two-component signaling systems using coevolutionary information. *Proc Natl Acad Sci USA* 2014;**111**:E563–71.

Clavero-Álvarez A, Di Mambro T, Perez-Gaviro S *et al.* Humanization of antibodies using a statistical inference approach. *Sci Rep* 2018;**8**:1–11.

Durbin R, Eddy S, Krogh A *et al. Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

Ekeberg M, Lövkvist C, Lan Y *et al.* Improved contact prediction in proteins: using pseudolikelihoods to infer Potts models. *Phys Rev E Stat Nonlin Soft Matter Phys* 2013;**87**:012707.

Figliuzzi M, Barrat-Charlaix P, Weigt M *et al.* How pairwise coevolutionary models capture the collective residue variability in proteins? *Mol Biol Evol* 2018;**35**:1018–27.

Finn RD, Clements J, Eddy SR *et al.* Hmmer web server: interactive sequence similarity searching. *Nucleic Acids Res* 2011;**39**:W29–37.

Grechishnikova D. Transformer neural network for protein-specific de novo drug generation as a machine translation problem. *Sci Rep* 2021;**11**:1–13.

Gueudré T, Baldassi C, Zamparo M *et al.* Simultaneous identification of specifically interacting paralogs and interprotein contacts by direct coupling analysis. *Proc Natl Acad Sci USA* 2016;**113**:12186–91.

Hawkins-Hooker A, Depardieu F, Baur S *et al.* Generating functional protein variants with variational autoencoders. *PLoS Comput Biol* 2021;**17**:e1008736.

Hesslow D, Zanichelli N, Notin P *et al.* Rita: a study on scaling up generative protein sequence models. arXiv preprint arXiv:2205.05789, 2022.

Hsu C, Verkuil R, Liu J *et al.* Learning inverse folding from millions of predicted structures. bioRxiv, 2022.

Jang E *et al.* Categorical reparameterization with gumbel-softmax. arXiv preprint arXiv:1611.01144, 2016.

Jumper J, Evans R, Pritzel A *et al.* Highly accurate protein structure prediction with alphafold. *Nature* 2021;**596**:583–9.

Kuhn HW. The hungarian method for the assignment problem. *Nav Res Logist* 1955;**2**:83–97.

Lin Z, Akin H, Rao R *et al.* Evolutionary-scale prediction of atomic level protein structure with a language model. *bioRxiv* 2022;**379**(6637):2022–07.

Madani A *et al.* Progen: Language modeling for protein generation. arXiv preprint arXiv:2004.03497, 2020.

Marchand A, Van Hall-Beauvais AK, Correia BE *et al.* Computational design of novel protein–protein interactions—an overview on methodological approaches and applications. *Curr Opin Struct Biol* 2022;**74**:102370.

McPartlon M *et al.* A deep SE (3)-equivariant model for learning inverse protein folding. bioRxiv, 2022.

Meier J *et al.* Language models enable zero-shot prediction of the effects of mutations on protein function. *Adv Neural Inform Process Syst* 2021;**34**:29287–29303.

Muscat M, Croce G, Sarti E *et al.* Filterdca: interpretable supervised contact prediction using inter-domain coevolution. *PLoS Comput Biol* 2020;**16**:e1007621.

Nambiar A *et al.* Transforming the language of life: Transformer neural networks for protein prediction tasks. In *Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, 2020, 1–8.

Rao RM *et al.* MSA transformer. In *International Conference on Machine Learning*, PMLR, 2021, 8844–56.

Reimer JM, Eivaskhani M, Harb I *et al.* Structures of a dimodular nonribosomal peptide synthetase reveal conformational flexibility. *Science* 2019;**366**:eaaw4388.

Repecka D, Jauniskis V, Karpus L *et al.* Expanding functional protein sequence spaces using generative adversarial networks. *Nat Mach Intell* 2021;**3**:324–33.

Riesselman AJ, Ingraham JB, Marks DS *et al.* Deep generative models of genetic variation capture the effects of mutations. *Nat Methods* 2018;**15**:816–22.

Rives A, Meier J, Sercu T *et al.* Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proc Natl Acad Sci USA* 2021;**118**:e2016239118.

Russ WP, Figliuzzi M, Stocker C *et al.* An evolution-based model for designing chorismate mutase enzymes. *Science* 2020;**369**:440–5.

Shin J-E, Riesselman AJ, Kollasch AW *et al.* Protein design and variant prediction using autoregressive generative models. *Nat Commun* 2021;**12**:1–11.

Szurmant H, Weigt M. Inter-residue, inter-protein and inter-family coevolution: bridging the scales. *Curr Opin Struct Biol* 2018;**50**:26–32.

Trinquier J, Uguzzoni G, Pagnani A *et al.* Efficient generative modeling of protein sequences using simple autoregressive models. *Nat Commun* 2021;**12**:1–11.

Tubiana J, Cocco S, Monasson R *et al.* Learning protein constitutive motifs from sequence data. *Elife* 2019;**8**:e39397.

Vaswani A, Shazeer N, Parmar N *et al.* Attention is all you need. In: *Advances in Neural Information Processing Systems*, 2017. Red Hook, NY, USA: Curran Associates Inc., 5998–6008.

Wu Z, Yang KK, Liszka MJ *et al.* Signal peptides generated by attention-based neural networks. *ACS Synth Biol* 2020;**9**:2154–61.

Wu Z, Johnston KE, Arnold FH *et al.* Protein sequence design with deep generative models. *Curr Opin Chem Biol* 2021;**65**:18–27.

Zhang H, Ju F, Zhu J *et al.* Co-evolution transformer for protein contact prediction. *Adv Neural Inform Process Syst* 2021;**34**:14252–14263.

Zhou X, Li Y, Zhang C *et al.* Progressive assembly of multi-domain protein structures from cryo-em density maps. *Nat Comput Sci* 2022;**2**:265–75.

4.2 Integrating Unsupervised Learning in TCR-Epitope Binding Prediction

The previous work's success with unsupervised matching of interacting protein domains sets the stage for applying a similar approach to the more complex problem of TCR-epitope interactions.

The prediction of TCR-epitope binding encompasses significant challenges, including the modeling of a 4-element interaction, the limitations imposed by incomplete datasets and the biases inherent in supervised learning methods. This work introduces TULIP, a novel methodology that leverages unsupervised learning and transformer architecture to overcome these hurdles, demonstrating the potential of unsupervised approaches in a more complex biological context involving the interaction of four elements: the alpha and beta chains of the TCRs, the Major Histocompatibility Complex (MHC) and the epitope.

4.2.1 A more complex interaction, enabling the sophisticated immune response

The TCR-epitope interaction intricately combines the specificity of T-cell receptors (TCR) and the diversity of Major Histocompatibility Complex (MHC) molecules, revealing the adaptive immune system's refined mechanism for recognizing and responding to intracellular threats. TCRs, composed of alpha and beta chains, each feature constant and variable regions, with the Complementarity Determining Region 3 (CDR3) being pivotal for antigen recognition. The CDR3 regions of the TCR chains are highly variable, allowing a vast repertoire of TCRs to bind to a very diverse array of epitopes. A figure summarizing this process was given in 1.4

Epitopes are peptides derived from the proteolytic processing of proteins, including those from intracellular pathogens or abnormal, cancerous cells. These peptides are presented on the cell surface by MHC molecules, a process central to immune surveillance. MHC molecules come in various alleles, contributing to the immune system's ability to recognize a broad spectrum of epitopes.

Contrary to antibodies that primarily target extracellular pathogens, the TCR-MHC-epitope axis allows the immune system to monitor and react to intracellular proteins. This capability makes TCRs natural weapons against cancer, as they can recognize and eliminate cells expressing cancer-specific or -associated antigens presented by MHC molecules.

The complexity of TCR-epitope interactions is further amplified by the need to account for the specificity of TCR binding to the peptide-MHC complex (pMHC), considering the variations across different MHC alleles. This specificity is not merely a function of the peptide alone but also how the peptide is presented by the MHC molecule, which varies significantly across the different MHC alleles found within and across individuals. Understanding this complex interplay is crucial for developing targeted immunotherapies, particularly in cancer, where the ability to target specific, intracellular antigens can lead to significant therapeutic advances [68, 69].

4.2.2 Challenges in TCR-Epitope Interaction Prediction

The accurate prediction of TCR-epitope binding is compounded by the complexity of the interactions and the requirement to account for the specificity and variability inherent in TCRs and their cognate epitopes. Previous models have struggled with incomplete data sources. Indeed a major issue, is that sequencing the pair of chains requires the more expensive technique of single cell sequencing. This causes the lack of one of the two chains in many data instances. Moreover, the prediction is usually framed as a binary classification problem and was made through the artificial generation of negative examples.

This introduction of artificial data points can introduce biases, and the proper way to generate these negative examples has not reached consensus yet.

TULIP addresses these challenges by employing an unsupervised learning framework that can utilize incomplete datasets and can be trained without relying on negative examples, thereby avoiding the biases introduced by such data in supervised approaches. TULIP has shown to be capable of recognizing specific TCRs binding to epitopes, including those not previously seen, if close enough to seen epitopes. This model represents a different approach in the field, and will participate in the collective effort and reflection on resolving this major challenge.

4.2.3 TULIP paper

TULIP — a Transformer based Unsupervised Language model for Interacting Peptides and T-cell receptors that generalizes to unseen epitopes

Barthelemy Meynard-Piganeau,[1,2] Christoph Feinauer,[2] Martin Weigt,[1] Aleksandra M. Walczak,[3,*] and Thierry Mora[3,*]

[1] *Computational and Quantitative Biology, LCQB UMR 7238, Institut de Biologie Paris Seine, CNRS, Sorbonne Université , Paris 75005, France*
[2] *Department of Computing Sciences, Bocconi University, Milan 20100, Italy*
[3] *Laboratoire de Physique de l'Ecole Normale Supérieure, ENS, Université PSL, CNRS, Sorbonne Université, Université de Paris Cité, F-75005, Paris, France*

The accurate prediction of binding between T-cell receptors (TCR) and their cognate epitopes is key to understanding the adaptive immune response and developing immunotherapies. Current methods face two significant limitations: the shortage of comprehensive high-quality data and the bias introduced by the selection of the negative training data commonly used in the supervised learning approaches. We propose a novel method, TULIP, that addresses both limitations by leveraging incomplete data and unsupervised learning and using the transformer architecture of language models. Our model is flexible and integrates all possible data sources, regardless of their quality or completeness. We demonstrate the existence of a bias introduced by the sampling procedure used in previous supervised approaches, emphasizing the need for an unsupervised approach. TULIP recognizes the specific TCRs binding an epitope, performing well on unseen epitopes. Our model outperforms state-of-the-art models and offers a promising direction for the development of more accurate TCR epitope recognition models.

I. INTRODUCTION

T cells detect foreign invaders such as viruses, bacteria and cancer cells through their membrane-bound T-cell receptor (TCR), which recognize specific epitopes presented on the surface of infected or tumor cells. Epitopes are short (8-17 amino acid) peptide fragments presented by the major histocompatibility complex (MHC) on the surface of presenting cells, which are bound to by the TCR (Fig. 1A). The TCR is a heterodimer composed of the alpha and beta chains, which are coded by separate genes that randomly recombine during thymic development, giving rise to a large diversity of possible TCRs. Binding between the TCR and the peptide-MHC (pMHC) complex is highly specific [1, 2] and plays a key role in the activation of the adaptive immune response. Predicting pMHC-TCR binding from their amino-acid sequences is an important challenge in immunology. It has important applications to diagnostics, cancer immunotherapy, and vaccination, including the engineering of TCR against target antigens [3], or the design of optimized antigens in personalized cancer vaccines [4].

Given the difficulty to predict the structure and binding interface of pMHC-TCR pairs, predicting their binding affinity from general rules of protein interactions remains a promising but arduous approach [5, 6]. Recent experimental advances [7, 8] have allowed for the generation of an increasing amount of data linking TCR sequences to peptide-MHC (pMHC) complexes, providing a large number of binding pairs. These data are gathered in several freely available databases: VDJdb [9], IEDB [10], McPAS-TCR[11]. However, the number of possible 7-16 amino-acid peptides is very large, and the potential number of possible TCRs even larger ($> 10^{60}$ [12]), meaning that experiments may only assay a small fraction of possible pairs. This calls for machine-learning methods capable of predicting the binding properties of unobserved pairs from a limited set of training data, by learning general rules of pMHC-TCR interactions.

Several studies have attempted to predict TCR specificity from sequence using a variety of machine learning techniques (see [13] for a recent benchmark), including deep convolutional networks (NetTCR2 [14]), decision trees and random forests (SETE [15], TCREX [16]) Gaussian process classification (TCRGP [17]), distance-based methods (TCRdist3 [18]), and language models (TITAN [19], Pan-Pep [20], ERGO2 [21], STAPLER [22]), and ensemble methods of Convolutional neural networks (DLpTCR [23]) . Many approaches are inherently incapable of, or show poor performance at, predicting TCR affinity to epitopes that were not present in the training set (unseen epitopes), either by design or by lack of generalizability across epitopes [19, 22]. This fundamentally limits their applicability, in particular in the context of cancer neoantigens which are often unique to each patient.

Existing models are often trained on a subset of all available data, because of requirements on quality and consistency. Experiments rarely report all four elements of the binding complex: the peptide, the MHC, the alpha and beta chains of the TCR (Fig. 1B). Because information about pMHC specificity is shared across both chains [7, 24], many methods choose to focus on data that report both chains, leaving out the large amount of information

* Corresponding authors. These authors contributed equally.

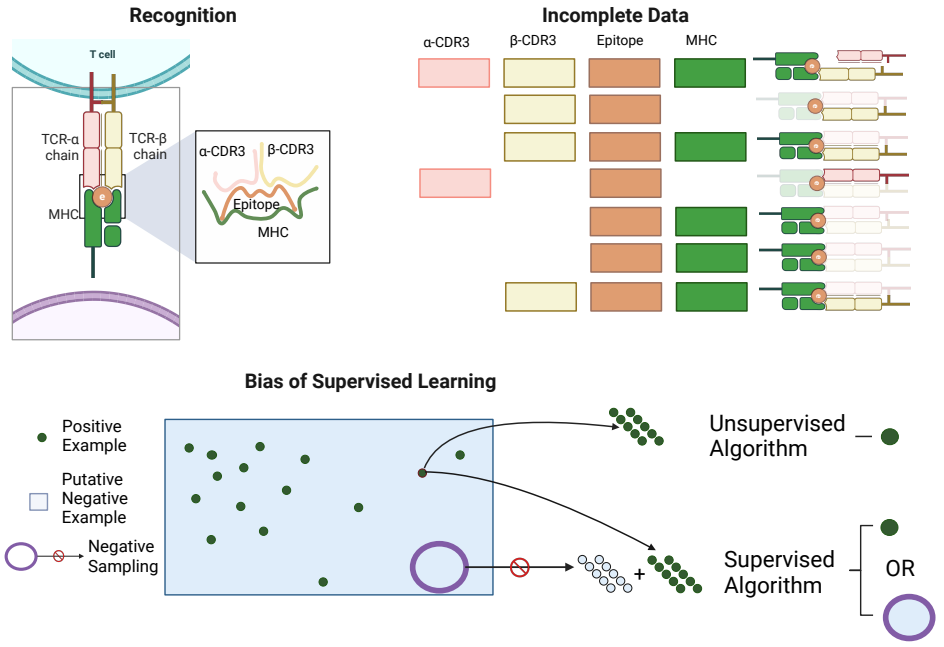


FIG. 1. A Recognition: The TCR is composed of an α chain and a β chain, each one interacting with the epitope through its CDR3. The epitope is presented by the MHC. B Incomplete Data: Schematic representation of the current state of data availability for this binding problem. C The bias of Supervised Learning: Comparison of Supervised and Unsupervised approach. The unsupervised approach is only seeing positive pairs, it will only learn to recognize the specific signal of interacting pMHC-TCR. On the contrary, the supervised approach needs to sample negative examples, and the model will also try to capture the signature of none interacting pMHC-TCR. This may introduce a bias as the model can be learning to recognize some specific signal coming from the method used to generate the negative examples. Created with BioRender.com

contained in incomplete datasets.

Another limitation of existing approaches is that they treat the binding prediction as a supervised learning task, which requires both positive and negative examples to train a binary classifier. However, the biological data at our disposal is not of this type, consisting only of positive examples. To address this issue, negative examples are often generated using random association, but these can lead to subtle biases [22]. The fraction of random pMHC-TCR functional associations is estimated to be $\approx 10^{-6}$–10^{-4} [25], meaning that non-binding pairs widely outnumber binding ones. Therefore sampling the negative space properly for training a supervised classifier is difficult. Using a supervised approach may push models to learn the biases in the negative data provided, rather than biologically meaningful patterns.

The case of having only data from one class is usually called One Class Classification (OCC), and is not new in biology [26]. Generative models are one solution to tackle this task, as we do not need any negative example to train it [27].

In this paper we present Transformer-based unsupervised language modeling for Interacting pMHC-TCR (TULIP-TCR), an encoder-decoder language model, which addresses these limitations. The model is flexible, leveraging all possible data sources regardless of their quality or completeness and including single-chain data, but also learning useful representations of the TCR and epitope space from examples where only one of them is present. The approach is unsupervised in the sense that we do not predict explicitly a binary variable that indicates binding, but a probability score trained only on interacting sequence pairs. This allows us to avoid the pitfalls associated with creating artificial samples of

non-interacting sequence pairs [27]. TULIP outperforms state-of-the-art methods on the most studied peptides for which data is abundant, and shows significant predictive power on unseen epitopes.

II. RESULTS

A. Model overview: a flexible and unsupervised architecture

Our model is inspired by techniques used in Natural Language Processing (NLP), where models are commonly trained on large text corpora [28]. In our approach, we adapt these techniques by replacing words with amino acids. The central concept behind our model is translation, which involves predicting the next token (word or word pieces in NLP, or amino acid in protein sequences) based on the previously generated tokens, as well as the source (sentence in NLP, or amino-acid sequence in proteins). During training, the model learns the patterns and dependencies that govern the relationships between tokens. By training only on positive examples, the model learns the rules that govern token ordering.

Models that predict each amino acid conditioned on the previous ones are called autoregressive. This allows us to compute the probability of each sequence as $p(a_1, ..., a_n) = \prod_{i=1}^{n} p(a_i|a_1...a_{i-1})$, and to efficiently sample new sequences, with tremendous recent success in modeling language [29] .

The training process involves maximizing the conditional likelihood of the observed sequences (positive pairs), effectively defining a probability distribution over the space of sequences. As a result, the model is trained to assign higher probabilities to positive pairs (binding pairs) compared to negative pairs (non-binding pairs) without having been trained on any negative pairs.

Our model uses the Transformer architecture, specifically the encoder-decoder variant originally developed for translation tasks [30]. In this architecture, the encoder receives a protein sequence as input (a sentence in the source language in NLP), and the decoder aims to generate an interacting protein sequence (the translated sentence in NLP) as its objective. The decoder coupled with the encoder is an autoregressive generative model, which defines the conditional probability distribution of the output given the input. The encoder-decoder approach has been successfully applied to investigate interacting amino acid sequences [31, 32].

Our problem implies interactions between 4 elements: the epitope, the MHC, and the alpha and beta chains of the TCR. We reduce the chains to their third complementarity determining regions (CDR3) known to be primarily contacting the epitope [33]. We denote the α-CDR3, β-CDR3 and epitope sequences as $\alpha = (a_1^\alpha, ..., a_{N_\alpha}^\alpha)$, $\beta = (a_1^\beta, ..., a_{N_\beta}^\beta)$ and $e = (a_1^e, ..., a_{N_e}^e)$. We extend the existing architecture and define 3 encoders and 3 decoders for the α-CDR3, β-CDR3, epitope sequences and a special embedding layer for the MHC, which we treat as a categorical variable MHC (its protein sequence is ignored, as we expect only the MHC class to be relevant). The details of this architecture are shown in Fig. 2A. Each model takes the MHC and the 3 chains as input, and try to predict each chain given the two other ones and the MHC.

They define conditional probabilities such as $p(e|\alpha, \beta, \text{MHC})$. These conditional probabilities can be used to match interacting protein sequences [31], since pairs that bind are expected to have higher probabilities than non binding ones.

This model can be used with incomplete data by determining, e.g., restricted conditional like $p(e|\alpha)$ when the beta chain and the MHC class are not available. This flexibility enables us to use every known data source available for model training and for prediction. More details about the architecture and the training can be found in the Methods section IV B.

B. Predicting new TCRs binding to known epitopes

We first evaluate the performance of TULIP for epitopes presented on the common HLA-A*02:01 allele, which is commonly used to assess such models [14]. We compare TULIP with NetTCR-2.0, a state-of-the-art supervised model [14]. We collected data for which the epitope, alpha chain, and beta chain were all present. We then created a random split of 85% for training and 15% for testing, excluding any sequences from training in which the TCR was also present in the test set. Negative examples were generated within each split by randomly pairing TCRs to a different epitope, and this process was repeated five times for each sequence. To avoid overlap between the training and test sets, negative examples were sampled within each split. We refer to this database, comprising both the training and test sets, as the Specialized Dataset (SD). NetTCR was trained on the training SD and its performance was evaluated on the testing SD for each epitope separately using the area under the curve (AUC) of the receiver operating characteristic (ROC) curve as a performance metric.

The primary aim of TULIP is to be trainable on a larger database. To ensure a fair comparison, we trained it using the following protocol: Firstly, we removed all sequences from the full database that shared the same alpha or beta chain as the test set of the SD. We trained the TULIP on this filtered dataset for 100 epochs and then fine-tuned it for an additional 40 epochs using the positive examples of the training SD. To compute the AUC, we approximated the probability of binding as $\log(p(e|\alpha, \beta, \text{MHC})) - \log(p(e|\text{MHC}))$ (see IV C), which quantifies the increase in the odds of observing e upon being recognized by the TCR.

We compared the performance of TULIP with

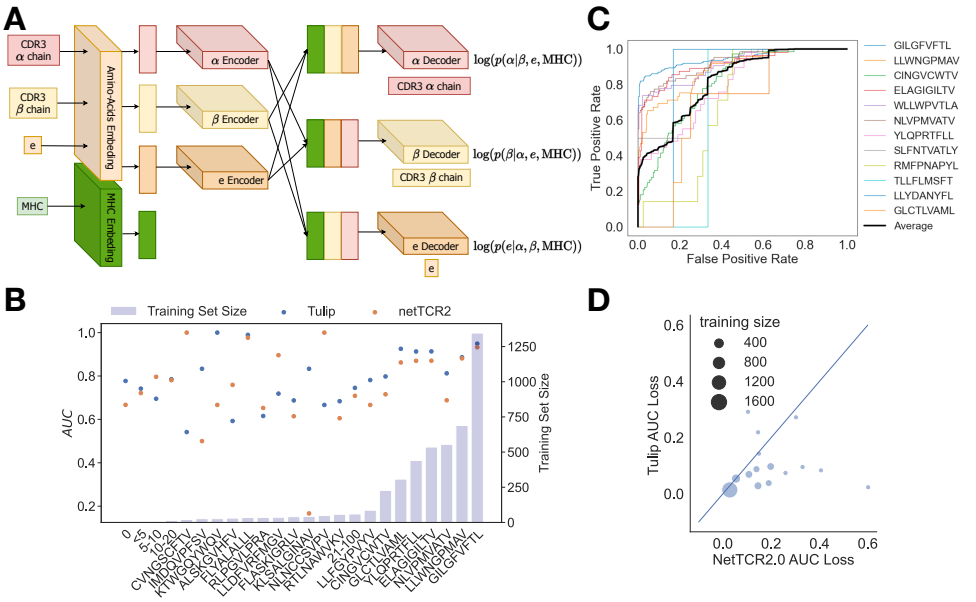


FIG. 2. A - TULIP architecture: Amino acids of each chain are embedded, then encoded by its chain-specific encoder. The MHC is also embedded. The MHC embedding and the encoded chains are then concatenated (all except the embedding of the sequence to decode) and given to the decoders. The decoders are then modeling the conditional probabilities of each chain given the MHC and the other available chains. B - Results of a finetuned TULIP on the most abundand peptides. Comparison is made with NetTCR2.0. C - ROC curve and PPV curves for the most studied peptide. The average of these ROC curves appears in red. D - We compare two ways of selecting the negative example. We compare the loss of performance of NetTCR2 and TULIP between an easy and a hard case of negative sampling. In the easy case the TCRs are randomly selected from the test set, whereas in the harder case, the TCRs are reweighted in order to have a uniform distribution over the true cognate epitope. This second choice removes the bias of having most negative examples using TCRs from the few highly over-represented peptides. Because TULIP is unsupervised it is more robust to change in the negative sampling.

NetTCR-2.0 on the testing set of SD, and computed the AUC separately for each epitope in Fig. 2B. Rare epitopes were grouped by similar training set size, and their AUC averaged. The results indicate that TULIP outperforms netTCR2.0 on almost all epitopes. For completeness, we plotted the ROC curves of TULIP in Fig. 2C. These curves reveal a very good performance on the top-ranked prediction as ROC curves start with a vertical line up to 0.5 of True Positive Rate before observing the first False Positives. This steep start is extremely interesting as it implies that the model is extremely good for the sample for which it is the most confident.

Because the AUC treats positive and negative examples symmetrically, it is particularly sensitive to the choice of negative samples, which the supervised method can exploit to artificially boost its performance [22]. To illustrate this bias, we implemented a different sampling approach for negative examples within our specialized datasets. Instead of uniformly sampling non-binding TCRs, we uniformly sampled another epitope and then selected one of its associated TCRs. This alternative sampling procedure aims to counteract the bias introduced by the over-representation of TCRs from the most commonly observed epitope in the negative sets, which leads supervised methods to learn the features of TCRs binding to that epitope, instead of learning the features of the positive TCRs. Fig. 2D shows that performances of both TULIP and NetTCR-2.0 decrease when this alternative sampling is applied, demonstrating the importance of this bias. This alternative sampling does not affect the TULIP model itself, whose training does not involve negative examples, but it does affect its AUC which relies on negative examples. However, the bias is more pronounced for a supervised method such as NetTCR-

2.0, as evidenced by the fact that most points fall below the diagonal.

C. Generalization to unknown epitopes

A major challenge of pMHC-TCR binding models is to be able to generalize, i.e. to make binding predictions on epitopes that were not used in the training set (unseen epitopes). This ability varies a lot depending on the considered epitope, notably as a function of how similar it is to other epitopes used during training, making comparisons between methods and different contexts difficult. Here, we propose a systematic approach for assessing generalization across thousands of unseen epitopes, by stratifying them according to their distance to the training set.

We split the full database into a test set comprised of epitopes with fewer than 20 examples, and a training set composed of those with more than 20 examples. TULIP was subsequently trained on the training set for 100 epochs, following which its performance was evaluated on the testing set, yielding 1796 AUCs. The Levenshtein distance between each unseen epitope and its closest counterpart in the training set was then computed. To mitigate potential bias from deep mutational scanning (DMS) experiments, which contain large numbers of closely related sequences, we identified TCRs that were associated to similar peptides and deleted them from the training set. For each peptide in the test set, the subset of peptides with minimal distance in the training set was identified, and all TCRs associated to them were removed from the test set. All TCR sequences associated with both the peptide from the test set and any peptide in the subset of closest peptides within the training set were removed from the training set. Despite these corrections, the dataset is still very biased. The distribution of TCR per epitope is skewed with a heavy tail (SI Fig. S1), and epitope representation is mostly biased towards COVID peptides and neoantigens (SI Fig. S2).

We computed the average AUC of epitopes as a function of their distance to the training set (Fig. 3A and SI Fig. S3, and SI Fig. S4 as a function of normalized distance). TCR similarity between the training and the testing sets is shown in SI Fig. S5. Machine learning methods tend to perform better in the region closer to its training set. It is a common phenomenon in all machine learning approaches for the model's capacity to extrapolate and generalize to decrease as one moves further away from the training set. We used 3 different methods for sampling the negative examples in the model evaluation (Fig. 3B). In the Unseen Unconnected Random Association (UURA) and Unseen Connected Random Association (UCRA) methods, negative pairs are drawn by picking a random TCR and a random epitope that were not in the training set. In the UURA, which is more rigorous, the true cognate epitope of the picked TCR is also unseen, while in the UCRA it can be any epitope (seen or unseen). In the Healthy Repertoire Sampling (HRS), the TCR is chosen at random from the repertoire of healthy individuals (for which the epitopes are unknown) taken from Ref. [34]. The results obtained with the most conservative negative sampling procedure (UURA, in blue) indicate that TULIP shows good generalization for epitopes that are close to the training set. This performance decays quickly with distance, reaching 1/2 (chance level) around at an edit distance of around 4.

For comparison, we also investigated the performance of existing models, PanPep [20], Ergo2 [21], and DLpTCR [23], but re-evaluated using the more rigorous UURA negative sampling method not used in the original studies (as the training/testing split of STAPLER [22] was not available at the moment of writing, we could not compare performance with that method). For instance, the performance of PanPep on unseen epitopes, which was originally assessed using the HRS method, drops to chance level when using the more stringent UURA (Fig. 3C). By contrast, TULIP, when tested on the same dataset (and re-trained on data that excluded that test set) retains some predictability. Similar results for DLpTCR are reported in SI Fig. S6A. We also compared our findings with ERGO2, which was trained using the UCRA method for negative sampling. Conducting a test by resampling the TCRs with the more conservative UURA shows that the resulting AUC also decays to values close to chance level (SI Fig. S6B). Note that STAPLER [22] was also evaluated using UCRA, potentially inflating its performance on unseen epitopes.

These findings underscore the risks of using negative samples during training. Since most pairs are negative, identifying a non-binding pair carries very little information. Any signal captured from negative examples is likely a result of batch effects introduced by the negative sampling procedure. This justifies the choice of an unsupervised architecture for pMHC-TCR binding. Thanks to this structure, TULIP is robust in the face of changes in negative sampling approaches, since training does not use any negative samples.

D. Predicting the effect of neoantigen mutations on TCR activation

To further test our model's ability to predict binding to different epitopes, and to predict epitope mutations that may evade immune recognition, we applied it to deep mutational scans of epitopes against fixed TCRs from [35]. A deep mutational scan of the epitope binding with a fixed TCR is a systematic analysis that explores the effects of multiple genetic mutations within the epitope on its interaction with a specific T-cell receptor. The study involved 6 deep mutational scans of two epitopes (HLA-A*02:01 restricted NLVPMVATV and IMDQVPFSV also present in the training set) against three TCR targets each. For each of the 19×9 single-amino acid variant of the epi-

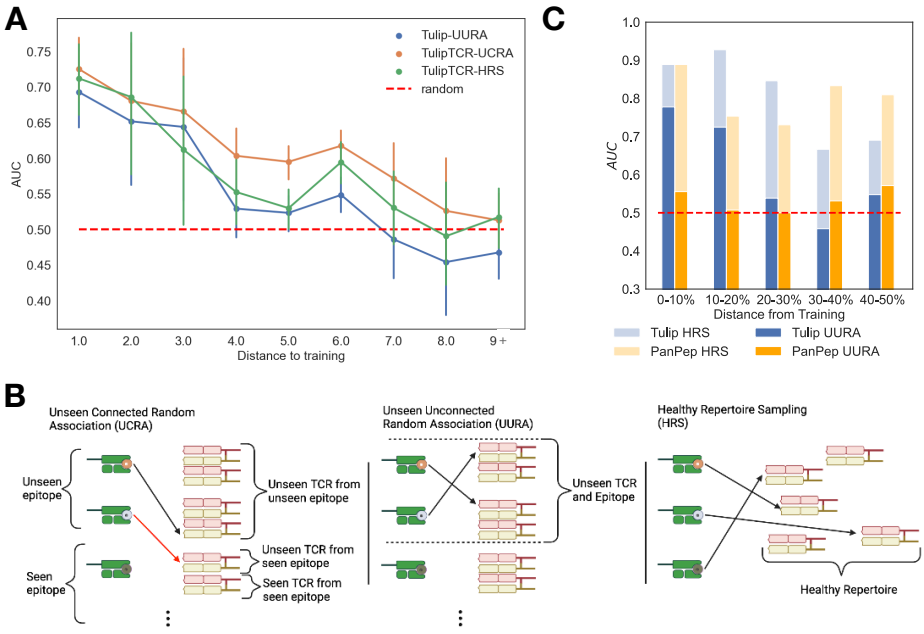
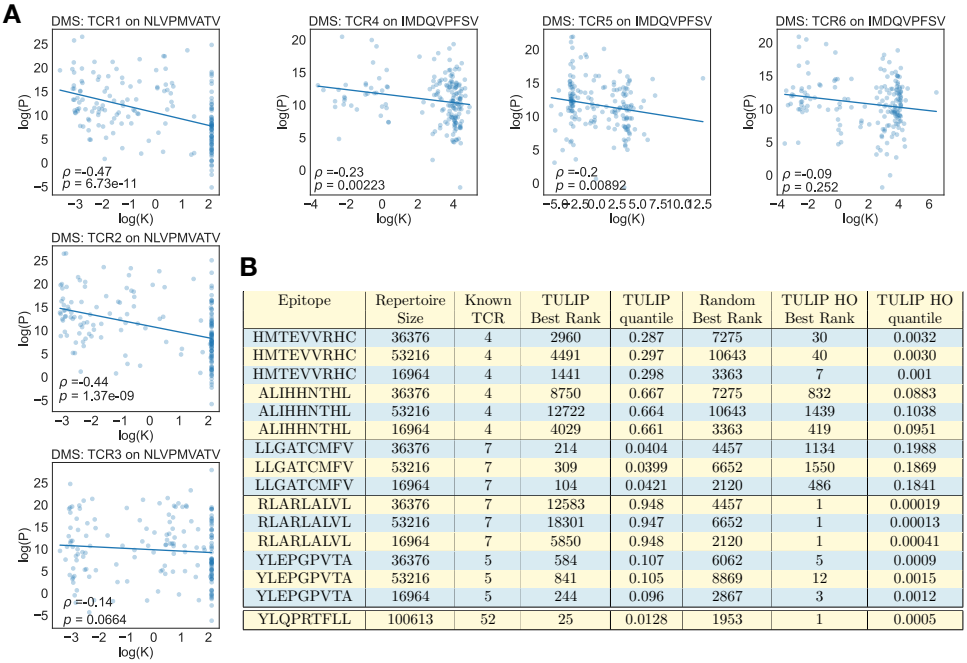


FIG. 3. A. Performance of TULIP on unseen peptides as a function of the distance to seen peptides. Up to edit distance 4, a clear signal can be seen. This analysis is done on a large set of peptides (171 at distance 1, 43 at distance 2, 44 at distance 3, 161 at distance 4, 501 at distance 5, 500 at distance 6, 103 at distance 7, 54 at distance 8, 219 at distance 9 and more). We also illustrate the role of negative sampling by showing the performance with three different Negative sampling methods. The details of these methods are explained in B. Our unsupervised methods show less variability with respect to the sampling methods compared with other supervised methods as shown in C and SI Fig.S6 B - We detail here three different methods to sample the negative of unseen epitopes. We illustrate the fact that in the original data, several TCR can be binding a single epitope, by putting two TCR in front of each epitope in the plots. Unseen Unconnected Random Association: the epitope and the TCRs are unseen and the TCRs used for the negative are binding with an unseen epitope. Unseen Connected Random Association: the epitope and the TCRs are unseen and the TCRs used for the negative can be binding to any epitope. We emphasize in red the association with a unseen connectd TCR, as it is the difference with UURA. Healthy Repertoire Sampling: Negatives are sampled from a healthy repertoire. Created with BioRender.com C - Testing the effect of change of the negative sampling on unseen peptides for PanPep and TULIP. We realize that PanPep performance does not resist changing the negative sampling process for unseen peptides contrary to TULIP. For the HRS, we reused the negative example from the original paper.

topes, the affinity to the TCR was assessed by measuring the epitope concentration at which 50% of T-cells were activated in culture (EC50). Observing binding in an experiment requires both the binding of the peptide with the MHC, and of the TCR with the pMHC. We used the joint probability of binding as a score $\log p(\text{binding}(\text{e}-\text{MHC}), \text{binding}(\text{e}-\text{TCR})|\alpha, \beta, \text{e}, \text{MHC})$ We approximate this quantity following the method in IV C by $\log p(e|\alpha, \beta) - \log p(e) + \log p(e|mhc) - \log p(e)$ as a predictor of this affinity. The comparisons between model and experiments are shown in Fig. 4A. Despite high variability, our model was able to capture the fundamental properties of binding in epitope space. To quantify performance, we measured the Spearman correlation between our score and the measured EC50. The score correlates up to 0.47 for the best TCRs. While predictability is limited, these results are encouraging considering that the model was trained on data with a large excess of TCRs relative to epitopes, and applied to data with a large excess of epitopes relative to TCRs. To assess how much of this predictability is due to peptide-MHC only (irrespective of the TCR), we compared these results with NetMHCpan [36], which is based solely on the epitope-MHC interaction, and found a lower correlation (SI Fig. S7). This highlights the importance of the TCR-epitope interaction in the experiment.

A



B

Epitope	Repertoire Size	Known TCR	TULIP Best Rank	TULIP quantile	Random Best Rank	TULIP HO Best Rank	TULIP HO quantile
HMTEVVRHC	36376	4	2960	0.287	7275	30	0.0032
HMTEVVRHC	53216	4	4491	0.297	10643	40	0.0030
HMTEVVRHC	16964	4	1441	0.298	3363	7	0.001
ALIHHNTHL	36376	4	8750	0.667	7275	832	0.0883
ALIHHNTHL	53216	4	12722	0.664	10643	1439	0.1038
ALIHHNTHL	16964	4	4029	0.661	3363	419	0.0951
LLGATCMFV	36376	7	214	0.0404	4457	1134	0.1988
LLGATCMFV	53216	7	309	0.0399	6652	1550	0.1869
LLGATCMFV	16964	7	104	0.0421	2120	486	0.1841
RLARLALVL	36376	7	12583	0.948	4457	1	0.00019
RLARLALVL	53216	7	18301	0.947	6652	1	0.00013
RLARLALVL	16964	7	5850	0.948	2120	1	0.00041
YLEPGPVTA	36376	5	584	0.107	6062	5	0.0009
YLEPGPVTA	53216	5	841	0.105	8869	12	0.0015
YLEPGPVTA	16964	5	244	0.096	2867	3	0.0012
YLQPRTFLL	100613	52	25	0.0128	1953	1	0.0005

FIG. 4. A. Effect of single epitope mutations on the TULIP score ($\log P$) predicts TCR binding (dissociation constant K in μg.ml^{-1}) measured by deep mutational scan experiments [35]. The reported ρ and p-values correspond to Spearman correlations. B. Repertoire mining for neoantigen-binding TCRs. The TCR repertoires of 3 healthy HLA*A02:01 donors from [34] were spiked with TCRs known from the literature to bind to 5 neoantigens. Sequences from the augmented repertoires were ranked by the model according to their predicted affinity to the neoantigen of interest. Reported is the rank of the best-scoring neoantigen-binding TCR. The p value corresponds to the probability of achieving that rank by chance. Two training procedures were used: one where all TCRs associated to the neoantigen of interest and related peptides were removed from the training set (TULIP), and one where only the TCR to be ranked was removed (leave-one-out; TULIP LOO). We also integrated the quantile of our prediction under the null model. This is easily interpretable as the probability that a random model would achieve equal or better performance than TULIP.

E. Repertoire mining for neoantigen recognition

We then asked whether the model could pick TCRs binding to a particular epitope from whole repertoires. We focused on TCRs binding to 6 HLA-A*02:01 restricted epitopes, including 5 cancer-associated neoantigens (Cyclin D1: LLGATCMFV [37]; p53: HMTEVVRHC [38, 39]; HER2: ALIHHNTHL [40]; TPBG: RLARLALVL[41]; and gp100: YLEPGPVTA [42], see Table S1 for the full list). In addition, we looked for TCRs specific to the SARS-CoV-2 spike protein epitope YLQPRTFLL in the CD8$^+$ repertoire of a COVID-19 infected donor at the peak of the response at day 15 [43]. YLQPRTFLL-specific TCR harbored by the same donor were identified in a separate study using a multimer-binding assay [44].

We first considered a scenario where no prior knowledge about TCRs binding the epitope was available, by removing these entries from the training set, as well as all TCR-epitope pairs whose epitope is similar to the epitope of interest (less than 4 amino acid substitutions). For the SARS-CoV-2 epitope, we also removed all TCRs associated with YLQPRTFLL as well as similar epitopes ('YLRPRTFLL' and 'YYVGYLQPRTFLL') to mitigate potential leakage effects. In the second scenario (hold out, or HO), we only removed from the training set the TCR that we want to find: one neoantigen-associated TCR at a time in the case of neoantigen, and all epitope-

specific TCR from the donor as reported by the multimer assay in the SARS-CoV-2 repertoire. In both cases, we removed redundancies of the alpha and beta chains: when several TCRs has the same alpha chain, we only retained one of them, and likewise for beta chains.

For each neoantigen, we mixed neoantigen-associated TCRs (all of them in the first scenario, and only the removed one in the LOO scenario) with 3 unrelated TCR$\alpha\beta$ repertoires of HLA*A02:01 positive donors from Ref. [34]. For the SARS-CoV-2 epitope, we simply considered the CD8$^+$ TCRβ repertoire at day 15 from [43]. We then asked TULIP to rank each TCR according to the predicted binding to the epitope of interest. The results, reported in Fig. 4B, show that TULIP in many cases narrows down the list of candidate TCR to a relatively small number, even when it was trained with no knowledge about the neoantigen-associated TCRs. When it does (TULIP LOO first index column), it can even identify the neoantigen-associated TCR within the very best ranked ones. In the case of the SARS-CoV-2 epitope, performance was excellent even when in the first scenario (no prior knowledge about the epitope), and perfect (best rank 1) in the hold-out scenario.

That analysis focuses on the top-ranking TCR for each neoantigen, emphasizing precision in detecting potent binders within the repertoire. This deliberate emphasis on the upper tiers of the score distribution provides insights into the model's discriminative power and its ability to identify TCRs with high binding affinity to specific epitopes.

III. DISCUSSION

In this study, we have presented a novel approach for TCR-epitope binding prediction that overcomes key limitations of current methods. We demonstrated the model's ability to generalize to unseen epitopes, which is a critical factor in real-world applications where the specific epitope of interest may not be known in advance. Furthermore, we addressed the recurrent bias that can arise from using negative examples generated through random pairing in previous supervised approaches. To mitigate this bias, we proposed an unsupervised learning framework that trains the model exclusively on positive examples, allowing it to focus on recognizing patterns within these interactions.

The elimination of negative examples in our approach was driven by the recognition that randomly generated negative examples can introduce biases, potentially compromising the model's predictive accuracy. By training solely on positive examples, our model avoids such biases and can more effectively capture the specific signal of interacting pMHC-TCR complexes.

One difficulty in evaluating and comparing methods is that the exact TCR-epitope binding prediction task may differ across studies and applications. For instance, looking for epitope-specific TCR within the peripheral repertoire is a different task than finding them within responding clones in lymph nodes or in tumor tissues. Likewise, identifying TCRs binding to a neoantigen but not to the wildtype is not the same as identifying the response to a specific antigen within a repertoire. Some of the biases discussed earlier arise from unclear or unrealistic definitions of the tasks. When the objective is to recognize patterns in binding complexes, the unsupervised approach emerges as the more natural choice. Supervised approaches can only demonstrate their potential in specific use cases where negative samples can be precisely defined (e.g. sorting cells that do not carry an activation marker or do not bind a tetramer, although these negative examples are typically not reported in studies). Careful consideration should also be given to the sampling of negative examples. Negative examples should be selected to be close enough to the classification boundary, making them challenging examples (referred to as Hard Negative Sampling). The combination of these constraints, including a well-defined and restricted negative subspace, the difficulty of examples, presents significant challenges for most use cases, lead us to conclude that unsupervised approaches should be preferred for most applications.

We emphasize the importance of utilizing all available data sources, regardless of their completeness or quality. The same is true for NLP approaches, which usually start by collecting and training on as much data as possible. The TCR-epitope binding prediction task often suffers from the scarcity of comprehensive data, as obtaining complete TCR sequences along with corresponding epitopes and MHC information is challenging. However, our approach is designed to be flexible, leveraging the available data and accommodating situations where only partial data is accessible. By using both alpha and beta chains when available, while being able to learn from one chain alone, our model can make the most of the data at hand and extract valuable insights.

While our proposed model shows promise, it is essential to conduct fair and rigorous model comparisons to assess its performance accurately. The field of TCR-epitope binding prediction often lacks standardized benchmark datasets and evaluation protocols (but see [13]), leading to difficulties in comparing different models. To address this challenge, future research should focus on establishing standardized benchmarks and evaluation procedures that encompass diverse datasets and evaluation metrics beyond classification.

One limitation of our approach is that the model yields only probabilities of pairs of sequences, rather than a proper binding constant, for which titration data (where the concentration of the epitope is varied) would be needed. Another limitation is that large areas of the epitope space have not been measured, and some parts are extremely hard to measure. For example, having a model able to determine the risk that a TCR binds to self-proteins, would be extremely useful for predicting the safety of T-cell therapy, but such TCRs are by construc-

tion hard to observe, and the lack of data is a major limitation for further progress in this direction.

Since our model is generative in nature, it would be interesting to experimentally test its ability to generate de novo TCR sequences for given epitopes, or for combinations of related epitopes to which it would be cross-reactive. This avenue of research could provide valuable insights into the design and discovery of TCRs with specific binding capabilities.

IV. METHODS

A. Data collection

Data acquisition in the field of immunology presents a major challenge. The intricate process of T cell receptor (TCR) binding to its respective epitope depends on four critical elements: the epitope itself, the major histocompatibility complex (MHC), and the alpha and beta chains of the TCR. While each component has been extensively studied in isolation, the number of instances where all four components are jointly available remains remarkably scarce.

In this study, we present a novel computational aiming approach at constructing a model capable of learning from incomplete data. To achieve this goal, we curated data from multiple sources, maximizing the total sample size at our disposal. Specifically, we first accessed the VDJdb database [9] in its entirety, which boasts the highest data quality among our available resources (see Table I).

We also added the IEDB database of Tcell receptors and McPAS-TCR dataset [10, 11] (see Table I).

The IEDB database is more diverse but we observed a much poorer quality of the data, and there was never used for finetuning.

This accounts for 209779 not redundant data points containing the epitope and at least one chain of the TCR.

The instances listed above consist of T-cell receptors along with their respective epitopes and major histocompatibility complex (MHC). Regrettably, the MHC or one of its two chains is frequently absent. Additionally, the diversity of epitopes is relatively low compared to that of TCRs, with each epitope possessing multiple T-cell receptors.

To supplement our data, we incorporated the training database of netMHC, which is solely composed of MHC and epitope information. Although this dataset does not directly aid in comprehending the correlation between TCR and epitope, it is advantageous in two ways. Firstly, the dataset encompasses a wide range of epitopes, which assists the model in comprehending the true diversity of potential epitopes. Secondly, in order to achieve effective transfer learning between MHC, the model must comprehend what is distinct to each MHC and what can be transferred. Therefore, the netMHC database aids in better modeling the specific role of MHC in the epitope modeling process. We gather 663, 767 peptides with their MHC (see Table I).

We gathered all this data in a single one that we will refer to as the Full Dataset (FD).

B. Model definition

Our model is an extension of the well-known Transformer model, in its encoder-decoder version. In the original version [30], the method was used for translation. During training the encoder was given a sentence in the source language and the decoder was given the translation in the target language as an objective to produce.

In our specific problem, we would like to condition our model on more than one interacting element. We, therefore, need to extend the existing architecture. We define 3 encoders, 3 decoders, and two embedding layers: an α-encoder that is specialized in encoding the α-CDR3, an α-decoder that is specialized in decoding the α-CDR3, a β-encoder that is specialized in encoding the β-CDR3, a β-decoder that is specialized in decoding the β-CDR3, an epitope-encoder that is specialized in encoding the epitope, an e-decoder that is specialized in decoding the epitope and finally an amino acid embedding and an MHC embedding, (as we decided to represent the MHCs as categorical variables). First experiments on initializing the decoders with the weights of pretrained general purpose proteins masked language models did not show any sign of improvement. TCRs α and β chains exhibit unique characteristics and patterns that are distinct from general protein sequences. The core of the loops of the CDR3 is extremely variable. On the other hand, epitopes are much smaller than usual proteins and presented inside an (MHC). All these factors imply that general rules for proteins do not transpose easily to our scenario. By utilizing dedicated encoders and decoders tailored to the specific nature of TCRs and epitopes, we can capture and encode their domain-specific features more effectively. This specificity enables the model to focus on relevant information and potential interactions specific to TCR-epitope binding.

While we refer to the original work on Transformer [30] for precise details on the attention layers and the encoder-decoder architecture, we review here the key components.

Sequences are encoded by their specific encoder and used as the input for the decoders. They are processed through alternating blocks of self-attention and linear layers.

Typical vocabulary sizes in NLP are in the order of 10^4 to 10^5, while in our case we have a vocabulary $\mathcal{V}$ is composed of the 20 amino acids and some special token (PAD for padding, EOS for End-od-Sentence, SOS for Start-Of-Sentence, UNK for the Unknown characters). The sequence embedding is composed of two parts, one for the amino acid identity and one for the position in

	VDJdb epitope with MHC	VDJdb epitope without MHC	McPAS-TCR epitope with MHC	McPAS-TCR epitope without MHC	IEDB epitope with MHC	IEDB epitope without MHC	netMHC epitope with MHC
Alpha and Beta	29251	0	5021	77	5021	77	0
Alpha alone	6750	0	1065	87	1065	87	0
Beta alone	23011	0	9898	218	145332	11	0
No TCR	0	0	0	0	451	11	663767

TABLE I. Summary of the data sources used for training.

the sequence. We learn a dictionary, mapping each of the amino-acids to a vector of dimension d_{model}. The sequence position is embedded as a vector in the same way, learning an embedding vector for every position. We also learn a specific embedding for the most common HLA types. The embedding of each sequence is taken as the sum of the amino-acids and positional embeddings. The embedded amino-acid sequences are then passed to the respective encoders, mapping them to a latent representation $z^T = (z_1, \ldots, z_n), T \in (\alpha, \beta, e)$. The encoded sequences are then concatenated with the MHC embeddings before being sent to the decoder. For each decoder, we concatenate the MHC embedding with all the encoded sequences except the one that the decoder will reproduce, as we do not want to give a decoder the sequence it is supposed to reproduce. The details of these groups can be seen in Fig2. For example, we concatenate the encoding of the α-CDR3 , the β-CDR3, and the MHC for the Epitope decoder, as the epitope decoder should be conditioned on everything but himself. The decoders are then trained to predict their respective amino-acid sequences conditioned on the encoded pieces of information.

The decoder implements an auto-regressive distribution, for example

$$P(a_i^\alpha | a_{<i}^\alpha, z^\beta, z^e, \text{mhc}), \tag{1}$$

defining the probability of the i^{th} amino acid in the $\alpha - CDR3$ sequence given the preceding amino acids $a_{<i}^\alpha$ and the hidden representation of the others elements. During training, we use the true amino acids for $a_{<i}^\alpha$. This way of predicting the next amino acid in a sequence is called Causal Language Modeling (CLM). The loss associated with this task for a single sequence is simply the cross entropy for every predicted token.

$$\begin{aligned} \text{Loss}^{\text{CLM}} = &- \sum_i^{N_\alpha} \log(P(a_i^\alpha | a_{<i}^\alpha, z^\beta, z^e, \text{MHC})) \\ &- \sum_i^{N_\beta} \log(P(a_i^\beta | a_{<i}^\beta, z^\alpha, z^e, \text{MHC})) \\ &- \sum_i^{N_e} \log(P(a_i^e | a_{<i}^e, z^\alpha, z^{beta}, \text{MHC})) \end{aligned} \tag{2}$$

We schematize the forward pass of Tulip in the following pseudo-code:

Algorithm 1 TULIP

for $C \in (\alpha, \beta, e)$ **do** ▷ In parallel
  $\text{embed}_C^{aa} \leftarrow (\texttt{embedding}(a_1^C,), ..., \texttt{embedding}(a_{N_\alpha}^C))$
  $\text{embed}_C^{pos} \leftarrow (\texttt{Posembedding}(1), ..., \texttt{Posembedding}(N_C))$
  $\text{input}_C \leftarrow \text{embed}_C^{aa} + \text{embed}_C^{pos}$
  $z^C \leftarrow \texttt{C-encoder}(\text{input}_C)$
end for
$r_\alpha \leftarrow \texttt{concat}(z^\beta, z^e, \text{embed}_{MHC})$
$r_\beta \leftarrow \texttt{concat}(z^\alpha, z^e, \text{embed}_{MHC})$
$r_e \leftarrow \texttt{concat}(z^\alpha, z^\beta, \text{embed}_{MHC})$
for $C \in (\alpha, \beta, e)$ **do** ▷ In parallel
  for $i \in (1, .., N_C)$ **do** ▷ In parallel
    $p(a_i^T | r_C, a_1^C, ...a_{i-1}^C) \leftarrow \texttt{C-decoder}(r_\alpha, a_1^T, ...a_{i-1}^T)$
  end for
end for
$p(\alpha|\beta, e, \text{MHC}) = \prod_i^{N_\alpha} p(a_i^\alpha | r_\alpha, a_1^\alpha, ...a_{i-1}^\alpha)$
$p(\beta|\alpha, e, \text{MHC}) = \prod_i^{N_\beta} p(a_i^\beta | r_\beta, a_1^\beta, ...a_{i-1}^\beta)$
$p(e|\beta, \alpha, \text{MHC}) = \prod_i^{N_e} p(a_i^e | r_\alpha, a_1^\alpha, ...a_{i-1}^\alpha)$

This approach has already been used for proteins in many works. Especially in [31] an encoder-decoder model was used to investigate interacting amino-acid sequences. The first thing to remark is that the decoder defines autoregressively a probability distribution over the generated sequence. It is generative as we can sample new examples but if we give it an existing specific sequence it will give us its probability. When coupling this to an encoder the probability distribution becomes a conditional probability distribution (conditioned on the input of the encoder). These conditional probabilities can be used for matching interacting protein sequences [31].

One interesting property of the Attention mechanism of the transformer is that it is position-blind and flexible with respect to the length of its input. This implies that it does not hard code in its weights where it is expecting to find specific elements. If a chain is missing, let's say the α-CDR3, we can only gather the MHC and the β-CDR3 before giving it to the epitope decoder. The encoded β amino acids end up in the first position of the gathered encoding. This is not a problem thanks to the position-blindness of the encoder-decoder attention. To be more precise, the missing α-CDR3 is not completely skipped but replaced by a learned vector, to inform the model that the chain is missing.

Because of the incompleteness of the data we want to learn as much as possible from every piece of data available. The decoder is in itself a language model, so it is

able to learn without or with little conditioning. In a standard encoder-decoder Transformer learning the encoder is only trained through the decoder. We want to avoid this so that the encoder learning will not be entirely dependent on another piece of data to predict. Luckily, encoders are also trainable alone (without a decoder) by doing Masked Language Modeling (MLM). During MLM training we pick 15% of the amino acid positions, we'll call this set of amino acids $\mathcal{M}$. From these ones 80 % are replaced by a mask token, and 20 % are replaced by a random amino acids. This deteriorated sequence is fed to the encoder. We learn a linear classifier on top to predict the original amino acids in $\mathcal{M}$. The logits output by this classifier are then passed by a softmax, defining for every position i a distribution over the amino acids a_i: $P_{cls}(a_i|z_{masked})$. The final MLM loss is simply the cross entropy:

$$\begin{aligned}\text{Loss}^{\text{MLM}} = &- \sum_{i\in\mathcal{M}^\alpha} \log(P_{\text{cls}}(a_i^\alpha|z^\alpha_{\text{masked}})) \\ &- \sum_{i\in\mathcal{M}^\beta} \log(P_{\text{cls}}(a_i^\beta|z^\beta_{\text{masked}})) \\ &- \sum_{i\in\mathcal{M}^e} \log(P_{\text{cls}}(a_i^e|z^e_{\text{masked}}))\end{aligned} \tag{3}$$

For example, it enables the epitope-encoder to still learn from the 600000 samples where we do not have TCRs.

In the end we combined the two losses, using a parameter λ that we always use equal to 0.5 in this paper, and sum over the sequence in the training set:

$$\text{Loss}(\theta) = \sum_{x\in\text{train}} (1-\lambda)\text{Loss}^{\text{MLM}}(x) + \lambda\text{Loss}^{\text{CLM}}(x) \tag{4}$$

where $x = (x_\alpha, x_\beta, x_e, \text{MHC})$ is our raw datapoint and θ are the parameters of the model. Details on the training of a transformer can be found in appendix.

Code and weights for the model can be found at `https://github.com/barthelemymp/TULIP-TCR/`

C. Mutual information as a proxy to the binding probability

The model presented before is autoregressive. The structure of the probabilities defined by the model is simply $P(a_i^\alpha|a^\alpha_{<i}, z^\beta, z^{epitope}, MHC)$ (resp β, epitope) and a simple multiplication over the position gives us a conditional probability on the sequences $(p(e|\alpha,\beta,\text{MHC}), \text{p}(\alpha|\text{e},\beta,\text{MHC}), \text{p}(\beta|\text{e},\alpha,\text{MHC})$. However, we should be more precise on what we want to evaluate. These conditional probabilities can be good for generating sequences, but here we first want to evaluate the probability of binding. We will show in this section how to approximate this quantity from the ones evaluated by our model. Let's introduce the random binary variable of binding or not b such that e T becomes dependant conditionally on b. The first thing we need to observe is that our TULIP model is trained only on positive, i.e., binding examples. As a first simplification let's look at the link between the binding posterior for a simple case of e being the epitope and T the alpha and beta chain of the TCR A simple bayesian approach will help us here.

$$p(b=1|e,T) = \frac{p(e|T,b=1)p(T|b=1)p(b=1)}{p(e,T)} \tag{5}$$

we can start to do some approximation here.

- All TCR sequenced in blood should have passed some positive thymic selection for epitope binding. This implies that $p(T|b=1) = p(T)$
- $p(e,T) = p(e)p(T)$ by construction as the dependence only appears when conditioning on b.

Leading to:

$$p(b=1|e,T) = \frac{p(e|T,b=1)p(b=1)}{p(e)} \tag{6}$$

noticing the $p(b=1)$ are constants of the problem, we see that the binding posterior is proportional to the pointwise mutual information (PMI) between T and e:

$$\begin{aligned}\log p(b=1|e,T) \propto \log p(e|T,b=1) - \log p(e) \\ = PMI(e;T|b=1)\end{aligned} \tag{7}$$

This quantity is the one we used to validate our models in the previous sections. Pushing further the derivation to include the role of the MHC, did not improve the results.

A similar computation can be done for the interaction between the epitope and MHC, by simply replacing T with MHC in the previous equation. This second term is used in Section. II D, where the experimental EC50 are the results of the simultaneous binding of the TCR with epitope and of the epitope with the MHC.

V. DATA AND CODE AVAILABILITY

Code is available at `https://github.com/barthelemymp/TULIP-TCR/`. The data used were collected from `https://vdjdb.cdr3.net/`, `https://www.iedb.org/` and `http://friedmanlab.weizmann.ac.il/McPAS-TCR/`.

ACKNOWLEDGMENTS

A.M.W. and T.M. were supported by grant COG 724208 from the European Research Council, and grant

ANR-19-CE45-0018 "RESP-REP" from the Agence Nationale de la Recherche. This work was granted access to the HPC resources of IDRIS under the allocation 2022-AD011013872 made by GENCI.

[1] O. Feinerman, J. Veiga, J. R. Dorfman, R. N. Germain, and G. Altan-Bonnet, Variability and robustness in t cell activation from regulated heterogeneity in protein levels, Science **321**, 1081 (2008).

[2] P. François, G. Voisinne, E. D. Siggia, G. Altan-Bonnet, and M. Vergassola, Phenotypic model for early t-cell activation displaying sensitivity, specificity, and antagonism, Proceedings of the National Academy of Sciences **110**, E888 (2013).

[3] M. Poorebrahim, N. Mohammadkhani, R. Mahmoudi, M. Gholizadeh, E. Fakhr, and A. Cid-Arregui, Tcr-like cars and tcr-cars targeting neoepitopes: An emerging potential, Cancer gene therapy **28**, 581 (2021).

[4] L. A. Rojas, Z. Sethna, K. C. Soares, C. Olcese, N. Pang, E. Patterson, J. Lihm, N. Ceglia, P. Guasp, A. Chu, et al., Personalized rna neoantigen vaccines stimulate t cells in pancreatic cancer, Nature , 1 (2023).

[5] P. Bradley, Structure-based prediction of t cell receptor: peptide-mhc interactions, eLife **12**, e82813 (2023).

[6] D. S. Shcherbinin, V. K. Karnaukhov, I. V. Zvyagin, D. M. Chudakov, and M. Shugay, Large-scale template-based structural modeling of t-cell receptors with known antigen specificity reveals complementarity features, bioRxiv , 2023 (2023).

[7] P. Dash, A. J. Fiore-Gartland, T. Hertz, G. C. Wang, S. Sharma, A. Souquette, J. C. Crawford, E. B. Clemens, T. H. Nguyen, K. Kedzierska, et al., Quantifiable predictive features define epitope-specific t cell receptor repertoires, Nature **547**, 89 (2017).

[8] J. Glanville, H. Huang, A. Nau, O. Hatton, L. E. Wagar, F. Rubelt, X. Ji, A. Han, S. M. Krams, C. Pettus, et al., Identifying specificity groups in the t cell receptor repertoire, Nature **547**, 94 (2017).

[9] D. V. Bagaev, R. M. Vroomans, J. Samir, U. Stervbo, C. Rius, G. Dolton, A. Greenshields-Watson, M. Attaf, E. S. Egorov, I. V. Zvyagin, et al., Vdjdb in 2019: database extension, new analysis infrastructure and a t-cell receptor motif compendium, Nucleic Acids Research **48**, D1057 (2020).

[10] R. Vita, S. Mahajan, J. A. Overton, S. K. Dhanda, S. Martini, J. R. Cantrell, D. K. Wheeler, A. Sette, and B. Peters, The immune epitope database (iedb): 2018 update, Nucleic acids research **47**, D339 (2019).

[11] N. Tickotsky, T. Sagiv, J. Prilusky, E. Shifrut, and N. Friedman, Mcpas-tcr: a manually curated catalogue of pathology-associated t cell receptor sequences, Bioinformatics **33**, 2924 (2017).

[12] T. Mora and A. M. Walczak, Quantifying lymphocyte receptor diversity, in Systems Immunology (CRC Press, 2018) pp. 183–198.

[13] P. Meysman, J. Barton, B. Bravi, L. Cohen-Lavi, V. Karnaukhov, E. Lilleskov, A. Montemurro, M. Nielsen, T. Mora, P. Pereira, et al., Benchmarking solutions to the t-cell receptor epitope prediction problem: Immrep22 workshop report, ImmunoInformatics **9**, 100024 (2023).

[14] A. Montemurro, V. Schuster, H. R. Povlsen, A. K. Bentzen, V. Jurtz, W. D. Chronister, A. Crinklaw, S. R. Hadrup, O. Winther, B. Peters, et al., Nettcr-2.0 enables accurate prediction of tcr-peptide binding by using paired tcrα and β sequence data, Communications biology **4**, 1060 (2021).

[15] Y. Tong, J. Wang, T. Zheng, X. Zhang, X. Xiao, X. Zhu, X. Lai, and X. Liu, Sete: Sequence-based ensemble learning approach for tcr epitope binding prediction, Computational Biology and Chemistry **87**, 107281 (2020).

[16] S. Gielis, P. Moris, N. De Neuter, W. Bittremieux, B. Ogunjimi, K. Laukens, and P. Meysman, Tcrex: a webtool for the prediction of t-cell receptor sequence epitope specificity, BioRxiv **373472** (2018).

[17] E. Jokinen, J. Huuhtanen, S. Mustjoki, M. Heinonen, and H. Lähdesmäki, Predicting recognition between t cell receptors and epitopes with tcrgp, PLoS computational biology **17**, e1008814 (2021).

[18] P. Dash, A. J. Fiore-Gartland, T. Hertz, G. C. Wang, S. Sharma, A. Souquette, J. C. Crawford, E. B. Clemens, T. H. Nguyen, K. Kedzierska, et al., Quantifiable predictive features define epitope-specific t cell receptor repertoires, Nature **547**, 89 (2017).

[19] A. Weber, J. Born, and M. Rodriguez Martínez, Titan: T-cell receptor specificity prediction with bimodal attention networks, Bioinformatics **37**, i237 (2021).

[20] Y. Gao, Y. Gao, Y. Fan, C. Zhu, Z. Wei, C. Zhou, G. Chuai, Q. Chen, H. Zhang, and Q. Liu, Pan-peptide meta learning for t-cell receptor–antigen binding recognition, Nature Machine Intelligence , 1 (2023).

[21] I. Springer, N. Tickotsky, and Y. Louzoun, Contribution of t cell receptor alpha and beta cdr3, mhc typing, v and j genes to peptide binding prediction, Frontiers in immunology **12**, 664514 (2021).

[22] B. P. Kwee, M. Messemaker, E. Marcus, G. Oliveira, W. Scheper, C. Wu, J. Teuwen, and T. Schumacher, Stapler: Efficient learning of tcr-peptide specificity prediction from full-length tcr-peptide data, bioRxiv , 2023 (2023).

[23] Z. Xu, M. Luo, W. Lin, G. Xue, P. Wang, X. Jin, C. Xu, W. Zhou, Y. Cai, W. Yang, et al., Dlptcr: an ensemble deep learning framework for predicting immunogenic peptide recognized by t cell receptor, Briefings in Bioinformatics **22**, bbab335 (2021).

[24] E. Lanzarotti, P. Marcatili, and M. Nielsen, T-cell receptor cognate target prediction based on paired α and β chain sequence and structural cdr loop similarities, Frontiers in immunology **10**, 2080 (2019).

[25] J. J. Moon, H. H. Chu, M. Pepper, S. J. McSorley, S. C. Jameson, R. M. Kedl, and M. K. Jenkins, Naive cd4+ t cell frequency varies for different epitopes and predicts repertoire diversity and response magnitude, Immunity **27**, 203 (2007).

[26] M. Yousef, S. Jung, L. C. Showe, and M. K. Showe, Learning from positive examples when the negative class is undetermined-microrna gene identification, Algorithms for molecular biology **3**, 1 (2008).

[27] P. Perera, P. Oza, and V. M. Patel, One-class classification: A survey, arXiv preprint arXiv:2101.03064 (2021).

[28] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, Advances in neural information processing systems **33**, 1877 (2020).

[29] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al., Improving language understanding by generative pre-training, OpenAI (2018).

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, Attention is all you need, Advances in neural information processing systems **30** (2017).

[31] B. Meynard-Piganeau, C. Fabbri, M. Weigt, A. Pagnani, and C. Feinauer, Generating interacting protein sequences using domain-to-domain translation, bioRxiv , 2022 (2022).

[32] Z. Wu, K. K. Yang, M. J. Liszka, A. Lee, A. Batzilla, D. Wernick, D. P. Weiner, and F. H. Arnold, Signal peptides generated by attention-based neural networks, ACS Synthetic Biology **9**, 2154 (2020).

[33] K. C. Garcia and E. J. Adams, How the t cell receptor sees antigen—a structural view, Cell **122**, 333 (2005).

[34] H. Tanno, T. M. Gould, J. R. McDaniel, W. Cao, Y. Tanno, R. E. Durrett, D. Park, S. J. Cate, W. H. Hildebrand, C. L. Dekker, et al., Determinants governing t cell receptor α/β-chain pairing in repertoire formation of identical twins, Proceedings of the National Academy of Sciences **117**, 532 (2020).

[35] M. Łuksza, Z. M. Sethna, L. A. Rojas, J. Lihm, B. Bravi, Y. Elhanati, K. Soares, M. Amisaki, A. Dobrin, D. Hoyos, et al., Neoantigen quality predicts immunoediting in survivors of pancreatic cancer, Nature **606**, 389 (2022).

[36] B. Reynisson, B. Alvarez, S. Paul, B. Peters, and M. Nielsen, Netmhcpan-4.1 and netmhciipan-4.0: improved predictions of mhc antigen presentation by concurrent motif deconvolution and integration of ms mhc eluted ligand data, Nucleic acids research **48**, W449 (2020).

[37] E. Kondo, B. Maecker, M. R. Weihrauch, C. Wickenhauser, W. Zeng, L. M. Nadler, J. L. Schultze, and M. S. von Bergwelt-Baildon, Cyclin d1–specific cytotoxic t lymphocytes are present in the repertoire of cancer patients: Implications for cancer immunotherapy, Clinical Cancer Research **14**, 6574 (2008).

[38] P. Malekzadeh, A. Pasetto, P. F. Robbins, M. R. Parkhurst, B. C. Paria, L. Jia, J. J. Gartner, V. Hill, Z. Yu, N. P. Restifo, et al., Neoantigen screening identifies broad tp53 mutant immunogenicity in patients with epithelial cancers, The Journal of clinical investigation **129** (2021).

[39] D. Wu, R. Gowathaman, B. G. Pierce, and R. A. Mariuzza, T cell receptors employ diverse strategies to target a p53 cancer neoantigen, Journal of Biological Chemistry **298** (2022).

[40] A. Scardino, D.-A. Gross, P. Alves, J. L. Schultze, S. Graff-Dubois, O. Faure, S. Tourdot, S. Chouaib, L. M. Nadler, F. A. Lemonnier, et al., Her-2/neu and htert cryptic epitopes as novel targets for broad spectrum tumor immunotherapy, The Journal of Immunology **168**, 5900 (2002).

[41] S. S. Tykodi, S. Satoh, J. D. Deming, J. Chou, R. Harrop, and E. H. Warren, Cd8+ t cell clones specific for the 5t4 antigen target renal cell carcinoma tumor-initiating cells in a murine xenograft model, Journal of immunotherapy (Hagerstown, Md.: 1997) **35**, 523 (2012).

[42] A. L. Cox, J. Skipper, Y. Chen, R. A. Henderson, T. L. Darrow, J. Shabanowitz, V. H. Engelhard, D. F. Hunt, and C. L. Slingluff Jr, Identification of a peptide recognized by five melanoma-specific human cytotoxic t cell lines, Science **264**, 716 (1994).

[43] A. A. Minervina, E. A. Komech, A. Titov, M. Bensouda Koraichi, E. Rosati, I. Z. Mamedov, A. Franke, G. A. Efimov, D. M. Chudakov, T. Mora, et al., Longitudinal high-throughput tcr repertoire profiling reveals the dynamics of t-cell memory formation after mild covid-19 infection, Elife **10**, e63502 (2021).

[44] A. S. Shomuradova, M. S. Vagida, S. A. Sheetikov, K. V. Zornikova, D. Kiryukhin, A. Titov, I. O. Peshkova, A. Khmelevskaya, D. V. Dianov, M. Malasheva, et al., Sars-cov-2 epitopes are recognized by a public and diverse repertoire of human t cell receptors, Immunity **53**, 1245 (2020).

[45] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

Chapter 5

Conclusion

5.1 Scope of the Thesis

This thesis investigated the application of artificial intelligence to protein engineering, focusing on the utilization of generative models for designing protein sequences.

The research began with an analysis of generative models' capacity to accurately model protein sequences, comparing complex neural networks to simpler pairwise distribution models. This revealed a significant reliance on second-order interactions for representing protein families effectively.

Further exploration introduced InvMSAFold, a method devised for generating protein sequences with the ability to fold into predefined structures. This represented an integration of MSA-based models with an inverse folding method, aiming to condition the fitness landscape with an emphasis on the protein's structure.

The study then expanded into sequence conditioning, applying these principles to the generation of interacting protein sequences and the prediction of TCR and epitope interactions.

5.2 Limitations

5.2.1 Limitations of Protein Generative Models Validation in Silico

One of the limitations encountered in this thesis is the significant challenge associated with validating in silico protein generative models.

While tools like Alphafold2 have revolutionized our ability to predict protein structures from their sequences, they are primarily trained and validated on functional and naturally occurring protein data. The predictive accuracy and reliability decrease when these tools are applied to synthetic sequences and do not detect completely dysfunctional protein variants [70]. There is a pressing need to develop and validate new computational methods that are specifically tailored to handle synthetic protein sequences. These methods must be capable of dealing not only with the prediction of protein structures but also validate functional properties of interest. This calls for a more integrative approach that combines structural predictions

with functional assays and interaction dynamics in silico.

5.2.2 Experimental Validation as a Critical Bottleneck

The key bottleneck in the pipeline of designing and utilizing synthetic proteins is experimental validation. Validating the functionality of a protein in a laboratory is resource-intensive and technically demanding. It involves long-term collaboration with experimentalists.

The lack of experimental validation impacted the scope and applicability of the findings in this thesis. Each phase of the research—from designing protein sequences using AI models to predicting their interactions—would have substantially benefited from more robust validation. The ability to experimentally test and confirm the predictions made by in silico models would enhance the credibility and utility of the research, providing a clearer path from computational design to practical applications. The increasing availability of less and less expensive methods for high-throughput gene synthesis, functional assaying, and library sequencing will potentially introduce a major change in these points.

5.3 Beyond TULIP

As part of my PhD project, I have initiated a collaboration with Viroxis@GustaveRoussy under the guidance of the group led by Thierry Heidmann, to investigate the use of AI methods for cancer therapeutics.

5.3.1 Selection of the TAX Cancer Target Epitope

The first phase of our collaboration involved selecting a cancer target epitope. The criteria for selection were based on three main considerations:

- Clinical Relevance: The chosen epitope needed to be associated with cancers that are hard to cure, thereby addressing a significant unmet medical need.
- Novelty of the Epitope: We focused on cancer-associated epitopes that were poorly explored, aiming to bring new insights and solutions to difficult-to-treat cancers.
- Feasibility for Testing: The epitope selection was also influenced by the practical aspects of testing in the lab environment at GustaveRoussy.

This careful selection ended in the selection of the HTLV-TAX epitope.

5.3.2 The TAX epitope

The Human T-cell lymphotropic virus type 1 (HTLV-1), identified in the early 1980s, is the causative agent of adult T-cell leukemia/lymphoma (ATLL), a malignancy of

CD4+CD25+ T cells [71]. This discovery marked HTLV-1 as the first human oncogenic retrovirus, revealing its role in a rare but aggressive cancer developing decades after initial infection. The virus employs multiple regulatory proteins, notably Tax and HTLV-1 basic leucine zipper factor (HBZ), to maintain persistence and latency while manipulating host cellular mechanisms to its advantage.

Tax is a key protein in promoting cancer, primarily by disrupting essential cellular pathways like NF-κB, which enhances the growth and survival of infected T cells [72]. This oncogenic potential of Tax, coupled with its ability to evade immune detection, underscores its role in cancer development long after initial infection. Targeting Tax for therapeutic intervention is a promising approach due to its central role in oncogenesis.

5.3.3 Detailed Overview of the HTLV-1 TAX Project

The project targeting the TAX epitope of HTLV-1 has unfolded in several phases, beginning with in silico experimentation. Initial efforts focused on evaluating different metrics, such as perplexity, adversarial loss [73], and Classification Accuracy Score [74], to refine the training of the generative model used in this research, TULIP. This involved generating data with TULIP, training a supervised classifier on this data, and then assessing the model's performance against actual data. This process enabled a detailed examination of the model's training, particularly the effects of randomly masking some Complementarity-Determining Regions to improve the model's resilience to incomplete sequence information. Additionally, the dynamics of model convergence were analyzed to identify the components of the model that required more focused training.

The project also investigated various sampling strategies. Techniques such as adjusting the sampling temperature, employing Classifier-Free Guidance [75] to enhance mutual information, and exploring filtering mechanisms to select the most promising TCR sequences based on specific metrics were all evaluated. This rigorous approach aimed to refine the process of generating artificial sequences, emphasizing quality and relevance.

In the subsequent phase, the artificially generated TCR sequences were forwarded to experimental collaborators at Gustave Roussy. These sequences were used to engineer Jurkat T cells [76], modifying their TCRs to express the synthetic sequences. This step allowed for the in vitro validation of the artificially generated TCRs, testing their functionality and potential therapeutic efficacy. At the time of writing, the experiments are still in progress, mainly at the step of inserting the CDRs inside the framework.

5.4 Future Directions

The work conducted on the HTLV-1 TAX project serves as a foundation for future research in protein engineering and the therapeutic application of AI-generated pro-

tein sequences. The iterative process of model refinement, combined with experimental validation, offers a model for advancing the design and application of TCRs in immunotherapy. The insights gained from this project, particularly in optimizing TCR generation and validation, point towards a continuing evolution of techniques in both computational biology and experimental immunotherapy.

As a further direction, I am also interested in advancing TCR-peptide prediction through multimodal approaches, utilizing structural methods and extensive TCR sequence data to improve TCR design. But more broadly, the integration of artificial intelligence into biological research has led to an expansion in methodological approaches, and the use of multimodal data sources is a challenging direction to explore. The combination of sequence information, structural data, and transcriptomics necessitates the development of specialized methods to manage the complexity and diversity of biological data. A multimodal foundation model for biology could significantly enhance our understanding and manipulation of biological systems.

Moreover, as we integrate increasing amounts of modalities, particularly DNA sequence data, the scalability of current computational models, including transformers, is being pushed to its limits. The complexities of biological data require not only large-scale processing capabilities but also models that can efficiently navigate long contexts. To address these challenges, new approaches such as state space models (SSMs) [77] are emerging. These models offer promising alternatives to traditional transformers by potentially providing more efficient ways to handle large datasets typical in biological research. These models provide a potentially more efficient means of managing large-scale multi-modal biological data, crucial for advancing computational biology.

In conclusion, this work contributed to the field of protein engineering by demonstrating the potential of generative models to enhance our understanding and capability in designing complex protein systems.

Thank you for reading!

Bibliography

[1] Carl Ivar Branden and John Tooze. *Introduction to Protein Structure*. Garland Science. ISBN 978-1-136-96989-8. Google-Books-ID: eUYWBAAAQBAJ.

[2] Pawel Durek and Dirk Walther. The integrated analysis of metabolic and protein interaction networks reveals novel molecular organizing principles. *BMC systems biology*, 2:1–20, 2008.

[3] Derek N Woolfson. A brief history of de novo protein design: minimal, rational, and computational. *Journal of Molecular Biology*, 433(20):167160, 2021.

[4] Vikas Nanda and Ronald L Koder. Designing artificial enzymes by intuition and computation. *Nature chemistry*, 2(1):15–24, 2010.

[5] Beth Allyn Krizek, Barbara T Amann, Valda J Kilfoil, Denise L Merkle, and Jeremy M Berg. A consensus zinc finger peptide: design, high-affinity metal binding, a ph-dependent structure, and a his to cys sequence variant. *Journal of the American Chemical Society*, 113(12):4518–4523, 1991.

[6] Frances H Arnold. Design by directed evolution. *Accounts of chemical research*, 31(3):125–131, 1998.

[7] Yang Yang and Frances H Arnold. Navigating the unnatural reaction space: directed evolution of heme proteins for selective carbene and nitrene transfer. *Accounts of Chemical Research*, 54(5):1209–1225, 2021.

[8] Brian Kuhlman and David Baker. Native protein sequences are close to optimal for their structures. *Proceedings of the National Academy of Sciences*, 97(19): 10383–10388, 2000.

[9] Atomic-Level Accuracy. Design of a novel globular protein fold with. *science*, 1089427(1364):302, 2003.

[10] Lin Jiang, Eric A Althoff, Fernando R Clemente, Lindsey Doyle, Daniela Rothlisberger, Alexandre Zanghellini, Jasmine L Gallaher, Jamie L Betker, Fujie Tanaka, Carlos F Barbas III, et al. De novo computational design of retro-aldol enzymes. *science*, 319(5868):1387–1391, 2008.

[11] Yi Liu and Brian Kuhlman. Rosettadesign server for protein design. *Nucleic acids research*, 34(suppl_2):W235–W238, 2006.

[12] Jeffrey J Gray, Stewart Moughon, Chu Wang, Ora Schueler-Furman, Brian Kuhlman, Carol A Rohl, and David Baker. Protein–protein docking with simultaneous optimization of rigid-body displacement and side-chain conformations. *Journal of molecular biology*, 331(1):281–299, 2003.

[13] Joost Schymkowitz, Jesper Borg, Francois Stricher, Robby Nys, Frederic Rousseau, and Luis Serrano. The foldx web server: an online force field. *Nucleic acids research*, 33(suppl_2):W382–W388, 2005.

[14] Raphael Guerois, Jens Erik Nielsen, and Luis Serrano. Predicting changes in the stability of proteins and protein complexes: a study of more than 1000 mutations. *Journal of molecular biology*, 320(2):369–387, 2002.

[15] Geoffrey E Hinton and Terrence J Sejnowski. Optimal perceptual inference. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, volume 448, pages 448–453. Citeseer, 1983.

[16] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[17] Miguel A Carreira-Perpinan and Geoffrey Hinton. On contrastive divergence learning. In *International workshop on artificial intelligence and statistics*, pages 33–40. PMLR, 2005.

[18] Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071, 2008.

[19] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[20] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37:183–233, 1999.

[21] Matching. `https://yang-song.net/blog/2021/score/`. Accessed: 2024-03-19.

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

[23] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pages 8162–8171. PMLR, 2021.

[24] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[26] Pascal Notin, Nathan Rollins, Yarin Gal, Chris Sander, and Debora Marks. Machine learning for functional protein design. *Nature Biotechnology*, 42(2):216–228, 2024.

[27] Andrew Giessel, Athanasios Dousis, Kanchana Ravichandran, Kevin Smith, Sreyoshi Sur, Iain McFadyen, Wei Zheng, and Stuart Licht. Therapeutic enzyme engineering using a generative neural network. *Scientific Reports*, 12(1): 1536, 2022.

[28] Kiera H Sumida, Reyes Núñez Franco, Indrek Kalvet, Samuel J Pellock, Basile IM Wicky, Lukas F Milles, Justas Dauparas, Jue Wang, Yakov Kipnis, Noel Jameson, et al. Improving protein expression, stability, and function with proteinmpnn. *bioRxiv*, pages 2023–10, 2023.

[29] Derek M Mason, Simon Friedensohn, Cédric R Weber, Christian Jordi, Bastian Wagner, Simon M Meng, Roy A Ehling, Lucia Bonati, Jan Dahinden, Pablo Gainza, et al. Optimization of therapeutic antibodies by predicting antigen specificity from antibody sequence via deep learning. *Nature Biomedical Engineering*, 5(6):600–612, 2021.

[30] Zachary Wu, SB Jennifer Kan, Russell D Lewis, Bruce J Wittmann, and Frances H Arnold. Machine learning-assisted directed protein evolution with combinatorial libraries. *Proceedings of the National Academy of Sciences*, 116(18): 8852–8858, 2019.

[31] R Verkuil, O Kabeli, Y Du, BI Wicky, LF Milles, J Dauparas, D Baker, S Ovchinnikov, T Sercu, and A Rives. Language models generalize beyond natural proteins. biorxiv. 2022.

[32] The UniProt Consortium. UniProt: the universal protein knowledgebase in 2023. 51:D523–D531. ISSN 0305-1048. doi: 10.1093/nar/gkac1052. URL https://doi.org/10.1093/nar/gkac1052.

[33] Stephen K Burley, Charmi Bhikadiya, Chunxiao Bi, Sebastian Bittrich, Henry Chao, Li Chen, Paul A Craig, Gregg V Crichlow, Kenneth Dalenberg, Jose M Duarte, Shuchismita Dutta, Maryam Fayazi, Zukang Feng, Justin W Flatt, Sai Ganesan, Sutapa Ghosh, David S Goodsell, Rachel Kramer Green, Vladimir Guranovic, Jeremy Henry, Brian P Hudson, Igor Khokhriakov, Catherine L Lawson, Yuhe Liang, Robert Lowe, Ezra Peisach, Irina Persikova, Dennis W Piehl, Yana Rose, Andrej Sali, Joan Segura, Monica Sekharan, Chenghua Shao, Brinda Vallat, Maria Voigt, Ben Webb, John D Westbrook, Shamara Whetstone, Jasmine Y Young, Arthur Zalevsky, and Christine Zardecki. RCSB pro-

tein data bank (RCSB.org): delivery of experimentally-determined PDB structures alongside one million computed structure models of proteins from artificial intelligence/machine learning. 51:D488–D508. ISSN 0305-1048. doi: 10.1093/nar/gkac1077. URL https://doi.org/10.1093/nar/gkac1077.

[34] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. 596(7873):583–589. ISSN 1476-4687. doi: 10.1038/s41586-021-03819-2. URL https://www.nature.com/articles/s41586-021-03819-2. Number: 7873 Publisher: Nature Publishing Group.

[35] Michael Heinzinger, Konstantin Weissenow, Joaquin Gomez Sanchez, Adrian Henkel, Martin Steinegger, and Burkhard Rost. Prostt5: Bilingual language model for protein sequence and structure. *bioRxiv*, pages 2023–07, 2023.

[36] Roderic Guigo. An introduction to position specific scoring matrices. *Bioinformatica. upf. edu*, 333, 2016.

[37] Robert D Finn, Jody Clements, and Sean R Eddy. Hmmer web server: interactive sequence similarity searching. *Nucleic acids research*, 39(suppl_2):W29–W37, 2011.

[38] Emre Sevgen, Joshua Moller, Adrian Lange, John Parker, Sean Quigley, Jeff Mayer, Poonam Srivastava, Sitaram Gayatri, David Hosfield, Maria Korshunova, et al. Prot-vae: protein transformer variational autoencoder for functional protein design. *bioRxiv*, pages 2023–01, 2023.

[39] Ali Madani, Ben Krause, Eric R Greene, Subu Subramanian, Benjamin P Mohr, James M Holton, Jose Luis Olmos, Caiming Xiong, Zachary Z Sun, Richard Socher, et al. Large language models generate functional protein sequences across diverse families. *Nature Biotechnology*, 41(8):1099–1106, 2023.

[40] Noelia Ferruz, Steffen Schmidt, and Birte Höcker. Protgpt2 is a deep unsupervised language model for protein design. *Nature communications*, 13(1):4348, 2022.

[41] Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C Lawrence Zitnick, Jerry Ma, et al. Biological

structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15): e2016239118, 2021.

[42] Roshan M Rao, Jason Liu, Robert Verkuil, Joshua Meier, John Canny, Pieter Abbeel, Tom Sercu, and Alexander Rives. Msa transformer. In *International Conference on Machine Learning*, pages 8844–8856. PMLR, 2021.

[43] Faruck Morcos, Andrea Pagnani, Bryan Lunt, Arianna Bertolino, Debora S Marks, Chris Sander, Riccardo Zecchina, José N Onuchic, Terence Hwa, and Martin Weigt. Direct-coupling analysis of residue coevolution captures native contacts across many protein families. *Proceedings of the National Academy of Sciences*, 108(49):E1293–E1301, 2011.

[44] William P Russ, Matteo Figliuzzi, Christian Stocker, Pierre Barrat-Charlaix, Michael Socolich, Peter Kast, Donald Hilvert, Remi Monasson, Simona Cocco, Martin Weigt, et al. An evolution-based model for designing chorismate mutase enzymes. *Science*, 369(6502):440–445, 2020.

[45] Magnus Ekeberg, Tuomo Hartonen, and Erik Aurell. Fast pseudolikelihood maximization for direct-coupling analysis of protein structure from many homologous amino-acid sequences. *Journal of Computational Physics*, 276:341–356, 2014.

[46] Lukas Theo Schmitt, Maciej Paszkowski-Rogacz, Florian Jug, and Frank Buchholz. Prediction of designer-recombinases for dna editing with generative deep learning. *Nature Communications*, 13(1):7966, 2022.

[47] Jannis Born and Matteo Manica. Regression transformer enables concurrent sequence regression and generation for molecular language modelling. *Nature Machine Intelligence*, 5(4):432–444, 2023.

[48] Mihaly Varadi, Stephen Anyango, Mandar Deshpande, Sreenath Nair, Cindy Natassia, Galabina Yordanova, David Yuan, Oana Stroe, Gemma Wood, Agata Laydon, Augustin Žídek, Tim Green, Kathryn Tunyasuvunakool, Stig Petersen, John Jumper, Ellen Clancy, Richard Green, Ankur Vora, Mira Lutfi, Michael Figurnov, Andrew Cowie, Nicole Hobbs, Pushmeet Kohli, Gerard Kleywegt, Ewan Birney, Demis Hassabis, and Sameer Velankar. AlphaFold protein structure database: massively expanding the structural coverage of protein-sequence space with high-accuracy models. 50:D439–D444. ISSN 0305-1048. doi: 10.1093/nar/gkab1061. URL https://doi.org/10.1093/nar/gkab1061.

[49] Namrata Anand, Raphael Eguchi, and Po-Ssu Huang. Fully differentiable full-atom protein backbone generation. 2019.

[50] Raphael R Eguchi, Christian A Choe, and Po-Ssu Huang. Ig-vae: Generative modeling of protein structure by direct 3d coordinate generation. *PLoS computational biology*, 18(6):e1010271, 2022.

[51] 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, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, 2023.

[52] 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, et al. Illuminating protein space with a programmable generative model. *Nature*, 623(7989):1070–1078, 2023.

[53] Justas Dauparas, Ivan Anishchenko, Nathaniel Bennett, Hua Bai, Robert J Ragotte, Lukas F Milles, Basile IM Wicky, Alexis Courbet, Rob J de Haas, Neville Bethel, et al. Robust deep learning–based protein sequence design using proteinmpnn. *Science*, 378(6615):49–56, 2022.

[54] Chloe Hsu, Robert Verkuil, Jason Liu, Zeming Lin, Brian Hie, Tom Sercu, Adam Lerer, and Alexander Rives. Learning inverse folding from millions of predicted structures. In *International conference on machine learning*, pages 8946–8970. PMLR, 2022.

[55] Thomas J Kindt, Richard A Goldsby, Barbara A Osborne, and Janis Kuby. *Kuby immunology*. Macmillan, 2007.

[56] Jinwoo Leem, Saulo H P de Oliveira, Konrad Krawczyk, and Charlotte M Deane. Stcrdab: the structural t-cell receptor database. *Nucleic acids research*, 46(D1): D406–D412, 2018.

[57] Neerja Thakkar and Chris Bailey-Kellogg. Balancing sensitivity and specificity in distinguishing tcr groups by cdr sequence similarity. *BMC bioinformatics*, 20: 1–14, 2019.

[58] Thierry Mora and Aleksandra M Walczak. Quantifying lymphocyte receptor diversity. In *Systems immunology*, pages 183–198. CRC Press, 2018.

[59] Pieter Moris, Joey De Pauw, Anna Postovskaya, Sofie Gielis, Nicolas De Neuter, Wout Bittremieux, Benson Ogunjimi, Kris Laukens, and Pieter Meysman. Current challenges for unseen-epitope tcr interaction prediction and a new perspective derived from image classification. *Briefings in Bioinformatics*, 22(4): bbaa318, 2021.

[60] Wen Zhang, Peter G Hawkins, Jing He, Namita T Gupta, Jinrui Liu, Gabrielle Choonoo, Se W Jeong, Calvin R Chen, Ankur Dhanik, Myles Dillon, et al. A framework for highly multiplexed dextramer mapping and prediction of t cell receptor sequences to antigen specificity. *Science advances*, 7(20):eabf5835, 2021.

[61] Alessandro Montemurro, Leon Eyrich Jessen, and Morten Nielsen. Nettcr-2.1: Lessons and guidance on how to develop models for tcr specificity predictions. *Frontiers in immunology*, 13:1055151, 2022.

[62] John-William Sidhom, H Benjamin Larman, Drew M Pardoll, and Alexander S Baras. Deeptcr is a deep learning framework for revealing sequence concepts within t-cell repertoires. *Nature communications*, 12(1):1605, 2021.

[63] Yao Tong, Jiayin Wang, Tian Zheng, Xuanping Zhang, Xiao Xiao, Xiaoyan Zhu, Xin Lai, and Xiang Liu. Sete: sequence-based ensemble learning approach for tcr epitope binding prediction. *Computational biology and chemistry*, 87:107281, 2020.

[64] Emmi Jokinen, Jani Huuhtanen, Satu Mustjoki, Markus Heinonen, and Harri Lähdesmäki. Predicting recognition between t cell receptors and epitopes with tcrgp. *PLoS computational biology*, 17(3):e1008814, 2021.

[65] Koshlan Mayer-Blackwell, Stefan Schattgen, Liel Cohen-Lavi, Jeremy C Crawford, Aisha Souquette, Jessica A Gaevert, Tomer Hertz, Paul G Thomas, Philip Bradley, and Andrew Fiore-Gartland. Tcr meta-clonotypes for biomarker discovery with tcrdist3 enabled identification of public, hla-restricted clusters of sars-cov-2 tcrs. *Elife*, 10:e68605, 2021.

[66] Alessandro Montemurro, Viktoria Schuster, Helle Rus Povlsen, Amalie Kai Bentzen, Vanessa Jurtz, William D Chronister, Austin Crinklaw, Sine R Hadrup, Ole Winther, Bjoern Peters, et al. Nettcr-2.0 enables accurate prediction of tcr-peptide binding by using paired tcrα and β sequence data. *Communications biology*, 4(1):1060, 2021.

[67] Daniel Hesslow, Niccoló Zanichelli, Pascal Notin, Iacopo Poli, and Debora Marks. Rita: a study on scaling up generative protein sequence models. *arXiv preprint arXiv:2205.05789*, 2022.

[68] Qinghua He, Zhaoyu Liu, Zhihua Liu, Yuxiong Lai, Xinke Zhou, and Jinsheng Weng. Tcr-like antibodies in cancer immunotherapy. *Journal of hematology & oncology*, 12:1–13, 2019.

[69] Mark R Middleton, Cheryl McAlpine, Victoria K Woodcock, Pippa Corrie, Jeffrey R Infante, Neil M Steven, Thomas R Jeffry Evans, Alan Anthoney, Alexander N Shoushtari, Omid Hamid, et al. Tebentafusp, a tcr/anti-cd3 bispecific fusion protein targeting gp100, potently activated antitumor immune responses in patients with metastatic melanoma. *Clinical Cancer Research*, 26(22):5869–5878, 2020.

[70] Marina A Pak, Karina A Markhieva, Mariia S Novikova, Dmitry S Petrov, Ilya S Vorobyev, Ekaterina S Maksimova, Fyodor A Kondrashov, and Dmitry N Ivankov. Using alphafold to predict the impact of single mutations on protein stability and function. *Plos one*, 18(3):e0282689, 2023.

[71] Xiaorui Zuo, Ruoning Zhou, Sikai Yang, and Guangyong Ma. Htlv-1 persistent infection and atll oncogenesis. *Journal of Medical Virology*, 95(1):e28424, 2023.

[72] Rita Hleihel, Hala Skayneh, Hugues de Thé, Olivier Hermine, and Ali Bazarbachi. Primary cells from patients with adult t cell leukemia/lymphoma depend on htlv-1 tax expression for nf-κb activation and survival. *Blood Cancer Journal*, 13(1):67, 2023.

[73] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1):53–65, 2018.

[74] Suman Ravuri and Oriol Vinyals. Classification accuracy score for conditional generative models. *Advances in neural information processing systems*, 32, 2019.

[75] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[76] Robert T Abraham and Arthur Weiss. Jurkat t cells and development of the t-cell receptor signalling paradigm. *Nature reviews immunology*, 4(4):301–308, 2004.

[77] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Appendix A

Appendix

A.1 Supplementary: Interpretable Pairwise Distillations for Generative Protein Sequence Models

APPENDIX: INTERPRETABLE PAIRWISE DISTILLATIONS FOR GENERATIVE PROTEIN SEQUENCE MODELS

Christoph Feinauer
Department of Decision Sciences
Bocconi Institute for Data Science and Analytics (BIDSA) Bocconi University, Milan, Italy
christoph.feinauer@unibocconi.it

Barthelemy Meynard-Piganeau
Laboratory of Computational and Quantitative Biology (LCQB) UMR 7238 CNRS - Sorbonne Université, Paris, France
Department of Applied Science and Technologies (DISAT), Politecnico di Torino, Torino, Italy
barthelemy.meynard@polytechnique.edu

Carlo Lucibello
Department of Decision Sciences
Bocconi Institute for Data Science and Analytics (BIDSA) Bocconi University, Milan, Italy
carlo.lucibello@unibocconi.it

June 15, 2022

A Data and Preprocessing

In the following we report the properties of the datasets used.

Dataset	Length	Sequences	Unique Sequences	Train Sequences	Test Sequences
BRCA1	75	39396	21639	19476	2163
GAL4	62	22985	15833	14250	1583
SUMO1	76	21695	8719	7848	871
UBE4B	75	16478	10248	9224	1024
YAP1	30	86353	17953	16158	1795

Table A1: Length and number of sequences used for all 5 datasets

Dataset	Measurement	Number of Mutants
BRCA1	function_score	494
GAL4	SEL_A_24h	1104
SUMO1	screenscore	1404
UBE4B	log2_ratio	603
YAP1	linear	313

Table A2: Characteristics of experimental datasets. The 'Measurement' column indicates which measurement was taken, where the names correspond to the ones used in the supplemental material of [1].

B Additional Results

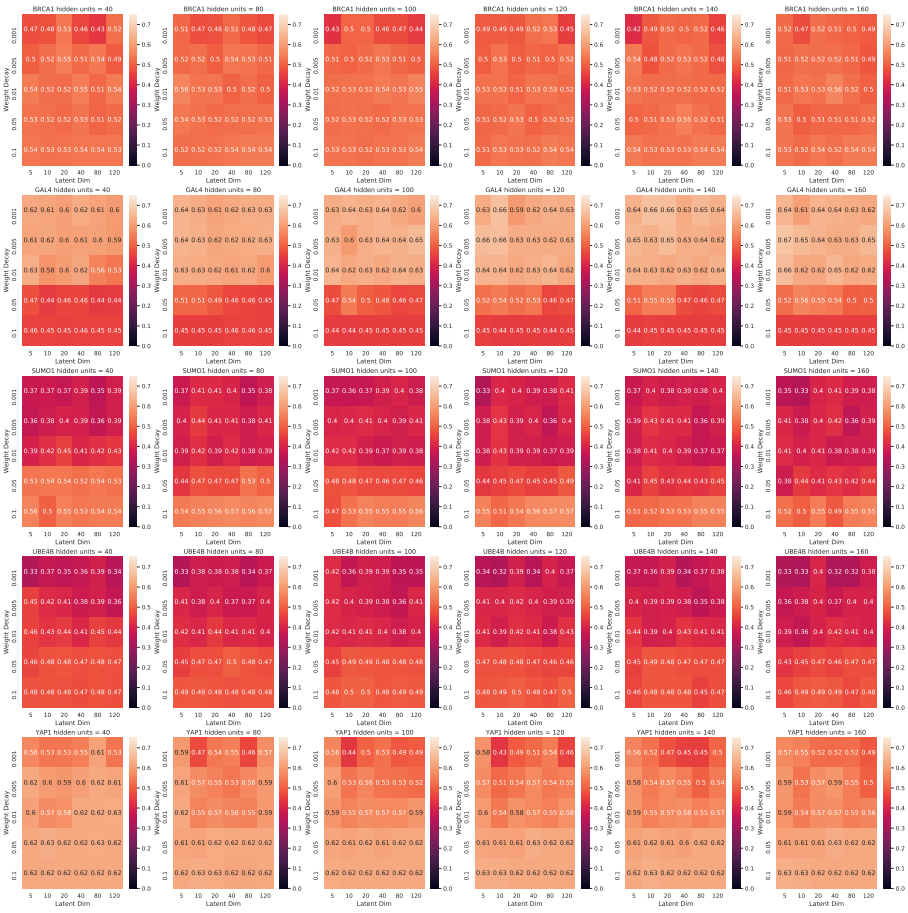


Figure B1: Spearman correlation with experimental values of VAE models trained with different hyperparameters. Every row corresponds to a dataset, every column to a different number of hidden units in the VAE encoders and decoders. Within every subplot, the rows correspond to different settings for the weight decay strength and the coulmns to different sizes for the latent dimension. The colors follow the Spearman correlation (the lighter the higher).

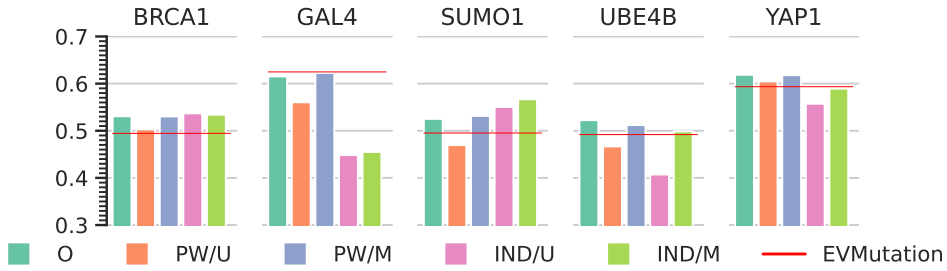


Figure B2: **Independent reproduction of main text Fig. 7: Spearman Correlation with experimental data of original (O) and extracted models (PW/U, PW/M, IND/U, IND/M) for ArDCA**. Shown is the Spearman rank correlation between the experimental data and the energies of the original model (O), the pairwise and independent models extracted using samples from a uniform distribution (PW/U and IND/U) and for the pairwise and independent models extracted using samples from the original model distribution (PW/M and IND/M). The random seeds changed for this reproduction are the seed used for the train-test split, the sampling of the sequences from the original models and the stochastic gradient descent using the Adam optimizer when extracting the model.

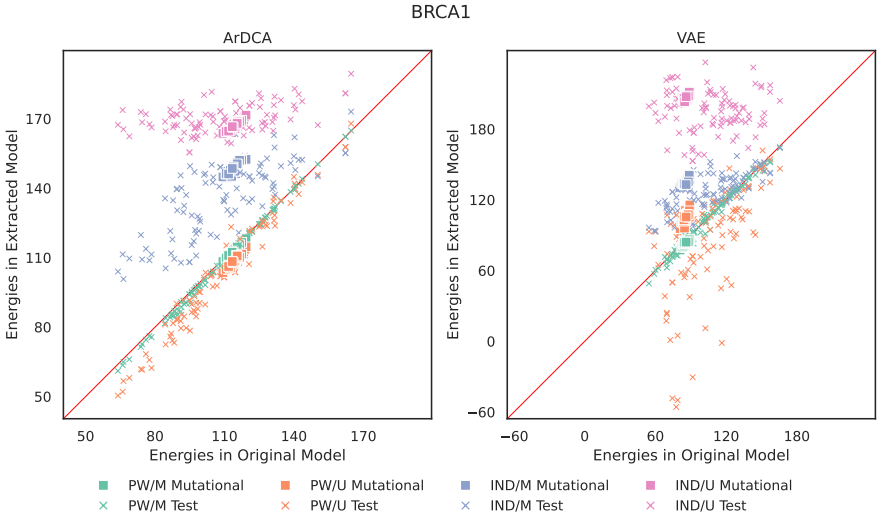


Figure B3: **Scatterplot Energies on BRCA1 Test Set** Shown are energies in the original models versus energies in the extracted models on the sequences in the test set (crosses) and the sequences from the mutational datasets (squares). The points for the test sequences correspond to 100 randomly chosen sequences. The VAE model used has 40 hidden units, a latent representation of size 5 and a weight decay setting of 0.01.

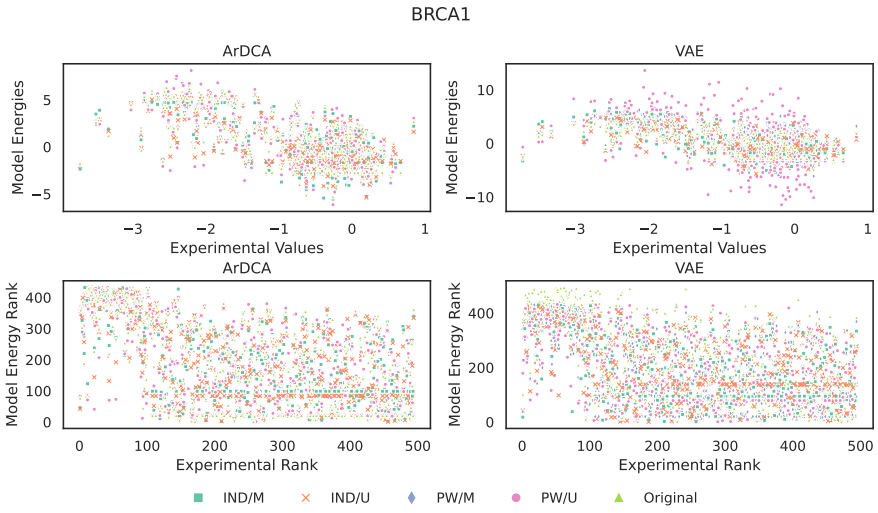


Figure B4: **Scatterplot Ranks and Energies on Sequences in Mutational Datasets for BRCA1** Shown are energies versus experimental fitness (upper two panels) and ranks of energies versus ranks of experimental fitness (lower two panels) for original and extracted models. Note that energies are negatively proportional to the log probability, so lower energy means higher probabilities. The VAE model used has 40 hidden units, a latent representation of size 5 and a weight decay setting of 0.01. If the number of mutants in the dataset was larger than 500, the plot shows data for a random subset of 500 mutants. The energies were normalized to have 0 mean for all models independently.

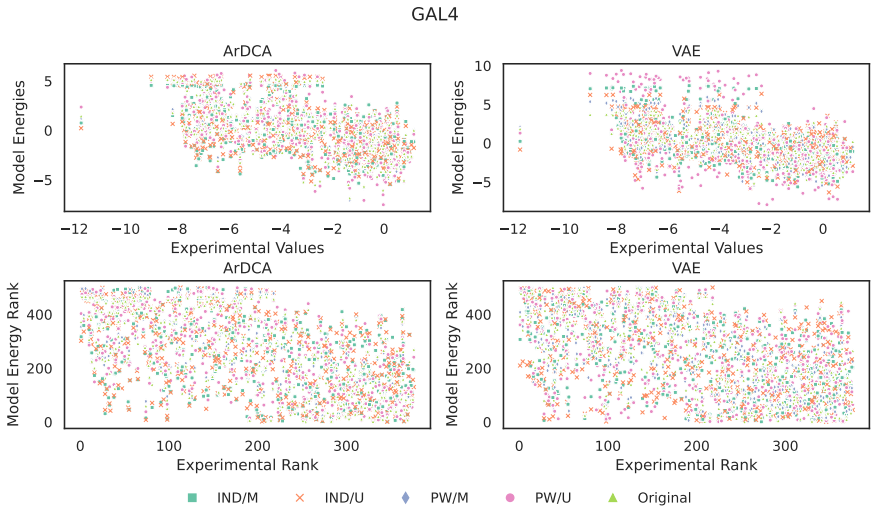


Figure B5: **Scatterplot Ranks and Energies on Sequences in Mutational Datasets for GAL4** Shown are energies versus experimental fitness (upper two panels) and ranks of energies versus ranks of experimental fitness (lower two panels) for original and extracted models. Note that energies are negatively proportional to the log probability, so lower energy means higher probabilities. The VAE model used has 40 hidden units, a latent representation of size 5 and a weight decay setting of 0.01. If the number of mutants in the dataset was larger than 500, the plot shows data for a random subset of 500 mutants. The energies were normalized to have 0 mean for all models independently.

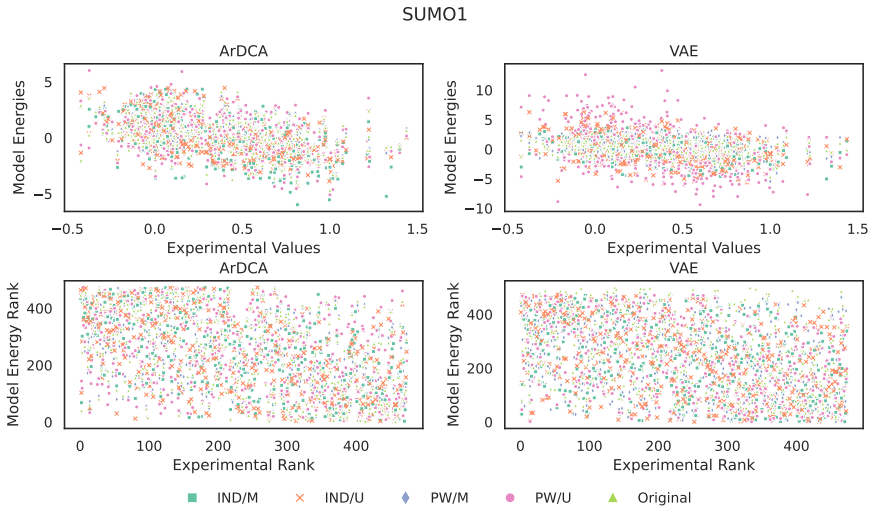


Figure B6: **Scatterplot Ranks and Energies on Sequences in Mutational Datasets for SUMO1** Shown are energies versus experimental fitness (upper two panels) and ranks of energies versus ranks of experimental fitness (lower two panels) for original and extracted models. Note that energies are negatively proportional to the log probability, so lower energy means higher probabilities. The VAE model used has 40 hidden units, a latent representation of size 5 and a weight decay setting of 0.01. If the number of mutants in the dataset was larger than 500, the plot shows data for a random subset of 500 mutants. The energies were normalized to have 0 mean for all models independently.

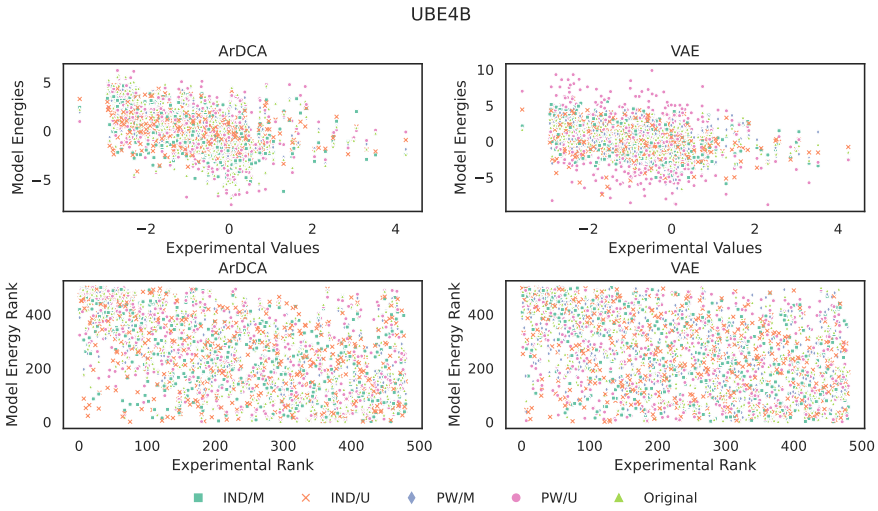


Figure B7: **Scatterplot Ranks and Energies on Sequences in Mutational Datasets for UBE4B** Shown are energies versus experimental fitness (upper two panels) and ranks of energies versus ranks of experimental fitness (lower two panels) for original and extracted models. Note that energies are negatively proportional to the log probability, so lower energy means higher probabilities. The VAE model used has 40 hidden units, a latent representation of size 5 and a weight decay setting of 0.01. If the number of mutants in the dataset was larger than 500, the plot shows data for a random subset of 500 mutants. The energies were normalized to have 0 mean for all models independently.

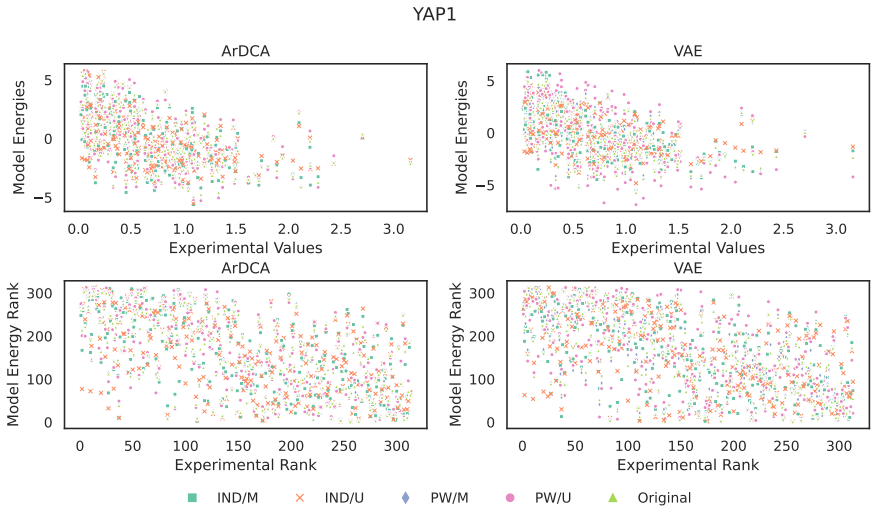


Figure B8: **Scatterplot Ranks and Energies on Sequences in Mutational Datasets for YAP1** Shown are energies versus experimental fitness (upper two panels) and ranks of energies versus ranks of experimental fitness (lower two panels) for original and extracted models. Note that energies are negatively proportional to the log probability, so lower energy means higher probabilities. The VAE model used has 40 hidden units, a latent representation of size 5 and a weight decay setting of 0.01. If the number of mutants in the dataset was larger than 500, the plot shows data for a random subset of 500 mutants. The energies were normalized to have 0 mean for all models independently.

C Contact Prediction

We use standard methods for contact prediction from pairwise models, following mainly [2]. We transform the extracted pairwise models into the zero-sum gauge and calculate the Frobenius norm of the $q-1 \times q-1$ submatrices J_{ij} corresponding to the pair of positions i and j (we do not sum over gap states, hence $q-1$ instead of q). We apply the *average-product correction* [3] and sort the positions pairs by the resulting score, excluding pairs for which $abs(i-j) < 5$. We map PDB 1PIN:A [4] to the MSA and use it to differentiate contacts from non-contacts (8 Å, Heavy-Atom criterion [5]).

D Zero-Sum Gauge

In the following we prove that the pairwise model E^{pw} corresponding to the minimizer of main text Eq. 7 is equivalent to the pairwise part of E^M in the zero-sum gauge when using the uniform distribution D for extraction.

D.1 Notation

We denote by $\mathcal{A} = \{1, .., q\}$ the (numeric) alphabet of the q possible amino acids. The terms $f_L : \mathcal{A}^{|L|} \to \mathcal{R}$ in the general expansion in main text Eq. 4 are functions mapping sequences of amino acids of length $|L|$ to a real number, where $L \subseteq I = \{1, \ldots, N\}$ is a subsequence of positions. In this notation, the pairwise model we train using the loss in main text Eq. 7 can be written as

$$E^{pw}(s) = \sum_{i=1}^{N} \sum_{j=i+1}^{N} f_{ij}^{pw}(s_i, s_j) + \sum_{i=1}^{N} f_i(s_i) + f_\emptyset. \tag{1}$$

In main text Eq. 5 we use a different notation for the pairwise model, but in this Appendix we decide to keep all notations compatible with the generic expansion in main text Eq. 4. The notations can be connected by identifying $f_i^{pw}(a) := -h_i(a)$, $f_{ij}^{pw}(a,b) := -J_{ij}(a,b)$ and $f_\emptyset := -C$ for arbitrary amino acids a and b.

Equivalently we define $f_L^M : \mathcal{A}^{|L|} \to \mathcal{R}$ as the interaction coefficients between the sites belonging to the set of positions $L \subseteq I$ in E^M in a certain gauge.

We will use $f_L(a_L)$ in order to denote a specific interaction coefficient for a fixed sequence of amino acids a_L of length $|L|$, for both pairwise models and models with higher-order interactions. We will use f^{pw} to denote the set of all parameters of the pairwise model and f^M for the set of all parameters of the original model.

D.2 Zero-Sum Gauge

The zero-sum gauge is a reparameterization of the interaction coefficients which leaves the energy invariant (see also Ref. [6] who discuss this gauge, calling it the *Ising* gauge). In this gauge, if $|L| > 0$, summing $f_L(a_L)$ over any of the amino acids in a_L while keeping the others fixed is 0. It can be applied both to the parameters of the extracted pairwise model f^{pw} and the parameters f^M of the original model. Since the sum over an amino acid is proportional to the expectation of $f_L(a_L)$ when the corresponding amino acid is sampled uniformly, this condition can be written as

$$\mathbb{E}_{s\sim U}[f_L(s_L)|s_J = a_J] = 0 \quad \forall\, J \subset L, \tag{2}$$

where $\mathbb{E}_{s\sim U}[f_L(s_L)|s_J = a_J]$ is the expectation of $f_L(s_L)$ if the subsequence s_J is fixed to a_J. Any model can be transformed into the zero-sum gauge using the identity $f_L(a_L) = (f_L(a_L) - \hat{f}_L(a_L)) + \hat{f}_L(a_L)$ with

$$\hat{f}_L(a_L) := \sum_{J\subseteq L} (-1)^{|J|} \frac{1}{q^{|J|}} \sum_{a_J} f^L(a_L). \tag{3}$$

It is easy to show that $\hat{f}(a_L)$ satisfies the condition in Eq. 2 and that $f_{a_L}(a_L) - \hat{f}_{a_L}(a_L)$ contains only interactions of order strictly less than $|L|$. Therefore, any model can be transformed into the zero-sum gauge by first applying the transformation to the interaction coefficients at the highest order $N = |I|$. This will lead to interaction coefficients at order N that satisfy the condition in Eq. 2 and new interaction coefficients of order lower than N. These can be

absorbed in the interaction coefficients in the lower orders of the expansion. Repeating this procedure at $N-1$, then at $N-2$ etc. leads to a final model where all interaction coefficients of all orders satisfy the condition in Eq. 2.

Since the expansion of E^M has exponentially many interaction coefficients in general, this procedure has no practical use in our setting. However, in the next section we show that the lower orders of E^M in the zero-sum gauge representation can be extracted with a simple sampling estimator.

D.3 Proof of Equivalence of Minimizer of Loss and Zero-Sum Gauge

The partial derivative of the loss in main text Eq. 7 with respect to a parameter $f_L^{pw}(a_L)$ in the pairwise model (note that $|L| \leq 2$ in this case) can be written as

$$\frac{\partial \mathcal{L}(f^{pw})}{\partial f_L^{pw}(a_L)} = 2\,\mathbb{E}_{s\sim U}\left[\left(E^{pw}(s) - E^M(s)\right)\frac{\partial E^{pw}(s)}{\partial f_L^{pw}(a_L)}\right]. \tag{4}$$

Setting the gradient to 0 leads to

$$\mathbb{E}_{s\sim U}[E^M(s)|s_L = a_L] = \mathbb{E}_{s\sim U}[E^{pw}(s)|s_L = a_L] \quad \forall L : |L| \leq 2 \tag{5}$$

which means that the minimisation of the loss with respect to the parameters of the pairwise model is equivalent to fitting the conditional expectation of the energy under uniform distribution up to the second order of the expansion.

Since the loss in main text Eq. 7 is invariant with respect to a gauge change in the pairwise model E^{pw}, we can assume without loss of generality that we extract the pairwise model in the zero-sum gauge representation. Using a hat to denote the parameters $\hat{f}^{pw}$ of the pairwise model in this specific gauge, it is easy to see from Eq. 1 and the condition in Eq. 2 that

$$\begin{aligned}
\mathbb{E}_{s\sim U}[E^{pw}(s)] &= \hat{f}^{pw}_{\emptyset} \\
\mathbb{E}_{s\sim U}[E^{pw}(s)|s_i = a] &= \hat{f}^{pw}_{i}(a) + \hat{f}^{pw}_{\emptyset} \\
\mathbb{E}_{s\sim U}[E^{pw}(s)|s_i = a, s_j = b] &= \hat{f}^{pw}_{i,j}(a,b) + \hat{f}^{pw}_{i}(a) + \hat{f}^{pw}_{j}(b) + \hat{f}_{\emptyset}.
\end{aligned}$$

Combining this with Eq. 5 we get at the minimum of the loss the conditions

$$\begin{aligned}
\mathbb{E}_{s\sim U}[E^{M}(s)] &= \hat{f}^{pw}_{\emptyset} \\
\mathbb{E}_{s\sim U}[E^{M}(s)|s_i = a] &= \hat{f}^{pw}_{i}(a) + \hat{f}^{pw}_{\emptyset} \\
\mathbb{E}_{s\sim U}[E^{M}(s)|s_i = a, s_j = b] &= \hat{f}^{pw}_{i,j}(a,b) + \hat{f}^{pw}_{i}(a) + \hat{f}^{pw}_{j}(b) + \hat{f}_{\emptyset}.
\end{aligned} \tag{6}$$

Similar to the pairwise model, we will use a hat to denote the parameters $\hat{f}^M$ of the model E^M in the zero-sum gauge. While the corresponding expansion

$$E^M(s) = \sum_{L\subseteq I} \hat{f}^M_L(s_L)$$

has interaction coefficients of all orders, we can again use the conditions in Eq. 2 to arrive at

$$\begin{aligned}
\mathbb{E}_{s\sim U}[E^{M}(s)] &= \hat{f}^{M}_{\emptyset} \\
\mathbb{E}_{s\sim U}[E^{M}(s)|s_i = a] &= \hat{f}^{M}_{i}(a) + \hat{f}^{M}_{\emptyset} \\
\mathbb{E}_{s\sim U}[E^{M}(s)|s_i = a, s_j = b] &= \hat{f}^{M}_{i,j}(a,b) + \hat{f}^{M}_{i}(a) + \hat{f}^{M}_{j}(b) + \hat{f}_{\emptyset}.
\end{aligned}$$

Taking these relations together leads to the minimizer condition

$$\hat{f}_L^{pw} = \hat{f}_L^M \quad \forall L : |L| \leq 2$$

which means that the E^{pw} minimizing the loss in main text Eq. 7 is the pairwise part of E^M in its zero-sum gauge representation. Note that the loss is still invariant with respect to a gauge change in the extracted pairwise model, so the extracted model can be in any gauge representation.

We also note that Eqs. 6 can be used to estimate the coefficients of the extracted pairwise model directly using uniform samples and the corresponding energies from the original models in order to approximate the expectations.

E List of Abbreviations in Main Text

Abbreviations in Figures and Text
PW: Pairwise Model **IND**: Independent Model **O**: Original neural network model **M**: Distribution induced by original neural network model **U**: Uniform Distribution **PW/M**: Pairwise model extracted using samples from the M distribution and energies from O **PW/U**: Pairwise model extracted using samples from the U distribution and energies from O **IND/M**: Independent model extracted using samples from the M distribution and energies from O **IND/U**: Independent model extracted using samples from the U distribution and energies from O **Test Distant**: 10% of test sequences with largest Hamming distance to the training set **Test Close**: Test sequences not in 'Test Distant'

References

[1] Riesselman AJ, Ingraham JB, Marks DS. Deep generative models of genetic variation capture the effects of mutations. Nature methods. 2018;15(10):816–822.

[2] Ekeberg M, Lövkvist C, Lan Y, Weigt M, Aurell E. Improved contact prediction in proteins: using pseudolikelihoods to infer Potts models. Physical Review E. 2013;87(1):012707.

[3] Dunn SD, Wahl LM, Gloor GB. Mutual information without the influence of phylogeny or entropy dramatically improves residue contact prediction. Bioinformatics. 2008;24(3):333–340.

[4] Ranganathan R, Lu KP, Hunter T, Noel JP. Structural and functional analysis of the mitotic rotamase Pin1 suggests substrate recognition is phosphorylation dependent. Cell. 1997;89(6):875–886.

[5] Morcos F, Pagnani A, Lunt B, Bertolino A, Marks DS, Sander C, et al. Direct-coupling analysis of residue coevolution captures native contacts across many protein families. Proceedings of the National Academy of Sciences. 2011;108(49):E1293–E1301.

[6] Zamuner S, Rios PDL. Interpretable Neural Networks based classifiers for categorical inputs. arXiv preprint arXiv:210203202. 2021;.

A.2 Supplementary: Uncovering Sequence Diversity from a Known Protein Structure

A Appendix

A.1 Fast pseudolikelihood computation

The main rationale behind the low rank approximation (1) is two-fold; first, it is known that, compared to the covariance matrix, the coupling matrix has often a rank which is much smaller than the maximum theoretical one(do we have a reference?). Secondly, for computational purposes we wanted to avoid bottleneck's of quadratic costs in the length of the protein, both from a memory and a computational time point-of-view. Technically, once we enforce a low rank constraint of the matrix J, the number of active parameters indeed passes from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$. If we though compute the pseudo-likelihood naively by pre-computing the full coupling matrix J as in (**??**), we end up again into a quadratic cost. Luckily, with a careful implementation, we can indeed achieve a linear memory and computational cost. To see this, let us rewrite the crucial part(as the fields are by nature linear in cost):

$$\sum_{i<j,a,b} J_{i,j}[a,b]\delta_i[a]\delta_j[b] = \frac{1}{2}\left(\sum_{i,j,a,b}\sum_{k=1}^{K} v_i^k[a]v_j^k[b]\delta_i[a]\delta_j[b] - \sum_{k=1}^{K}\sum_{i,a}(v_i^k[a]\delta_i[a])^2\right) \tag{5}$$

$$= \frac{1}{2}\left(\sum_{k=1}^{K}\left(\sum_{i,a} v_i^k[a]\delta_i[a]\right)^2 - \sum_{k=1}^{K}\sum_{i,a}(v_i^k[a]\delta_i[a])^2\right). \tag{6}$$

In the previous equation $\delta_i[a]$ is the Kronecker symbol, which equals 1 is the amino-acid in the i-th position is equal to a, and zero otherwise. This has solved half of the problem, in the sense that we can now compute every term of the pseudo-likelihood summation in linear time, but again if we naively compute each term of the summation independently, we again end up with a cost of $\mathcal{O}(N^2)$. Logically though the different terms of the pseudo-likelihood have a lot of shared structure, hence we can hope to recover a linear cost. Indeed, using Bayes' theorem, we have that

$$p(s_p|s_{-p}) = \frac{p(s)}{\sum_c p(s_p = c, s_{-p})} = \frac{\exp\{-E(s)\}}{\sum_c \exp\{-E(s_p = c, s_{-p})\}}. \tag{7}$$

we want to compute efficiently the energies and the calculate the logarithm of the loss. For the moment we will discard the fields contribution to the energy E as its cost is linear and needs no careful implementation. The crucial part is that we should do minimal effort to calculate $E(s_p = c, s_{-p})$ once we have already calculated $E(s)$. To do this we note that, labelling as δ_p^c the indicator variable for the mutates sequence having amino acid c in position p, we have that

$$\sum_{i,a} v_i^k[a]\delta_i^c[a] = \sum_{i\neq p,a} v_i^k\delta_i[a] + v_p^k[c] = \sum_{i,a} v_i^k\delta_i[a] - \sum_a v_p[a]\delta_p[a] + v_p^k[c]. \tag{8}$$

As we can see the first term is common to all energies, hence we can calculate it just once, while the other have a smaller cost. The thing to remember is that for any $p(s_p|s_{-p})$ we have to calculate all the energies $E(s_p = c, s_{-p})$ for all the dictionary letters c, and the we have to iterate across all positions p to get the final loss.

Notice, that in case we also insert a quadratic penalty, we also have to make sure to calculate the latter in $\mathcal{O}(N)$. This though is rather simple, in fact one can just observe the following

$$\sum_{i,j,a,b} J_{i,j}[a,b]^2 = \sum_{k,k'}\left(\sum_{i,a} v_i^k[a]v_i^{k'}[a]\right)^2 - \sum_{k,k'}\left(\sum_i\left(\sum_a v_i^k[a]v_i^{k'}[a]\right)^2\right), \tag{9}$$

A.2 Experiments details

A.2.1 Second order reconstruction

The experimental procedure for both datasets and can be organized in the following steps:

1. We filtered the structures, keeping only those for which the MSA generated by MMseqs2 has at least $2k$ sequences. This because since we need to compute covariances, small MSAs could give very biased benchmarks, leading to wrong conclusions.

Hypertuning results						
Model	Dropout	B	M	K	(λ_J, λ_h)	lr
ArDCA	0.1	8	32	48	(3.2e-6, 5.0e-5)	3.4e-4

Table 2: Parameters selected arDCA by hyperparamter optimization

2. For every sequence in the filtered dataset, we generate $10k$ syntetic samples for all of the three models under consideration. To generate the samples from Potts we leverage the efficient library bmDCA, for ESM-IF1 we used the built in sampler(with some changes for speed improvements) and for ArDCA we built our own sampler. For bmDCA, we ran 10 parallel chains and then pooled the results, for more details on the sampler parameters refer to the supplement.
3. For ESM-IF1, given the samples, we re-aligned the samples using the full MSA to get a fair comparison. This because, while arDCA and Potts have seen many gaps during training, ESD-IF1 has never seen one since he focuses only on the native sequence. Hence he will produce un-aligned sequences which seldomly have gaps. To align we used the PyHMMER library.
4. Given the samples, we compute the covariance matrices of the generated samples and the one of the true MSA. We then compute the Pearson correlation between the flattening of the two.

A.2.2 Melting temperature prediction

For both the structure and superfamily test dataset, we take one representative for every CATH superfamily in it. As in the iso-structure experiment, We then generate 50 proteins at different hamming distance from the native sequence for every structure and every model. We also get 100 sequences from the structure's MSA as a comparison.

In order to predict T_m, DeepStab [19] needs as input not only the primary structure, but also the conditions of the TPP experiment(ether *lysate* or *cellular* and the optimal growth temperature of the organism under consideration. For the former in our experiemnts we set the TPP enviroment to *lysate*, while for the latter, since in our sythetic design we do not know the optimal growth temperature of the organism, we run the prediction at four different optimal growth temperatures $\{0, 12, 25, 37\}$. The choice of those values was driven by the fact that the first three cover the most common outside environmental conditions over the year, and hence should adequately represent the protein diversity of most heterotermic organism. On the other hand, 37 represents homeothermic organisms(humans, mice, E coli, Bacillus), which are the organisms DeepStab has seen most of during training, and hence where he should be more accurate.

A.3 Hyper-tuning details

To train select some hyperparameters for arDCA we relayed on the library Optuna. The parameters we optimized for where Adam's learning rate(lr), the rank of the approximation for the Couplings matrix $J(K)$, the penalties for the couplings and the fields (λ_J, λ_h), the batch size used for training(B), the batch MSA size used in the loss(M) and the value of dropout. To sample parameters we use the TPESampler, and we allowed for median pruning of bad trials to improve the speed of out hyper-tuning. We gave optuna 50 trials of 90 epochs each(irrespective of batch size), while as a selecting metric we used the average of the pseudo-likelihood on the structure and superfamily test dataset. In the table below we recap the parameter values selected by the hyper-tuning

We actually applied an identical strategy for those same parameters for the Potts model, but there the hyper-tuned values did not perform better(actually slightly worse) than those reported in the Methods 2 section. It is definitely something we would like to investigate further in subsequent research, but seemed out of scope for the current work.

A.3 Supplementary: Generating Interacting Protein Sequences using Domain-to-Domain Translation

Source Family	Target Family	N_{in}	N_{out}	M_{train}	M_{val}	d_{med}
PF00689	PF00122	184	223	4460	488	62
PF00289	PF02786	112	213	4502	495	35
PF02785	PF00289	110	112	4396	490	24
PF00004	PF07724	134	173	4886	1118	21
PF00006	PF02874	215	71	5370	464	15
PF02785	PF02786	110	213	4542	494	35
PF00207	PF07677	94	94	552	138	10
PF00207	PF01835	94	98	1027	241	43
PF08264	PF00133	154	604	5846	501	110
PF00501	PF13193	425	76	17050	498	120
PF01591	PF00300	225	196	760	189	31
PF08240	PF00107	111	132	16903	500	38
PF02770	PF02771	99	115	12048	499	33
PF00441	PF02771	152	115	12703	501	38
PF01842	PF13840	69	67	851	190	19
PF08545	PF08541	82	92	2548	556	35
PF00441	PF02770	152	99	12850	497	26
PF00005	PF08402	139	78	4933	1066	49
PF00005	PF08352	139	67	5751	1243	40
PF00664	PF00005	276	139	15889	3473	106
PF03171	PF14226	103	120	5624	1317	29
PF12780	PF12781	270	222	1498	361	37
PF12775	PF12780	274	270	1632	386	39
PF07724	PF10431	173	83	7315	1649	25
PF00690	PF00702	71	212	4450	486	63
PF00690	PF00122	71	223	5356	484	37
PF00004	PF10431	134	83	4685	1088	25
HK	RR	64	112	4086	1021	33

Table 1. List of the pairs of domains used in this dataset. N_{in} and N_{out} are the length of the input domain and the target domain. M_{train} and M_{val} and the size of the training and validation dataset.

Appendix for: Generating Interacting Protein Sequences using Domain-to-Domain Translation

Appendix A Datasets

In this section, we give details about the 27 family pairs used to measure the performance of the different models. The quantities N_{in} and N_{out} are the domain length of the source family and the target family, M_{train} and M_{val} the size of the training set and validation set and d_{med} is the median distance of a sequence in the validation set to the training set. This distance was used as a cutoff for distinguishing the matching performance for sequences close or far from the training set, which are denoted by $\mathcal{M}_{Close}$ and $\mathcal{M}_{Far}$.

Appendix B Methods and Models

B.1 Transformer

The translation model we use is matching closely the original Transformer model from Ref. (Vaswani *et al.*, 2017), featuring an encoder-decoder architecture. While we refer to this work for more details, we review here the key components. Sequences from the source family are encoded by the encoder and used as the input for the decoder. The source sequence is processed through alternating blocks of self-attention and linear layers. The same is done for the already translated part of the target sequence, while the part of the target sequences not yet decoded is masked. Typical vocabulary sizes in NLP are in the order of 10^4 to 10^5, while in our case we have a vocabulary $\mathcal{V}$ is composed of 21 tokens, corresponding to 20 amino acids and an alignment gap symbol.

The input embedding is composed of two parts, one for the amino acid identity and one for the position in the sequence. We learn a dictionary W, mapping each of the 21 symbols to a vector of dimension d_{model}. The sequence position is embedded as a vector PE, calculated as

$$\begin{aligned} PE_{(i,2k)} &= \sin\left(\frac{i}{1000^{\frac{2k}{d_{model}}}}\right) \\ PE_{(i,2k+1)} &= \cos\left(\frac{i}{1000^{\frac{2k}{d_{model}}}}\right) \end{aligned}, \tag{8}$$

where i is the position in the sequence and k is the dimension in the embedding vector. The embedding of a sequence is then taken as the sum of the amino acid and positional embeddings. The embedded amino acid sequences are then passed to the encoder, mapping them to a latent representation $z = (z_1, \ldots, z_n)$. This latent representation is then passed to the decoder that predicts the interaction partner sequence.

The decoder implements an auto-regressive distribution

$$P(a_i|z, a_{<i}), \tag{9}$$

defining the probability of the i^{th} amino acid in the interaction partner sequence given the preceding amino acids in the interaction partner $a_{<i}$ and the hidden representation z of the input sequence B. During training, we use the true amino acids for $a_{<i}$, while during sampling we sample the sequence A sequentially.

Attention Mechanism Both the encoder and decoder use self-attention mechanisms and the decoder also the cross-attention mechanism.

Following (Vaswani *et al.*, 2017), we define the attention operation as

$$Attention\,(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_{model}}}\right)V, \tag{10}$$

where Q is the query matrix, K is the key matrix, V is the value matrix and d_{model} is the dimension of the keys, which we will define below.

For each element of the query Q we compute its similarity with the different values of the keys K. This yields weights used to compute a weighted average of the value V.

The output of the i^{th} attention head, called $head_i$, is calculated as

$$head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right) \tag{11}$$

We then linearly combine the different heads, resulting in

$$Multihead\,(Q, K, V) = (head_1, \ldots, head_h)\,W^O. \tag{12}$$

In the self-attention layers, the keys, queries, and values are calculated from the same input. In this case, $K = Q = V$.

In cross attention, the keys and values are based on z and the queries on the intermediate decoder representations. In the results presented in this paper, we only used models with a single head.

Transformer Architecture The complete architecture is represented in Fig. B.1 and is based on stacking encoder and decoder blocks in the two parts. At the end of these blocks, linear with dimension d_{ff} and residual connections to the input of the blocks are added.

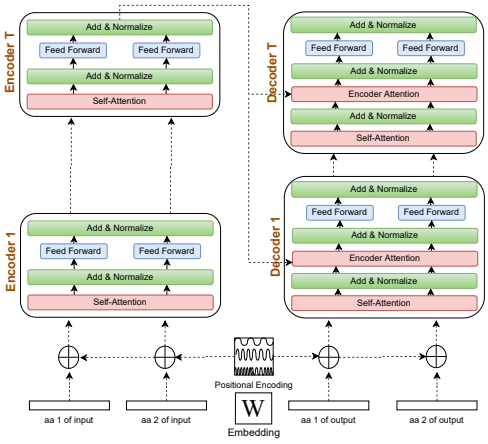


Fig. B.1. Transformer architecture for an input and output protein of length 2 and T layers: W is an embedding matrix, matching each amino acid to a vector of size e. The architecture here represents proteins of length 2 for simplicity, but the encoder and decoder can handle inputs of arbitrary length. The positional encoding uses sine functions of different frequencies to generate an embedding of the protein position.

The code is based on the PyTorch implementation of the Transformer https://github.com/pytorch/pytorch.

B.2 arDCA Baseline

As a baseline, we use the recently introduced arDCA (Trinquier *et al.*, 2021), which is an efficient autoregressive model for protein sequences. or an amino acid sequence $A = (a_i, ..., a_N)$ of length N, arDCA defines the conditional probability $P(a_i|a_{i-1}, ..., a_1)$ as

$$P(a_i|a_{i-1}, ..., a_1) = \frac{\exp\left\{h_i(a_i) + \sum_{j=1}^{i-1} J_{ij}(a_i, a_j)\right\}}{z_i(a_{i-1}, ..., a_1)}, \tag{13}$$

with $z_i(a_{i-1}, ..., a_1) = \sum_{a_i} \exp\{h_i(a_i) + \sum_{j=1}^{i-1} J_{ij}(a_i, a_j)\}$ being a normalization factor. The parameters h depend on a single position and the amino acid found at that position and the parameters J on pairs of positions and the two amino acids found at that pair of positions.

The probability of a sequence can be computed using the decomposition

$$P(a_1, ..., a_L) = P(a_1) \cdot P(a_2|a_1) \cdots P(a_L|a_{L-1}, ..., a_1), \tag{14}$$

which is tractable. Training can be done using standard convex optimization methods.

For our purposes, we concatenate the source protein B and the target protein A into a single sequence during training. During evaluation, we just need the conditional probability $P(A|B)$, which we calculate using

$$P(a_1, ..., a_{N_{out}}|B) = P(a_1|B) \cdot P(a_2|B, a_1) \cdots P(a_{N_{out}}|B, a_{N_{out}-1}, ..., a_1)\,. \tag{15}$$

We also added an L2 regularization on the parameters h and J. During our experiments, we used the regularization parameters communicated by the authors ($\lambda_h = \lambda_J = 0.0001$).

B.3 RITA

Rita is a decoder-only Transformer without conditioning information. It means that it only uses the decoder part of B.1, where the encoder-decoder attention layer has been deleted. This defines a generic autoregressive model. We used the Rita L model composed of Large $680M$ parameters, a model dimension of 1536, and 24 layers. This huge model was then trained on the UniRef-100 database. Moreover, we should note that the training was done in both directions. This explains why we present both scores when evaluating the performance of Rita finetuned. For finetuning Rita we used a batch-size of 6 and the Adam optimizer on the full-length sequence of the proteins in our train-set. Sequences were passed in both directions. Every Validation loss was computed every 200 gradient updates, and the best-performing model was kept for the experiments shown in the paper.

Rita is a language model based on the decoder of the original Transformer model in Fig.B.1. This means it does not use encoder-decoder attention and implements a generic unconditioned autoregressive sequence model. In our experiments, we use Rita L, which has $680M$ parameters, a model dimension of 1536, and 24 layers. The model we used for finetuning was pre-trained on Uniref-100 predicting in both the natural and the reverse direction of the protein sequences. We used a batch-size of 6 and the Adam optimizer for finetuning on the full-length sequences (both directions) on our datasets. We calculated the loss on the validation every 200 gradient updates on a total of 4000 gradient updates. We used the best-performing model for the experiments shown in the paper. The number of steps to finetuned each model varies between families but usually stands around 1000, way before the end of our training. To make the loss comparable we only took into consideration the positions that were match state for the Pfam HMM of the domain.

B.4 Joined Transformer

Here we detail more the performance of training a single Transformer (the "joined Transformer") on all the pairs in order to see if we can benefit from transfer learning between pairs. To do so we joined all the datasets except 4 in one single dataset. The left-out pairs are used for evaluating if there is transfer learning to unseen pairs. To make the comparison fairer we replaced the <SOS> token with a specific token for the pair (the idea is to tell to the Transformer the task it has to perform). To make this model also compatible with unseen pairs (and unseen specific tokens) we replaced this token only 50% of the time in the training examples, 100% of the time in the validation sets of seen pairs, and never for the unseen pairs. The training takes significantly longer. We trained in parallel for two days on 12 GPU whereas the smaller models were trained on a single one for usually less than a day.

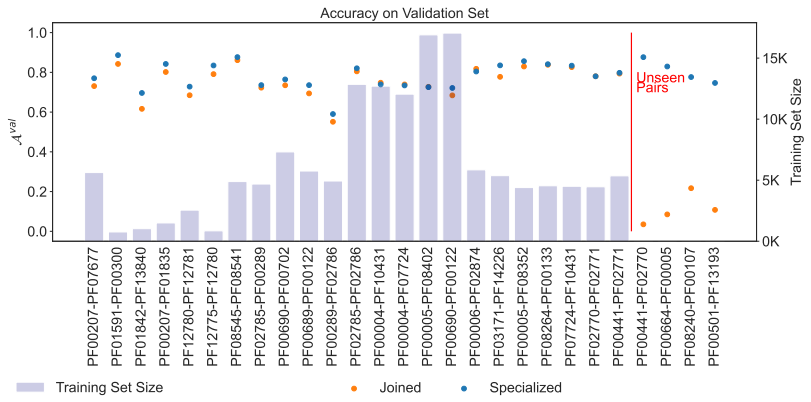


Fig. B.2. Accuracy $\mathcal{A}^{val}$ on the validation set for the shallow Transformer, and the Joined Transformer. The families are ordered by training set size.

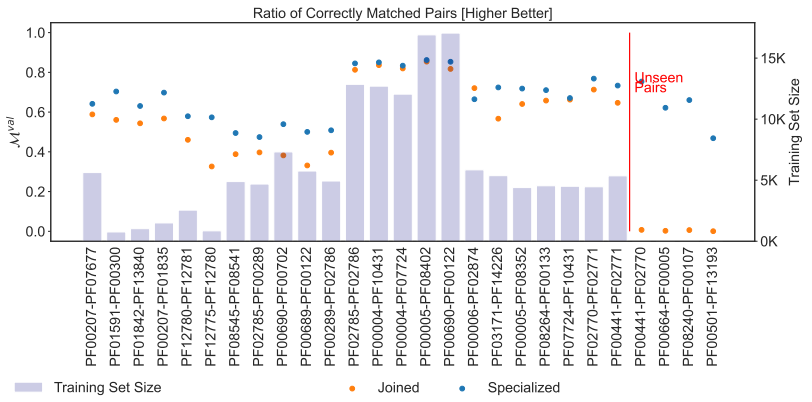


Fig. B.3. True positive rate for matching on the validation set for the shallow Transformer, and the Joined Transformer. The families are ordered by training set size.

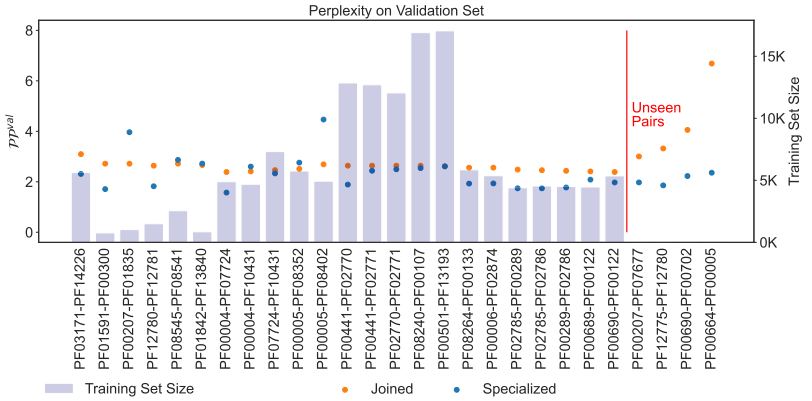


Fig. B.4. Perplexity $\mathcal{PP}^{val}$ for the shallow Transformer, the Joined Transformer on the validation set. The families are ordered by training set size.

The results seem to show very little transfer learning between pairs. Probably such an effect could only appear when training on thousands of pairs. Moreover, the spirit of this paper is intended to fit the line of work of domain-specific models like in Potts, VAE, RBM Hawkins-Hooker *et al.* (2021b); Tubiana *et al.* (2019); Russ *et al.* (2020). We intend to guide specific design task when one wants to redesign a specific domain/protein to increase its fitness for a desired task.

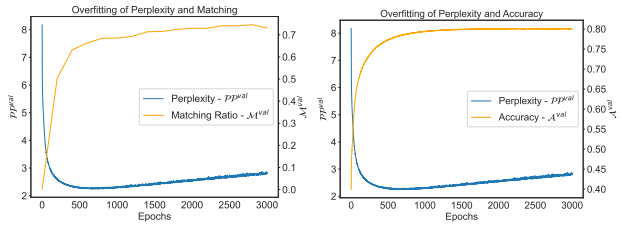


Fig. C.1. Evolution of the perplexity and the fraction of correctly matched pairs (a) on the validation set and (b) the accuracy during training for PF013171-PF14226 for the large Transformer.

B.5 Training Time

In this section, we present a table with the training time of our models using a single Nvidia V100 GPU. The first column refers to the family pair, the second to the training time of the shallow Transformer, the third one to the large Transformer with entropic regularization, and the last one to the large Transformer without entropic regularization.

Pair	Runtime Shallow	Runtime Renyi	Runtime Large
PF00289_PF02786	0 days 10:13:10	3 days 17:10:52	0 days 15:10:43
PF02785_PF02786	0 days 10:20:21	2 days 10:45:01	0 days 11:24:58
PF02785_PF00289	0 days 09:50:34	2 days 02:05:06	0 days 15:45:50
PF00004_PF07724	0 days 19:54:55	2 days 15:57:20	0 days 23:05:37
PF00006_PF02874	0 days 12:00:52	2 days 07:07:35	0 days 12:28:15
PF00207_PF07677	0 days 01:44:11	0 days 06:14:45	0 days 01:09:31
PF00207_PF01835	0 days 03:30:15	0 days 11:15:13	0 days 02:17:15
PF08264_PF00133	0 days 18:21:48	6 days 05:37:06	0 days 19:24:28
PF00501_PF13193	1 days 16:47:12	12 days 12:27:26	1 days 07:23:54
PF01591_PF00300	0 days 03:20:45	0 days 19:01:09	0 days 02:23:11
PF08240_PF00107	1 days 13:11:32	6 days 20:56:44	1 days 18:54:09
PF02770_PF02771	1 days 02:09:05	6 days 01:37:54	1 days 07:02:04
PF00441_PF02771	1 days 01:16:55	10 days 18:35:05	1 days 08:18:41
PF00441_PF02770	1 days 02:20:40	6 days 01:56:31	1 days 11:05:05
PF01842_PF13840	0 days 02:38:04	0 days 10:54:23	0 days 02:37:06
PF08545_PF08541	0 days 07:48:42	1 days 04:28:05	0 days 05:54:03
PF00005_PF08402	0 days 11:30:09	1 days 19:43:05	0 days 12:58:03
PF00005_PF08352	0 days 13:42:05	2 days 13:55:30	0 days 16:48:21
PF00664_PF00005	2 days 01:30:23	10 days 19:21:16	2 days 02:02:05
PF03171_PF14226	0 days 22:10:30	4 days 04:10:56	0 days 21:30:18
PF12780_PF12781	0 days 07:10:21	1 days 00:33:05	0 days 03:09:31
PF12775_PF12780	0 days 08:13:18	2 days 05:13:49	0 days 03:43:15
PF07724_PF10431	0 days 18:03:28	3 days 14:18:21	1 days 08:33:09
PF00690_PF00702	0 days 09:55:26	2 days 16:45:33	0 days 10:26:16
PF00690_PF00122	0 days 12:05:13	3 days 04:58:34	0 days 16:48:12
PF00689_PF00122	0 days 10:37:22	5 days 08:58:05	0 days 17:20:30
PF00004_PF10431	0 days 13:00:08	2 days 19:41:44	0 days 18:51:31
HKRR	0 days 9:03:08	2 days 15:41:14	0 days 15:38:34

Appendix C Regularization

C.1 Dropout and Weight-Decay Benchmark

In this section, we show learning curves related to overfitting behavior and regularization.

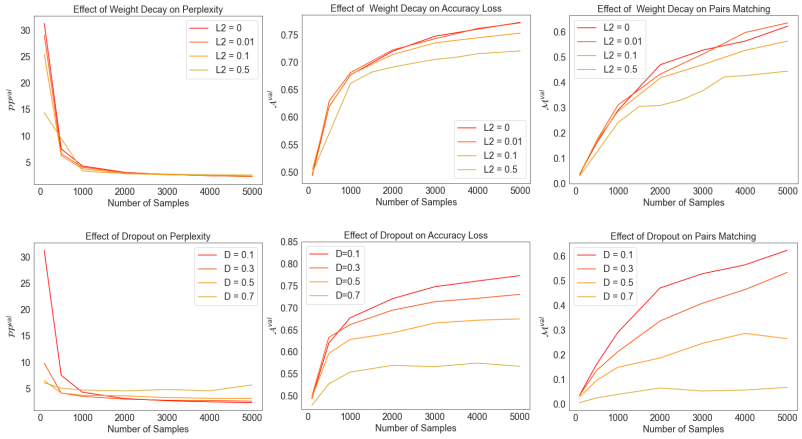


Fig. C.2. Performance of the Transformer on PF013171-PF14226 for a varying number of training examples: Left plots show the perplexity $\mathcal{PP}^{val}$, center plots show the accuracy $\mathcal{A}^{val}$, right plots show the fraction of correctly matched pairs $\mathcal{M}^{val}$. All values are for the validation set.

C.2 Entropic Formulation of The Regularization

There is a relation between the entropic regularization and the Rényi entropy: For a number of sampled sequences S sufficiently large we can rewrite the regularization term

$$\begin{aligned} R_{ent}(A_i, B_i) &= \log P(A_i|B_i) - \log S - \log\left(\frac{1}{S}\sum_{k=1}^{S} P(A_{i,k}|B_i) + \frac{1}{S}P(A_i|B_i)\right) \\ &\approx \log P(A_i|B_i) - \log S - \log\left(\mathbb{E}_{A\sim P(A|B_i)}[P(A|B_i)]\right) \\ &\approx \log P(A_i|B_i) - \log S - \log\left(\sum_A p^2(A|B_i)\right), \end{aligned} \tag{16}$$

where the last sum is over all possible sequences A. The first term in the last line is equivalent to the standard loss and can be absorbed there. The second term is a constant and will not influence the gradient. The last term is the logarithm of the Rényi entropy of order 2, also called the *collision entropy*, of the distribution over target sequences conditioned on B_i.

C.2.1 Entropic Regularization Performance

This section presents the results of the large Transformer trained with $\alpha = 0.7$, and $S = 5$ in comparison with arDCA and the shallow Transformer in Fig. C.3, Fig. C.4 and Fig. C.5. The comparison with the large Transformer without regularization or with weight decay is presented in the following section, see Appendix Sec. C.3.

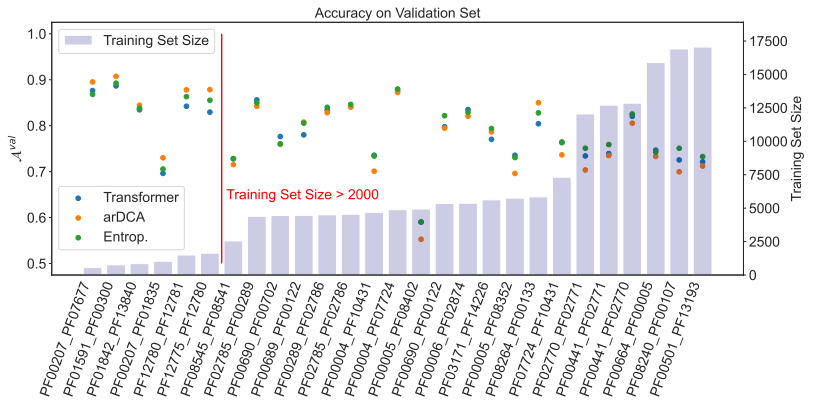


Fig. C.3. Accuracy $\mathcal{A}^{val}$ on the validation set for the shallow Transformer, arDCA, and the large Transformer with entropic regularization. The families are ordered by training set size. -For datasets below 2000 examples, arDCA is always above Transformer with an average difference of 0.02 in terms of accuracy -For datasets above 2000 examples, Transformer is below arDCA in 90.4% of cases with an average difference of 0.02 in terms of accuracy

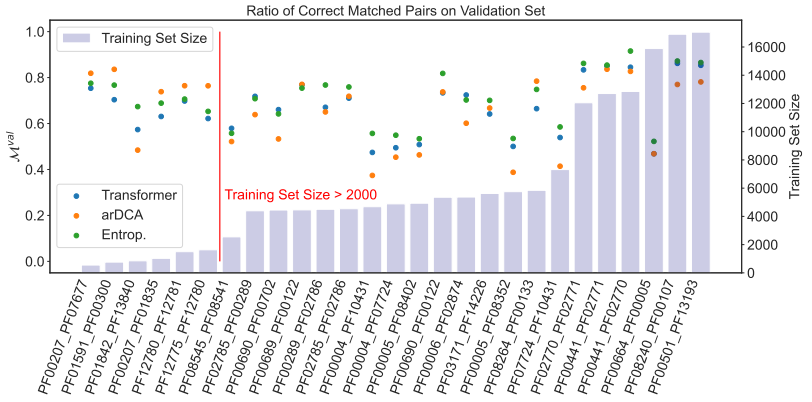


Fig. C.4. True positive rate for matching on the validation set for the shallow Transformer, arDCA, and the large Transformer with entropic regularization. The families are ordered by training set size. -For datasets below 2000 examples, arDCA is above Transformer in 83% of cases with an average difference of 0.02 in terms of matching fraction -For datasets above 2000 examples, Transformer is below arDCA in 90.4% of cases with an average difference of 0.08 in terms of matching fraction

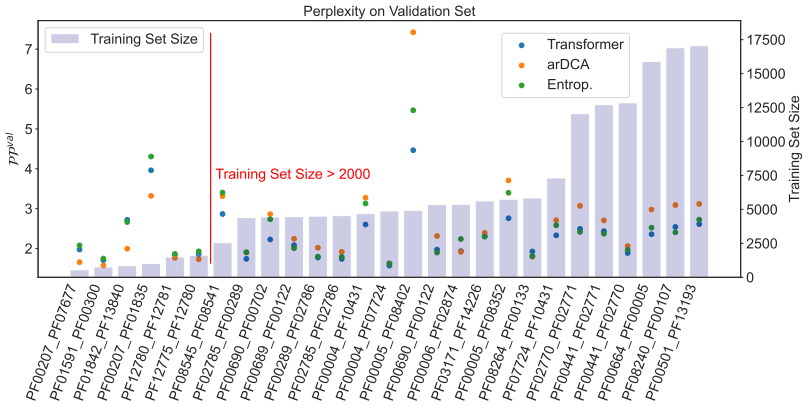


Fig. C.5. Perplexity $\mathcal{PP}^{val}$ for the shallow Transformer, the large Transformer with entropic regularization and arDCA on the validation set. The families are ordered by training set size. -For datasets below 2000 examples, arDCA is always below Transformer with an average difference of 0.42 in terms of perplexity -For datasets above 2000 examples, Transformer is below arDCA in 76% of cases with an average difference of 0.28 in terms of perplexity

C.3 Entropic Regularization Compared with Weight Decay

In this section, we compare the entropic regularization and weight decay on different metrics, averaged over all 27 families.

Entropic versus Weight Decay In this section, we compare the performance of the large Transformer on different metrics for all families individually. The resulting plots were split into different figures in order to make them fit on the pages.

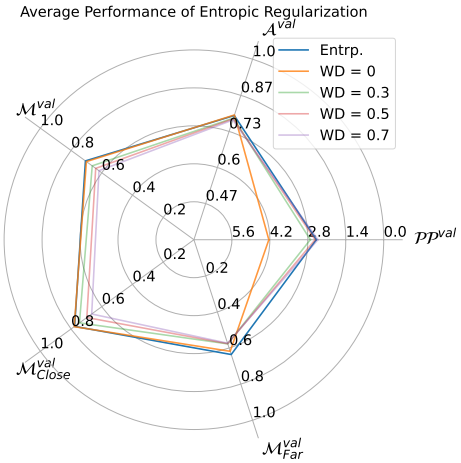


Fig. C.6. Radar plot comparing different regularization schemes, entropic (Entrp.), and weight decay (WD) with different strengths for the large Transformer. The radial direction of the perplexity $\mathcal{PP}$ is reversed in order to have the same direction for increasing performance as for the other metrics. The plot was done by averaging the metrics of all families.

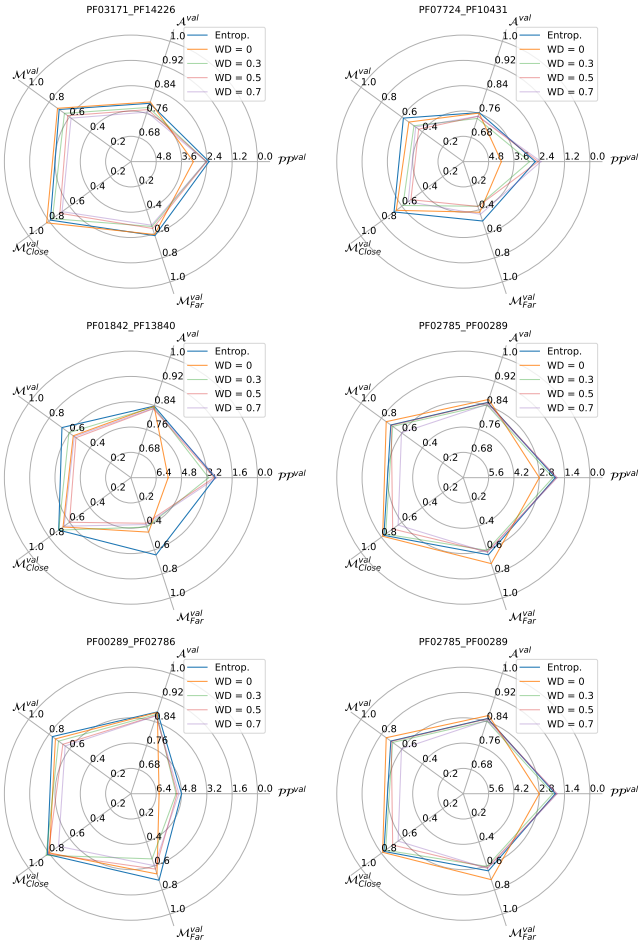


Fig. C.7. Comparing the performance of the large Transformer with entropic regularization and weight decay (WD). The radial direction of the perplexity $\mathcal{PP}$ is reversed in order to have the same direction for increasing performance as for the other metrics.

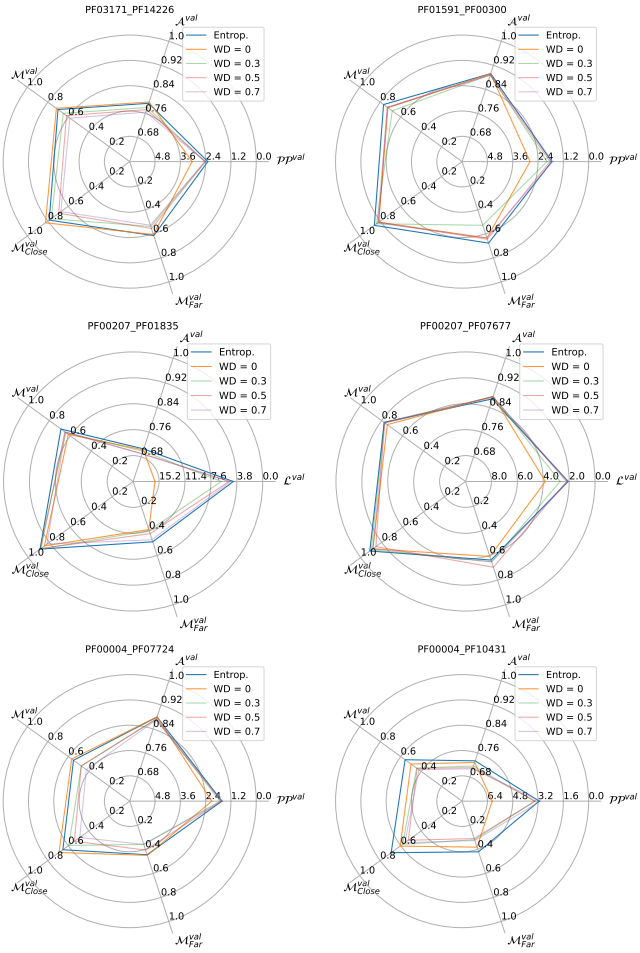


Fig. C.8. Comparing the performance of the large Transformer with entropic regularization and weight decay (WD). The radial direction of the perplexity $\mathcal{PP}$ is reversed in order to have the same direction for increasing performance as for the other metrics.

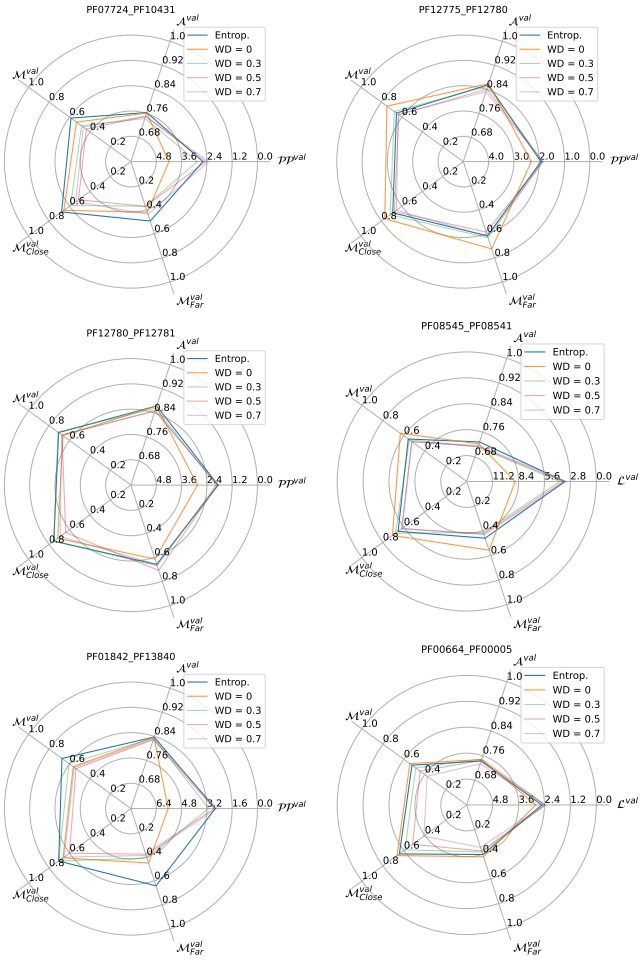


Fig. C.9. Comparing the performance of the large Transformer with entropic regularization and weight decay (WD). The radial direction of the perplexity $\mathcal{PP}$ is reversed in order to have the same direction for increasing performance as for the other metrics.

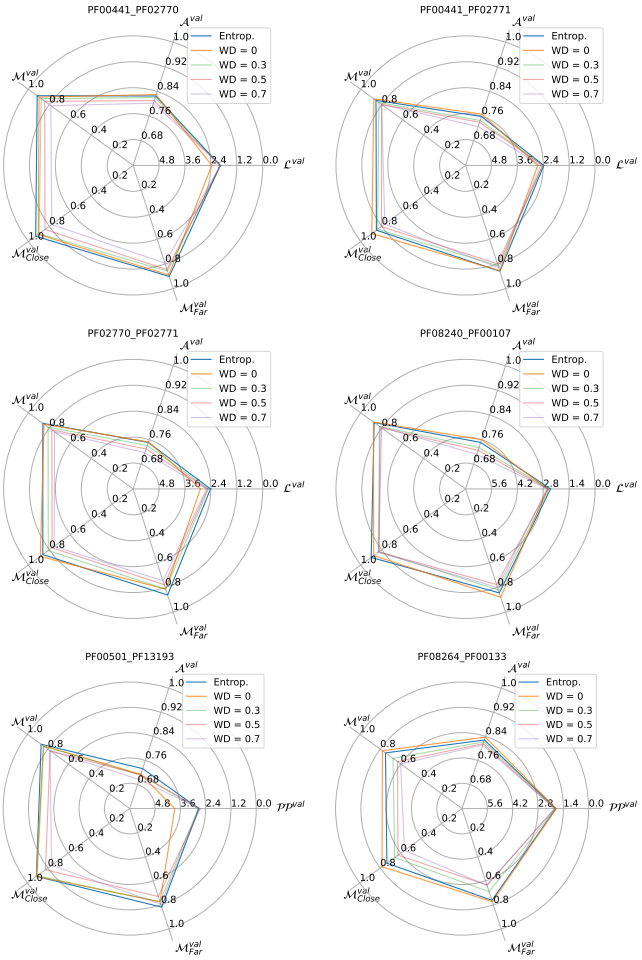


Fig. C.10. Comparing the performance of the large Transformer with entropic regularization and weight decay (WD). The radial direction of the perplexity $\mathcal{PP}$ is reversed in order to have the same direction for increasing performance as for the other metrics.

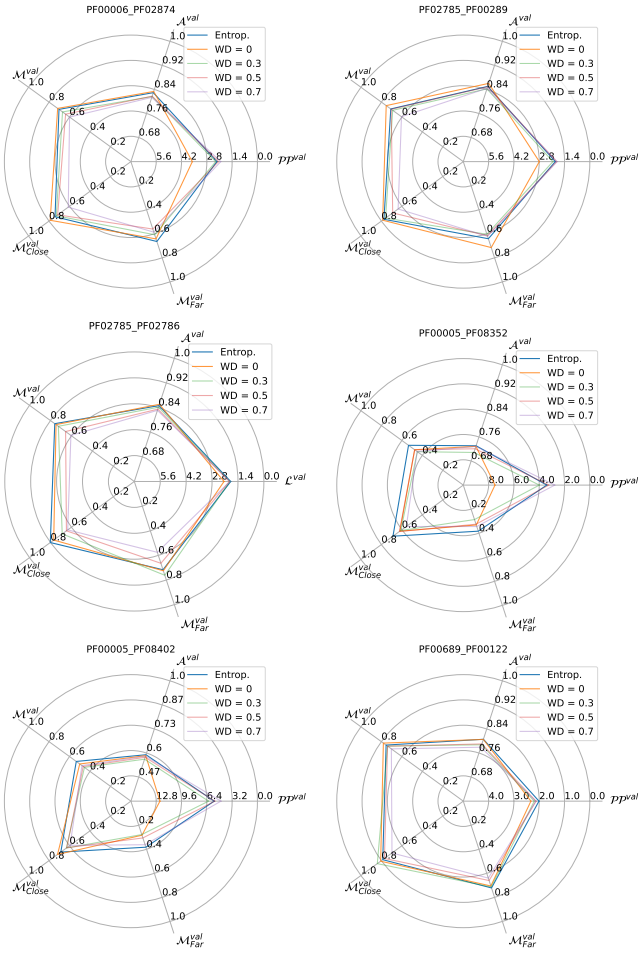


Fig. C.11. Comparing the performance of the large Transformer with entropic regularization and weight decay (WD). The radial direction of the perplexity $\mathcal{PP}$ is reversed in order to have the same direction for increasing performance as for the other metrics.

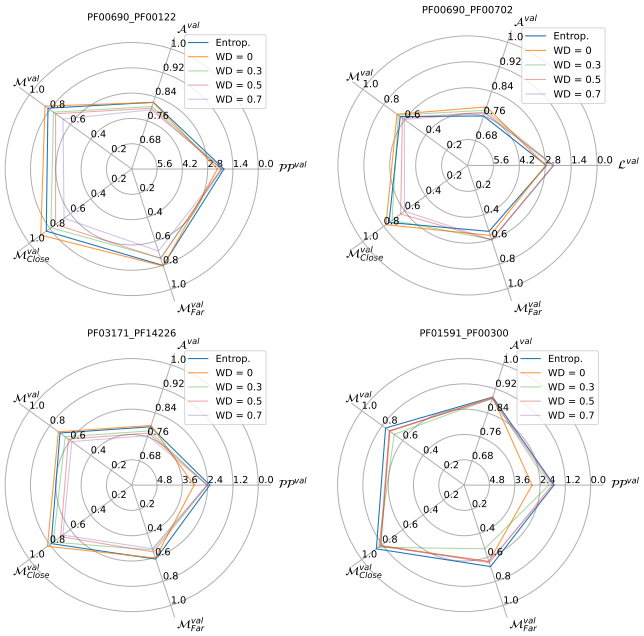


Fig. C.12. Comparing the performance of the large Transformer with entropic regularization and weight decay (WD). The radial direction of the perplexity $\mathcal{PP}$ is reversed in order to have the same direction for increasing performance as for the other metrics.

Appendix D Additional Structural Results

D.1 Structural comparison using AlphaFold

For all our computations we used the implementation of CollabFold with template search, 5 models, and 3 recycle. (Mirdita *et al.*, 2022).

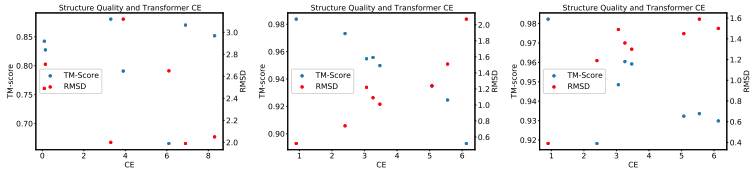


Fig. D.1. TM-Scores and RMSD values when comparing the Alphafold predicted structures of true sequences with Alphafold predicted structures of sequences where single domains have been replaced with homologous natural sequences. (Left is based on G0S4G4, which contains domains PF00004 and PF07724, which are in contact in PDB 5D4W. Homologous sequences are sampled from the validation set and inserted into the G0S4G4 sequence. Center is based on Q8ZN46, which contains domains PF00207 and PF01835, which are in contact in PDB 4U4J. Homologous sequences are sampled from the validation set and inserted into the Q8ZN46 sequence. Right is based on Q13SV4, which contains domains PF08545 and PF08541, which are in contact in PDB 4EFI. Homologous sequences are sampled from the validation set and inserted into the Q13SV4 sequence. In all of these proteins, we measure the change in structural scores and in cross-entropy in the shallow Transformer model (abscissa).

D.2 Structural Information using DCA

Direct Coupling Analysis is a group of unsupervised methods for modeling aligned protein sequences, see Ref. (Cocco *et al.*, 2018). Apart from other applications, it can be used for predicting structural contacts from MSAs.

plmDCA plmDCA is a specific method of DCA based on a pseudolikelihood approximation for training the Potts Model. In this paper we used the asymmetric version of the method from https://github.com/pagnani/ArDCA.jl with default hyperparameters. The sequences sampled from the Transformer were realigned using HMMer (Finn *et al.*, 2011).

Results per Pairs In this section, we show the contact prediction results obtained with plmDCA for the 27 families.

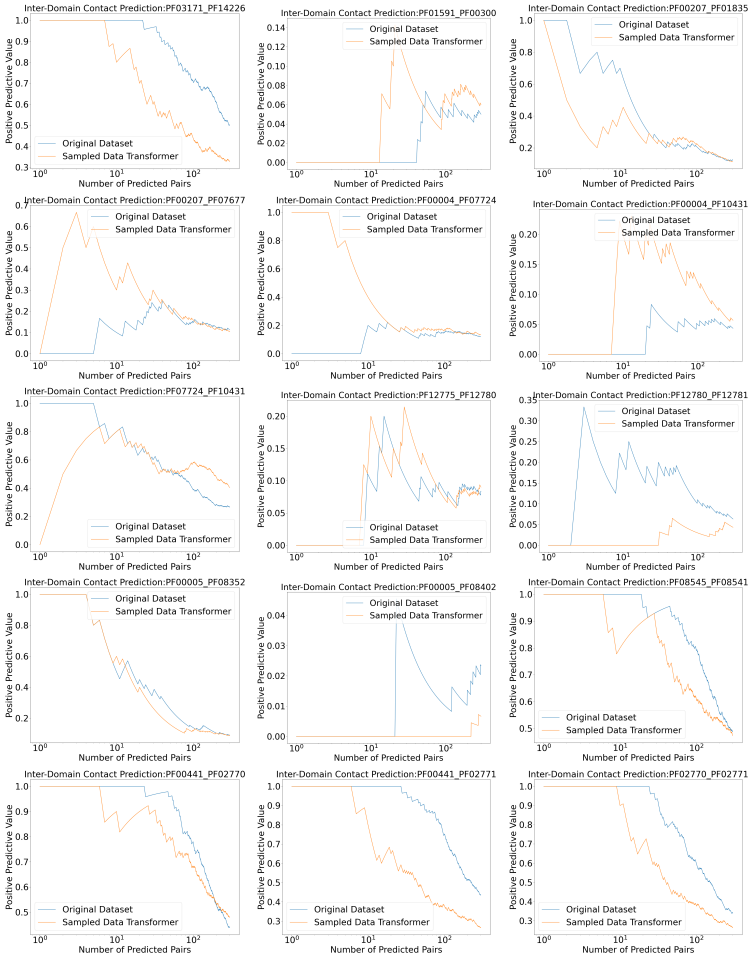


Fig. D.2. Contact Prediction using plmDCA for the original training set, and a sampled dataset from the shallow Transformer. The curves represent the Positive Predictive Value (fraction of true positives) with respect to the number of predicted contacts. To make it fit the page format, we split the results on the different families into two figures: this one and the following.

Fig. D.3. Contact Prediction using plmDCA for the original training set, and a sampled dataset from the shallow Transformer. The curves represent the Positive Predictive Value (fraction of true positives) with respect to the number of predicted contacts.

Appendix E Additional Results on Generalization

E.1 Matching Performance for Different Distances from Training Set

Here we plot the fraction of correctly matched pairs in the validation set, separated into below-median and above-median distances from the training set.

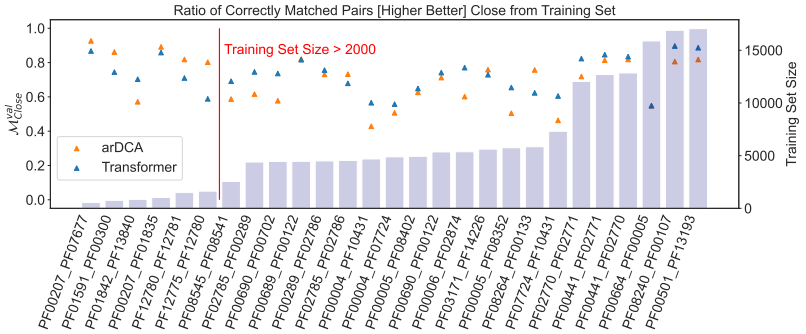


Fig. E.1. Fraction of correctly matched pairs in the validation set for the shallow Transformer and arDCA. The families are ordered by training set size. Shown are results for the 50% of sequences closest to the training set.

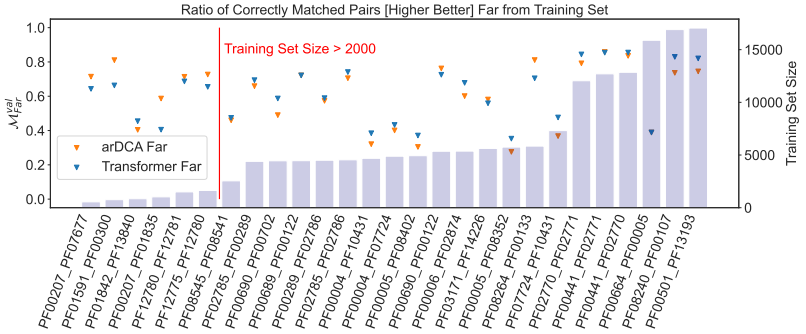


Fig. E.2. Fraction of correctly matched pairs in the validation set for the shallow Transformer and arDCA. The families are ordered by training set size. Shown are results for the 50% of sequences farthest to the training set.

E.2 Accuracy and Perplexity with Distance from Training Set

In this section, we present the results for each pair of the analysis of Sec. 4.5.

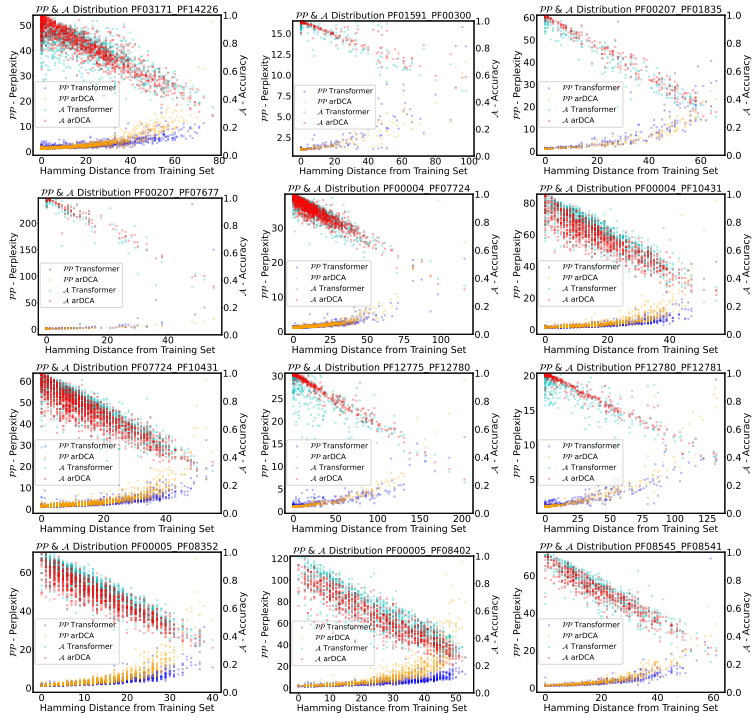


Fig. E.3. Distribution of the perplexity $\mathcal{PP}$ and the accuracy $\mathcal{A}$ of every sequence pair in the validation set with respect to their distance from the training set for the shallow Transformer and arDCA. To fit the page format, we split the results on the different families into two figures: this one and the following

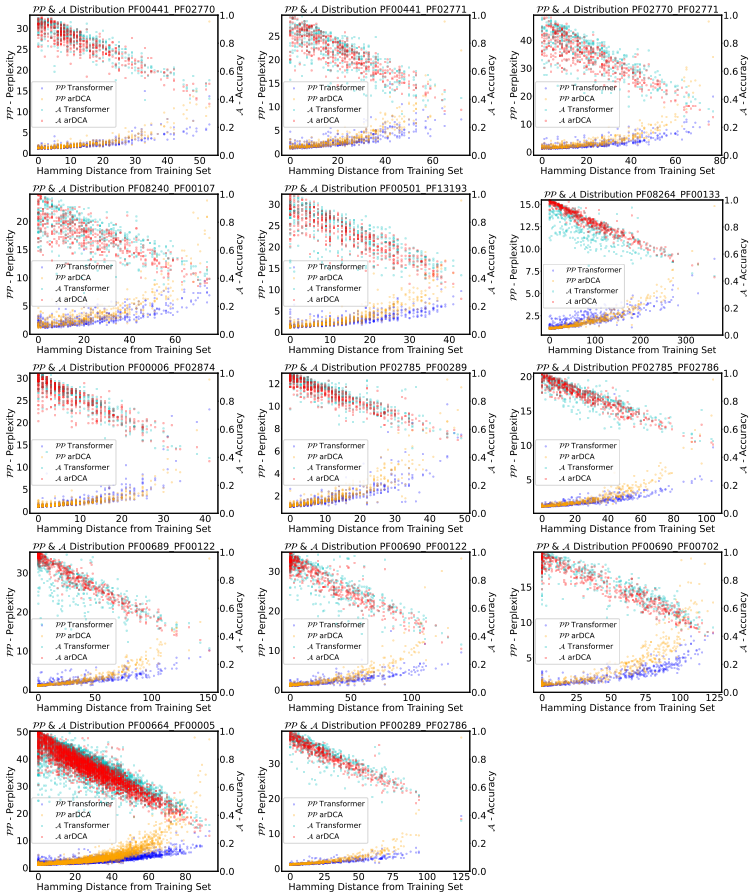


Fig. E.4. Distribution of the perplexity $\mathcal{PP}$ and the accuracy $\mathcal{A}$ of every sequence pair in the validation set with respect to their distance from the training set for the shallow Transformer and arDCA.

Appendix F Additional Matching evaluation

For each protein family, we measure the fraction of correctly matched pairs when restricting the problem to the first n sequence pairs.

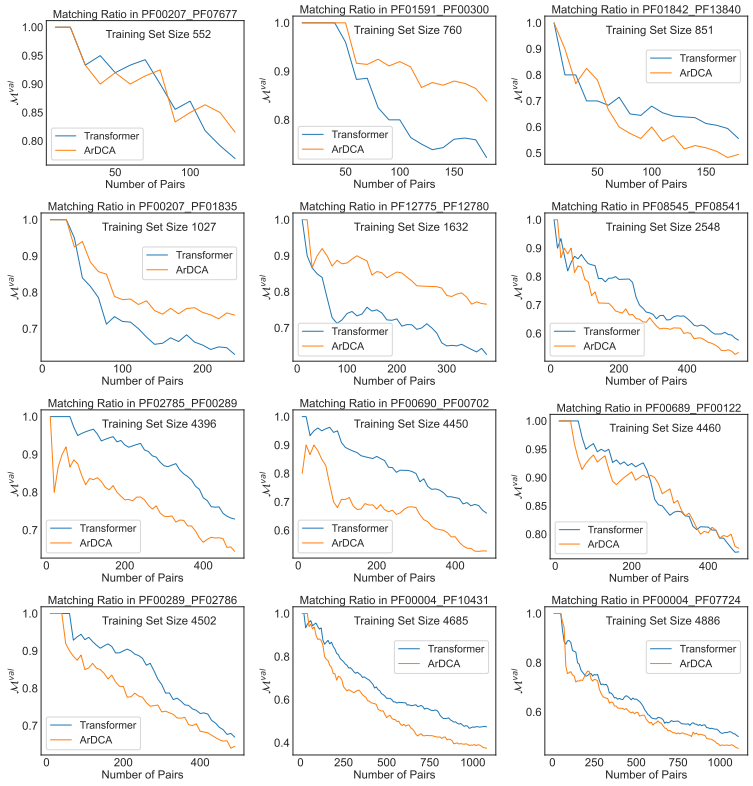


Fig. F.1. Fraction of correctly matched pairs $\mathcal{M}^{val}$ for increasing number of pairs for different families. Shown are the results for the shallow Transformer (blue) and arDCA (orange). Only a subset of the families is shown in order to save computational resources. The families are ordered according to the training set size. In order to fit the page format, we split the results into this and the next figure.

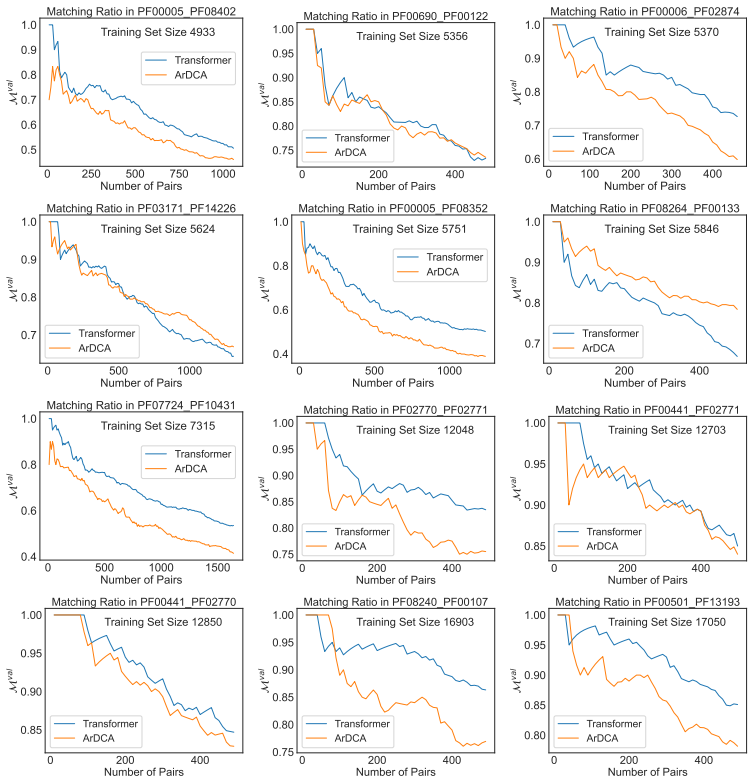


Fig. F.2. Fraction of correctly matched pairs $\mathcal{M}^{val}$ for increasing number of pairs for different families. Shown are the results for the shallow Transformer (blue) and arDCA (orange). Only a subset of the families is shown in order to save computational resources. The families are ordered according to the training set size. In order to fit the page format, we split the results into this and the previous figure.

F.1 D

F.1.1 D.3 Sequence Logo and Loss

In Fig. F.3 we show the perplexity per position for family pair PF013171-PF14226, together with the sequence logo. It is evident that biologically conserved positions correspond to lower perplexities, which is to be expected.

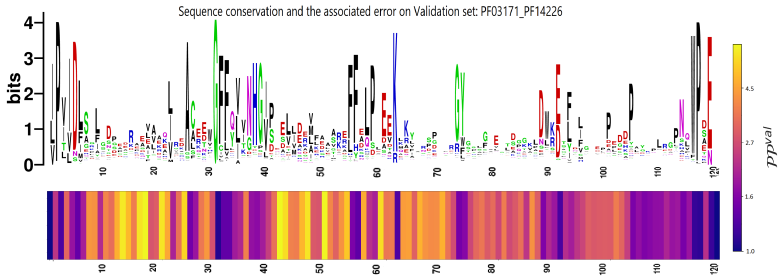


Fig. F.3. Top: Sequence logo of PF013171-PF14226 paired MSA. Bottom: Distribution of the perplexity with respect to the positions. The errors are concentrated on the most variable position, highlighting that the Transformer has understood the basic site-wise structure of the distribution.

Supplementary References

A.4 Supplementary: Tulip

APPENDICES

Appendix A: Training details

In all the examples presented in this paper, we used the following architecture: embedding dim = 128, hidden size = 128, and each encoder and each decoder have 2 layers. The MHC embedding was limited to the 50 most represented MHC. In all the examples presented in this paper, we used the following architecture: embedding dimension = 128, hidden size = 128, and 2 layers for each encoder and each decoder. The MHC embedding was limited to the 50 most represented MHC, and none in the training with unseen epitopes. For the repertoire mining experiment, we used 100 epochs in the zero shot setting. During the training we use the Adam optimizer [45] with a learning rate of 0.0001 for 100 epochs. During the finetuning process, we freeze the encoder and the embedding. We train using the same losses with adam optimizer. The finetuning is done for 40 epochs (but on much smaller dataset) keeping the loss function the same.

Appendix B: Comparison to other methods

For comparing with DLpTCR we fine-tuned our model on HLA-A02*01 in the same way as for the experiments of section II.A. For ERGO, metrics are computed on the subset of peptide that were both left out by TULIP and ERGO.

Peptide	CDR3a	CDR3b
ALIHHNTHL	CAVNSNSGYALNF	CASSQSETGDGYTF
ALIHHNTHL	CAMHRDDKIIF	CASSLAVQRPSGNTIYF
ALIHHNTHL	CVVSGVNVWGTYKYIF	CASSIESGSKQRNEQFF
ALIHHNTHL	CAVSDLNSGGYQKVTF	CASSPRDRVHEQYF
HMTEVVRHC	CAMSGLKEDSSYKLIF	CASSIQQGADTQYF
HMTEVVRHC	CAFMGYSGAGSYQLTF	CAISELVTGDSPLHF
HMTEVVRHC	CALDIYPHDMRF	CASSLDPGDTGELFF
HMTEVVRHC	CVVQPGGYQKVTF	CASSEGLWQVGDEQYF
LLGATCMFV	CAADSWGKLQF	CATSDSTGSYGYTF
LLGATCMFV	CAVNPSNQFYF	CASRGPYHNEQFF
LLGATCMFV	CVVSEEYTNAGKSTF	CASSLERLRVYSGYTF
LLGATCMFV	CAMDSSYKLIF	CASSALAGGQADTQYF
LLGATCMFV	CAAGGSYIPTF	CASSGTGGYSGANVLTF
LLGATCMFV	CAVNDYKLSF	CASSWTGANYGYTF
LLGATCMFV	CAVYSGGYNKLIF	CASSFVNTGELFF
RLARLALVL	CASMYSGGGADGLTF	CASSFFSNTGELFF
RLARLALVL	CASGGGADGLTF	CASSFLTDTQYF
RLARLALVL	CSSGGGADGLTF	CASMDLAFKQYF
RLARLALVL	CAYRSGSDGGSQGNLIF	CASSQVSGYEQYF
RLARLALVL	CAVRDDYGQNFVF	CASSPQGDNEQFF
RLARLALVL	CAVPDDAGNMLTF	CASSELPAGGTNEQFF
RLARLALVL	CAGGGGADGLTF	CASSYMGPEAFF
YLEPGPVTA	CAPGIAGGTSYGKLTF	CASSLAYSYEQYF
YLEPGPVTA	CGTETNTGNQFYF	CASSLGRYNEQFF
YLEPGPVTA	CAASTSGGTSYGKLTF	CASSLGSSYEQYF
YLEPGPVTA	CAVLSSGGSNYKLTF	CASSFIGGTDTQYF
YLEPGPVTA	CATDGDTPLVF	CASSIGGPYEQYF

TABLE S1. TCRs used in the repertoire mining tests.

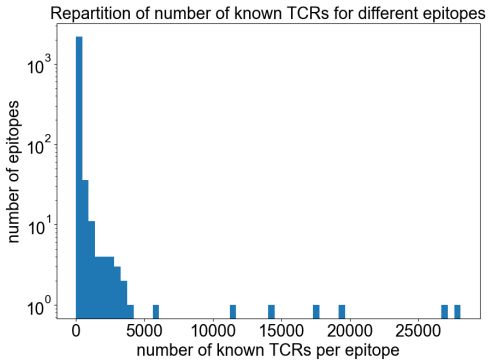


FIG. S1. For each epitope in our database, we counted the number of known T-cell receptors (TCRs) binding to that epitope. The histogram shows a strong imbalance in the dataset, where a handful of epitopes harbor a substantial number of known TCRs, while the majority of epitopes have only a limited number of associated TCRs.

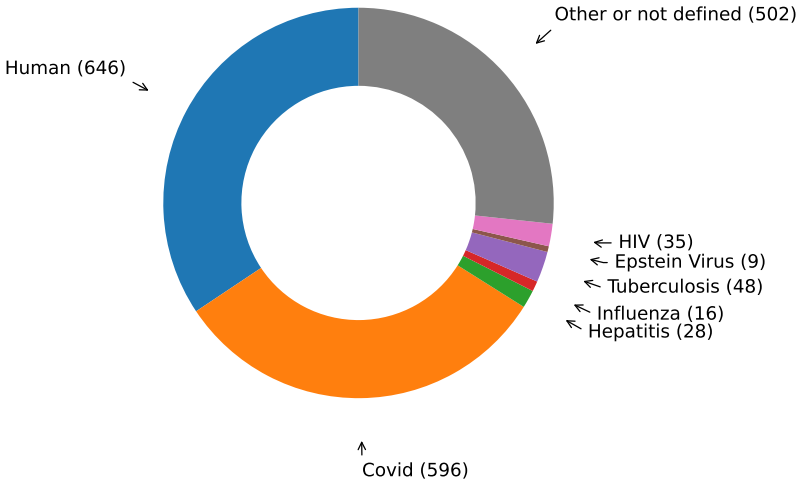


FIG. S2. Distribution of peptides by organism of origin.

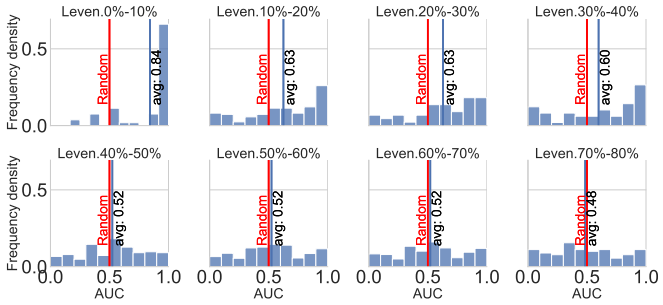


FIG. S3. Distribution of the AUCs for the different distance groups. The plot complements Fig. 3A.

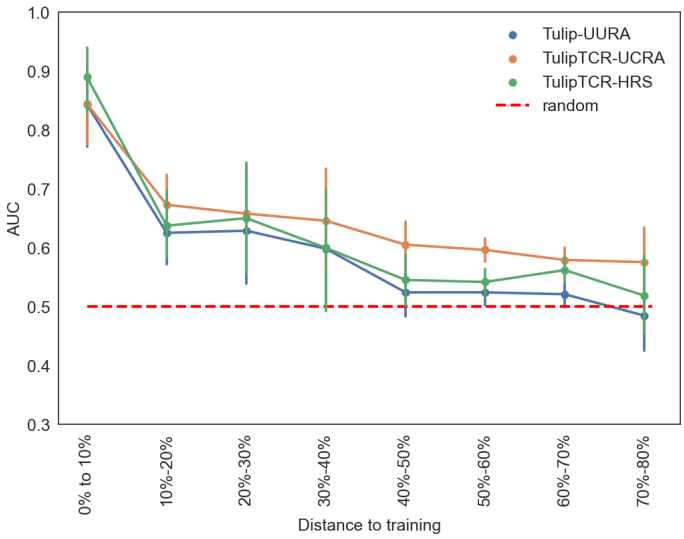


FIG. S4. Mean AUC of the different distance groups. Equivalent to Fig. 3A, but with normalized distance (Hamming distance divided by length).

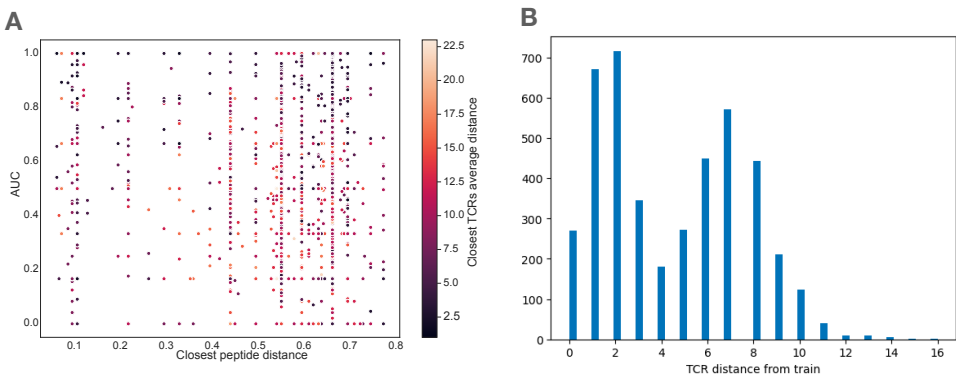


FIG. S5. A. Scatter plot of AUC of individual peptides, versus the normalized distance (Hamming distance divided by CDR3 length) to closest peptide in the training set. The color indicates the average distance between a TCR associated to the peptide of interest, and the closest TCR associated to its closest peptides. B. Distribution of Hamming distances between a sequence from the test set and its closest TCR in the train set.

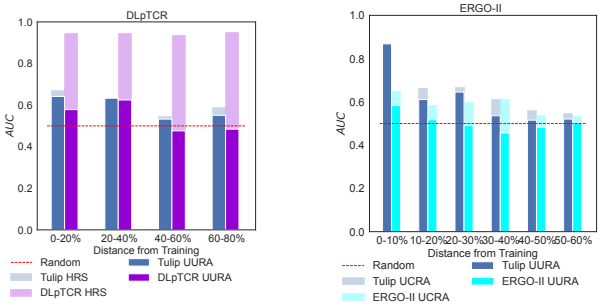


FIG. S6. Performance of DLpTCR and ERGO2 on unseen peptides (complement to Fig. 3C).

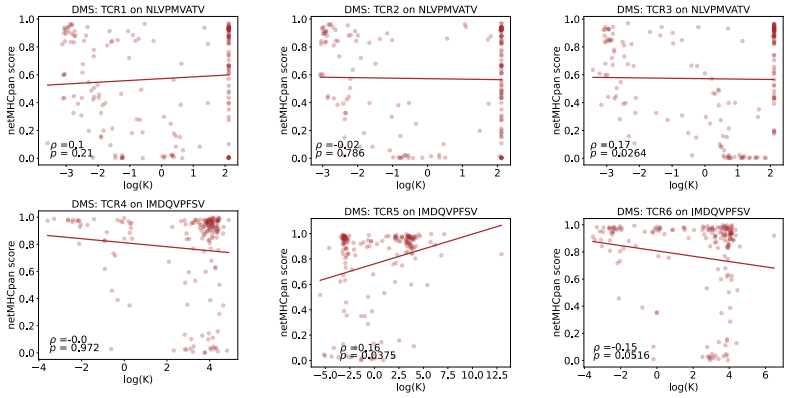


FIG. S7. Effect of single epitope mutations on the NetMHCPan [36] score ($\log P$) predicts TCR binding (dissociation constant K, in μg.ml^{-1}) measured by deep mutational scan experiments [35]. The reported ρ and p-values correspond to Spearman correlations.