

Guiding Protein Embeddings to Low-Energy States via Subspace Relaxation

Pranav Mahableshwarkar, Colin Baker, Ritambhara Singh, and Sorin Istrail

Advisor: Sorin Istrail, Reader: Ritambhara Singh

*A thesis submitted in partial fulfillment of the requirements for the degree of
Bachelor of Science with Honors in Computer Science at Brown University*



Brown University
Providence, RI
May 2025

Guiding Protein Embeddings to Low-Energy States via Subspace Relaxation

Contents

1	Introduction	2
2	Computational Models for Protein Folding	3
2.1	Computationally Simulating Protein Folding	3
2.1.1	Energy Functions	3
2.1.2	Relaxation	3
2.1.3	Exploring Conformational Spaces with Molecular Dynamics	4
2.2	Estimating Protein Conformations Using MMC	5
2.3	AlphaFold2	7
2.3.1	Model Introduction	7
2.3.2	The Evoformer Module	8
2.3.3	AlphaFold2’s Reliance on Homology	9
2.4	Analysis of AlphaFold2 Intermediate Subspaces	10
2.4.1	Individual Intermediate Embedding Subspaces	11
2.4.2	Comparison of Subspaces Across Blocks and Cycles	12
3	Subspace Relaxation Operator for Protein Structure Predictions	14
3.1	Approach	14
3.2	Dataset Generation	14
3.3	Architecture	15
3.3.1	Conditioning on Energy Gradients	15
3.3.2	Architecture Components	16
3.3.3	Extending the AlphaFold Loss	17
3.3.4	Subspace Relaxation Model	18
3.4	Training Procedure and Infrastructure	19
3.4.1	GPU Infrastructure and Gradient Smoothing	19
3.4.2	Hyperparameter Tuning and Weights	20
3.5	Results	21
3.5.1	Model Selection	21
3.5.2	Force Conditioning Enables SRO Predictive Ability	21
3.6	Future Integration into AlphaFold2 Inference	23
4	Conclusion	23
5	Acknowledgments	24

1 Introduction

Proteins drive essential cellular reactions, gene regulation, immune responses, and more. In a broad sense, students are often taught that structure defines function. Yet, when considering the exponentially large number of possible amino acid configurations, it is incredible that a peptide sequence collapses to its designed function within seconds. This contradiction is highlighted in Levinthal’s paradox [1], and has driven decades of research interrogating the mechanisms behind protein folding. As we move towards more advanced computational drug design and molecular discovery, there is a need for methods that can make accurate 3D structure predictions using sequence information.

First introduced in 2021, AlphaFold2 is a Nobel Prize-winning protein structure prediction method proposed by Google’s DeepMind, leveraging a complex deep learning architecture and a vast dataset [2]. While AlphaFold2 - and its current successor in AlphaFold3 [3] - achieve undeniable success in this task, the model, like many other existing methods, fails to explicitly account for driving forces behind protein folding. Instead, the model relies on homologous structures to build an implicit understanding of amino acid relationships. As shown in the AlphaFold studies, this design leads to unreliable structural predictions in the absence of sufficient homology data.

In this work, we propose an integration of molecular dynamics simulations into protein structure prediction, leveraging what we call the **Subspace Relaxation Operator**. Following a Metropolis Markov Chain (MMC) paradigm [4], we propose modifying AlphaFold2’s iterative embeddings to push structural predictions towards lower energy conformations. This construction enables us to embed various solvent conditions, such as pH, into protein structure prediction. Further, by operating over an existing embedding space, we can leverage the millions of existing weights learned over terabytes of data in AlphaFold.

In the first section, we discuss the foundations for complete structural modeling of protein folding, focusing on the underlying energy functions and molecular dynamics simulations. We will then transition into a discussion of MMC and how the method has been used to explore complex conformational spaces. The challenges revealed in these sections will highlight the significance of AlphaFold’s predictive capabilities. We will then transition into an analysis of the AlphaFold2 architecture, with a particular focus on the iterative Evoformer module. This discussion will focus on the importance and pitfalls of homology while also analyzing the embedding subspaces in the protein representations.

These analyses will motivate the second section, where we comprehensively introduce the Subspace Relaxation Operator (SRO). First, we will connect energy gradient conditioning into the SRO and lay out the complete model architecture. Then, we will discuss the creation of our novel intermediate embedding dataset, concluding with the results of training our refinement model. These results will reveal that when conditioned by external force gradients, the AlphaFold embedding spaces are operable. In particular, the SRO refines intermediate embeddings to yield proteins within lower energy states that contain more physical structures. With this ability, we can integrate physical context into AlphaFold structural prediction. These findings open the door to leveraging AlphaFold as a foundation model and fine-tuning operators for task-specific purposes in the protein folding domain.

2 Computational Models for Protein Folding

2.1 Computationally Simulating Protein Folding

2.1.1 Energy Functions

Proteins lie in three-dimensional space, which means that a protein of sequence length n has $3n - 6$ degrees of freedom in its number of possible configurations. We can evaluate each configuration by computing its energy.

Definition 1

An energy function is some function $U(\cdot) : \mathbb{R}^{3n} \rightarrow \mathbb{R}$. For an arbitrary set of positions $p \in \mathbb{R}^{n \times 3}$, where n is the length of the protein, $U(p)$ defines the energy of this configuration.

In this work, we will refer to energy as the configuration’s potential energy U , an aggregation of physical properties including bond stretching, bond angle bending, bond torsions, van der Waals interactions, and electrostatic interactions, not to mention quantum mechanical effects. Proteins, in their native conformations, are generally associated with low potential energy U . Over high-dimensional spaces, these conformations will be found in global or almost global minima of U .

Energy functions tend to be incredibly nonconvex over a vast domain, making locating global or nearly global minima through methods such as gradient descent infeasible for large biomolecular systems. With this in mind, we will consider how methods such as molecular dynamics, Metropolis Markov Chain (MMC), and AlphaFold allow us to find rough estimates of these minima and sample these conformational spaces.

2.1.2 Relaxation

One popular approach for finding states of low energy is by performing energy minimization - the problem of determining a local minimum on some energy function. Leaving aside the task of finding the global minimum, let us first consider the task of **relaxation**. When relaxing a protein structure, we start at a point on our multi-dimensional energy surface - some initial structure - and seek to descend to some nearby stable state or local minimum [5].

Suppose we start from some initial set of positions p and want to find a nearby local energy minimum. To do this, we must look to Newton’s second law, which says that a particle accelerates along the direction of the force it experiences at a given instant. Intuitively, we would expect a particle to move in a direction that minimizes its potential energy. Then, given an energy function, we can extend Newton’s law,

$$\mathbf{f}_i = m_i \mathbf{a}_i = -\nabla_{p_i} U$$

Guiding Protein Embeddings to Low-Energy States via Subspace Relaxation

where $\mathbf{f}_i$ is the force a particle experiences due to the potential energy U . m_i and $\mathbf{a}_i$ respectively represent the mass and acceleration of the particle while ∇_{p_i} represents the gradient operator. Notice that we take the negative of the gradient, the direction of maximal increase in the energy function, because the force will act in the direction that decreases the potential energy. When this gradient operator equals zero, we can infer that the energies are at a local or global minimum. To approximate U in practice and consequently take its gradient, we must leverage force fields.

Definition 2

Given a differentiable energy function U , a force field $F(t)$ is a callable function that returns $-\nabla U(t)$. For computational simplicity, U is often defined as a sum of simple differentiable functions such as Lennard-Jones or harmonic potentials.

In this study, we use the Amber 99SB force field - one of the most popular approximations in biomolecular simulations [6] - to approximate U and $-\nabla U$. Returning to energy minimization with the ability to approximate $\nabla_{p_i} U$, one simple approach is to iteratively follow this gradient in steps to roll down to the local minimum with small step size α_k :

$$p^{(k+1)} = p^{(k)} - \alpha_k \nabla U(p^{(k)})$$

While this procedure will find a particular state of local minimum, we are more interested in finding wide "basins" of low free energy [5]. This can be understood as finding a larger collection of low-energy micro-states rather than a singular low-energy state. To resolve this issue, we will have to look beyond relaxation and energy minimization to molecular dynamics (MD) and MMC.

2.1.3 Exploring Conformational Spaces with Molecular Dynamics

When performing molecular dynamics simulations, the goal instead becomes to navigate the Boltzmann distribution for some set of atoms,

$$p_b(p) = \frac{1}{Z} e^{-\beta U(p)}$$

which describes the probability of observing a particular atomic configuration. Notice that by raising the energy to the exponent, we exponentially increase the probability of observing low-energy states. However, to normalize these energy-dependent values to a probability distribution, we must divide the term by the partition function,

$$Z = \sum_{p_i \in S} e^{-\beta U(p_i)}$$

If you consider the infinite cardinality of S and the unknown functional form of U , it is computationally intractable to compute the partition function, which is why sampling through MD and MMC is incredibly important. Returning to our Newtonian equations,

Guiding Protein Embeddings to Low-Energy States via Subspace Relaxation

$$\mathbf{m}_i \frac{d^2 p_i}{dt^2} = \mathbf{f}_i = -\nabla_{p_i} U$$

Supposing we know $\mathbf{a}_i(t) = \frac{d^2 p_i}{dt^2}$, $\mathbf{r}_i(t)$, and $\mathbf{v}_i(t)$, we want to determine the positions, velocities, and accelerations for the next instant in time $t + \Delta t$. By integrating our acceleration, using the Verlet algorithm, for example, we can compute the next step in our simulation trajectory. To sample our initial velocities $\mathbf{v}(0)$, we can sample from the Maxwell-Boltzmann distribution. With this complete setup, we can generate a complete trajectory in the following iterative process:

1. Calculate the forces and accelerations from $-\nabla U(t)$ given a states current $\mathbf{a}$, $\mathbf{v}$, and p .
2. Use numerical integration methods to determine the positions and velocities for the next step $t + \Delta t$, preparing for the computation of $-\nabla U(t + \Delta t)$

The key advantage in using MD is that while we generally tend toward lower-energy states, battering from the simulated solvent and random thermal motions enables the process to go uphill and potentially discover other energy wells. This ability enables MD to better capture the conformational dynamics of a protein. In practice, it is possible to enhance this sampling by filling wells with "computational sand", forcing MD to move out of wells and explore the energy landscape.

While this is a highly accurate procedure, it is also incredibly time-intensive. To properly complete these calculations, proteins must be surrounded by a solvent box that can contain more than a hundred thousand atoms. Combined with the protein, each time step, which is around 2×10^{-15} seconds, requires calculating gradients for each of those atoms. If the goal is to simulate a protein's entire folding process, say around a second, that could take months on some of the most advanced computational resources. In our work, we leverage OpenMM to apply the previously discussed integration and iterative simulation process [7].

2.2 Estimating Protein Conformations Using MMC

Metropolis Markov Chain (MMC) is a variant of the Markov Chain Monte Carlo (MCMC) class of algorithms [8]. MCMC algorithms, generally, seek to generate estimates from high-dimensional probability distributions like our Boltzmann distribution $p(x) = Z^{-1}e^{-\beta U(x)}$. The primary advantage of using MMC is that it only requires the energies of a particular configuration x , removing the need for expensive force fields and derivatives. Further, MMC is not limited to the femtosecond time step, allowing for broader and more rapid exploration of conformational spaces.

In simplified terms, MMC proceeds by first choosing an initial conformation x_k and evaluating its energy $U(x_k)$ [4]. Then, sampling from a proposal distribution, we choose a random move to a new configuration x_{k+1} and evaluate its energy $U(x_{k+1})$. However, our Markov process only accepts this proposed x_{k+1} based on the *Metropolis acceptance probability*.

Guiding Protein Embeddings to Low-Energy States via Subspace Relaxation

Definition 3

Given a conformation x_k and proposal x_{k+1} , we define

$$R_+(k) = \frac{p(x_{k+1})}{p(x_k)} = e^{-\beta(U(x_{k+1}) - U(x_k))}$$

as the *ratio of probabilities* between the two sets. Given this ratio, we define the *Metropolis acceptance probability* for the proposal x_{k+1}

$$a_{k \rightarrow k+1} = f(a, b) = \begin{cases} 1, & \text{if } R_+(k) > 1, \\ R_+(k), & \text{if } R_+(k) \leq 1 \end{cases}$$

Intuitively, $a_{k \rightarrow k+1}$ ensures that if a proposed move is downhill in our energy landscape, we will always accept it. On the other hand, if the proposed move is uphill, you accept x_{k+1} with probability $R_+(k)$. This construction encourages the Metropolis procedure to sample the most populated or low-energy states the most often, but also makes it possible for it to explore trajectories that go uphill towards other energy wells. If we collect the sequence of visited states $\langle x_1, \dots, x_k \rangle$, then as $k \rightarrow \infty$, the MMC procedure approaches the equilibrium Boltzmann distribution for $p(x_k)$. Proving this requires a closer examination of the symmetry of the proposal function and the acceptance probability, which we will omit here.¹ In addition, it is possible that our Markov chain will be functionally irreducible, which would provide additional convergence issues.

Regardless, the clear value here is that our entire process is solely based on a ratio of energies. As a result, we no longer need the partition function Z , removing a large computational barrier. All being said, the key to the success of the MMC procedure is the choice of the proposal function or move set. In practice, moves made are often directly linked to individual components within the calculation of our potential energy U : single-bond rotations, two-bond rotations, or three-bond rotations, as an example. It would not be feasible for the moveset to act over the entirety of a protein, especially considering the vast number of possibilities that would entail. Altogether, MMC is a highly effective approach for rapidly sampling conformational spaces with significantly lower computational resources in comparison to MD. However, sampling over the domain of entire protein structures demands that MMC protocols must run for significant periods to yield proper results. As a side effect of this, choosing an effective proposal function is also incredibly challenging.

¹In our proposed application, our proposal function AlphaFold inherently should be biased towards lower energy states. This is ultimately acceptable because we are not seeking to sample the entire conformational space.

Guiding Protein Embeddings to Low-Energy States via Subspace Relaxation

2.3 AlphaFold2

2.3.1 Model Introduction

Thus far, we have introduced energy functions, molecular dynamics simulations, and MMC sampling algorithms, three significant topics of study in the protein folding domain. A unifying theme within these topics is leveraging statistical or physical methods to approximate and explore a high-dimensional energy landscape. However, these approaches are still burdened by the computational feasibility required for accurate modeling.

In 2021, Google’s DeepMind released AlphaFold, a deep-learning-based method that leverages vast structure databases to predict 3D protein structure with significant accuracy [2]. In turn, this enables AlphaFold to avoid directly optimizing based on the energy of the predicted conformation. Guided by the patterns in homologous protein structures, the model implicitly learns the complex relationships between amino acids that yield low-energy structures. This learning occurs by optimizing over a loss function that penalizes physically invalid structures and encourages predicted structural similarity to an experimentally validated protein structure.²

From an architectural perspective, AlphaFold can be broken into three major components: multiple sequence alignment (MSA), the Evoformer, and structure prediction. In the first component of the model, an input peptide sequence is passed through genetic and protein structure databases, generating an MSA and pair-wise representation. Then, the Evoformer block uses various attention mechanisms to refine the pair-wise peptide relationships. This process is similar to how large language models utilize attention to recognize semantic meaning within text [9]. Finally, these refined protein representations are then passed to the structure module, where the final protein structure is predicted. This cycle between the Evoformer and structure module is repeated 3-4 times and can be interpreted as AlphaFold incrementally moving towards and refining its final structure prediction.

Through numerous metrics, this model has achieved great success and, most recently, won the Nobel Prize. Various studies and our perturbations show how the generated MSA and homologous data are key drivers for this success. Notably, the authors also state that the model’s performance is only guaranteed when the MSA has a minimum depth of around 30 significantly similar sequences. Beyond the homology data, there have been minimal studies into the embedding spaces that underlie this prediction. Analyzing these learned numerical representations of protein structure can also help us to better understand AlphaFold’s predictive capabilities.

Here, we will primarily focus our analysis on two main components of AlphaFold: homology and Evoformer. First, we will discuss relevant mechanisms within the Evoformer, and then transition into how one of its inputs, the MSA, significantly biases the ultimate prediction. Then, we will transition into a linear algebraic analysis of the high-dimensional spaces in which the protein representations are embedded. These analyses will ultimately

²In reality, AlphaFold2’s training dataset does not only contain validated protein structures. One-fourth of the dataset is derived from true structures, while the remaining three-fourths is a distilled dataset. These computationally generated samples are a product of noisy student training.

Guiding Protein Embeddings to Low-Energy States via Subspace Relaxation

reveal spaces in which our introduced Subspace Relaxation Operator can influence protein structure prediction.

2.3.2 The Evoformer Module

First, we must discuss the underlying architecture of the Evoformer, a derivative of the popular transformer in the language modeling task [9]. Just as transformer blocks are stacked, the Evoformer is a stack of 48 individual blocks. For a particular protein sequence X of length n , the inputs to each block are

$$\text{MSA}_k \in \mathbb{R}^{\text{depth} \times n \times d_{\text{msa}}}, \text{Pair}_k \in \mathbb{R}^{n \times n \times d_{\text{pair}}}.$$

Instead of existing in sequence form, the MSA of a specified depth is embedded into $\mathbb{R}^{d_{\text{msa}}}$. This can be interpreted as a numerical representation for every position of every sequence deemed highly related to our input X . On the other hand, our pair representation can be interpreted as a fully-connected graph, where $(\text{Pair}_k)_{ij} \in \mathbb{R}^{d_{\text{pair}}}$ represents the relationship - or vertex - between the i -th and j -th amino acids in X . With each block in the Evoformer, the goal is to enhance these representations such that the final MSA_{47} and Pair_{47} are input into the structure module and yield a favorable structure. Notably, the dimensionality and shape of the inputs and the outputs remain the same across each step or block.

Within a block, numerous attention-based operations enable the Evoformer to relate homologous sequences in the MSA and specific amino acid pairs in the sequence. Self-attention, popularized by its success in language modeling, compares each position in an input sequence to every other position, allowing for a model to consider long-range relationships in input data [9]. Applied to the MSA, for a given position X_i , the Evoformer attends to all other positions X_j in column attention and the i -th position of every other sequence aligned in the original MSA. This enables a comprehensive integration of the sequence information of every homologous sequence and the other amino acids within our sample X .

However, the cost of these operations rapidly increases. For a protein of length n and MSA depth d , this would demand $O(n^2 + nd)$. If we were to apply a similar scheme to the pair-wise representation containing n^2 elements, the cost would explode to $O(n^4)$. To combat this, AlphaFold introduces triangle attention, which seeks to decrease the cost to $O(n^3)$.

Similar to how the MSA attention is decomposed into rows and columns, we attend to the pair representation by aggregating attention scores along rows and columns. In particular,

$$\text{TriAttn}(P) = \text{LayerNorm}(\text{TriStart}(P) + \text{TriEnd}(P))$$

where TriStart represents the multi-head attention that for a fixed position i, j attends to every pair j, k while TriEnd , for a fixed position i, j attends to every pair i, k . Adding these together, the attention mechanism has then built a triangle for every combination of indices i, j, k where $i, j, k \in [0, n]$. These components of the triangle attention mechanism can be more formally defined as follows:

Guiding Protein Embeddings to Low-Energy States via Subspace Relaxation

$$\text{TriStart}_{i,j}(P) = \sum_{k=1}^n \text{softmax} \left(\frac{Q_{i,j} K_{i,k}^T}{\sqrt{d_{\text{pair}}}} \right) V_{i,k}$$

$$\text{TriEnd}_{i,j}(P) = \sum_{k=1}^n \text{softmax} \left(\frac{Q_{i,j} K_{k,j}^T}{\sqrt{d_{\text{pair}}}} \right) V_{k,j}$$

Here, Q for query, K for key, and V for value are linear projection matrices with more advanced gating mechanisms. Note that when we fix an arbitrary position, i, j in TriStart for example, we use $Q_{i,j}$ as the query, and then use the key $K_{i,k}$ and value $V_{i,k}$ from each compared position to determine the attention output. Notably, this decrease the naive pair-to-pair comparison cost from $O(n^4)$ to $O(2n^3) = O(n^3)$. When working with incredibly large sequences and high-dimensional embedding spaces, this optimization is necessary to save memory.

At the end of each Evoformer block, the information from the MSA is fused into the pair embedding. Following all of the Evoformer blocks, we are left with the two matrices MSA_{47} and Pair_{47} that can then be passed to the structure module.

2.3.3 AlphaFold2’s Reliance on Homology

Previous studies have shown that the MSA depth drives overall accuracy. In the AlphaFold paper, when major data sources such as BFD and MGnify are removed [10, 11], the average performance degrades significantly. Further, when the median alignment depth drops below around 30 sequences, a similar drop in performance is observed [2]. A consequence of this is that AlphaFold is only trained to make predictions for wild-type sequences and is not sensitive to point mutations.

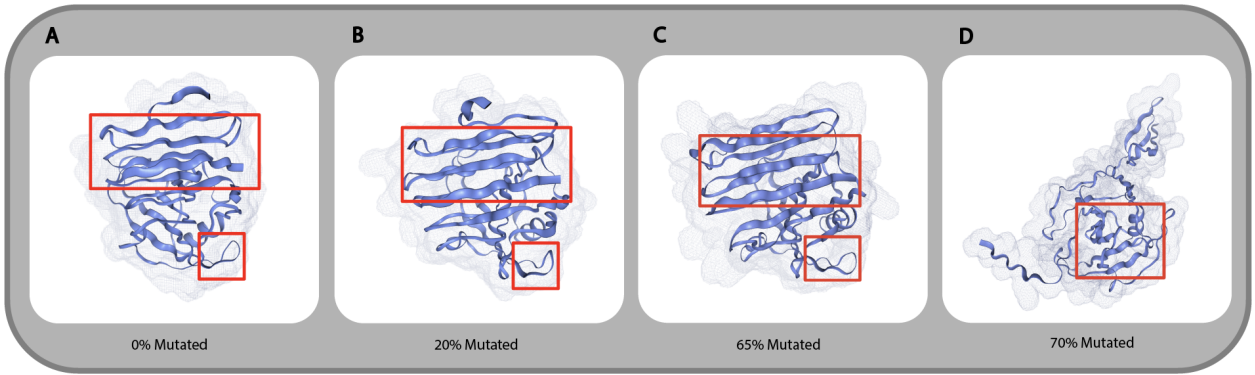


Figure 1: Selected Structural Changes of 6kwc After ϵ -Fractional Mutation, $\epsilon \in \{0.02i \mid 0 \leq i \leq 50, i \in \mathbb{Z}\}$. **A** Structure of 6kwc with no mutation. Red highlights indicate key structural components to track through further mutation. **B** Structure of 6kwc with 20% of its amino acids mutated uniformly randomly. **C** Structure of 6kwc with 65% mutation. **D** Structure of 6kwc with 70% mutation. Highlighted structural elements decompose, with some reminiscence in the new highlighted region.

Guiding Protein Embeddings to Low-Energy States via Subspace Relaxation

Here, we will examine how the predicted structure for 6kwc, or Endo-1,4-beta-xylanase 2 [12], changes under progressive mutation ³. First, in 1A, we predict the baseline AlphaFold structure for 6kwc without any perturbation. We also note that the top hit in the MSA was 2DFB (Endo-1,4-beta-xylanase) with p-value 5.4×10^{-71} . Here, we use OpenFold, an open-source implementation of AlphaFold2 with pre-trained weights instead of the true DeepMind AlphaFold [13]. For convenience, we will refer to OpenFold as AlphaFold.

Then, we progressively mutated an ϵ -fraction of the protein (length = $n = 191$) with random amino acid substitutions. After mutating 20% of the original amino acids, we see that the predicted protein in 1B remains fairly similar to the original structure, retaining the large beta sheets, smaller structural elements, and the same top hit in the MSA.

Interestingly, when 65% (123 positions) are randomly mutated, the resulting structure still retains the same key structural features highlighted in the original prediction (1C). Even though this sequence practically represents an entirely new protein, the continued presence of 2DFB (p-value 4.6×10^{-4}) seems to push the model towards a homologous structure. Finally, when we mutate just one more amino acid, 2DFB drops from the MSA altogether, and this corresponds with a significant alteration in the predicted core and an unraveling of surrounding structural elements (1D).

While homology is an effective method for generating ground-truth protein structure predictions, it yields robustness that allows the model to produce nonphysical and unlikely results. One important caveat in these analyses is that we are using a subset of the homology data generally available to AlphaFold. However, having a smaller quantity of data allows us to more easily illustrate the impacts that poor MSAs can have on the model’s predictive capacity.

2.4 Analysis of AlphaFold2 Intermediate Subspaces

The ability to interpret "black box" deep learning models is very important. In genomics and medical settings, for example, attention mechanisms are probed to reveal important motifs or image segments for classification tasks [14, 15, 16]. In language modeling, dimensionality-reduction algorithms are applied to cluster and analyze semantic meaning in texts [17]. Finally, in contrastive learning settings, similar dimensionality-reduction analyses allow us to visualize how embeddings shift in training over time. These analyses provide insights into what signals a deep learning model may or may not be picking up on in training or in prediction.

While some work has shown the utility of the Evoformer and MSA embeddings in downstream functional classification tasks [18], there has not been a comprehensive analysis of how these embedding spaces evolve throughout inference. Probing these embeddings could provide a lens into the empirical success of AlphaFold. Here, we will center our embedding analysis on the principal components of the spaces that our pair embeddings lie within.

³This protein was selected as it is featured on the main documentation for OpenFold, the open-source implementation of AlphaFold2.

Guiding Protein Embeddings to Low-Energy States via Subspace Relaxation

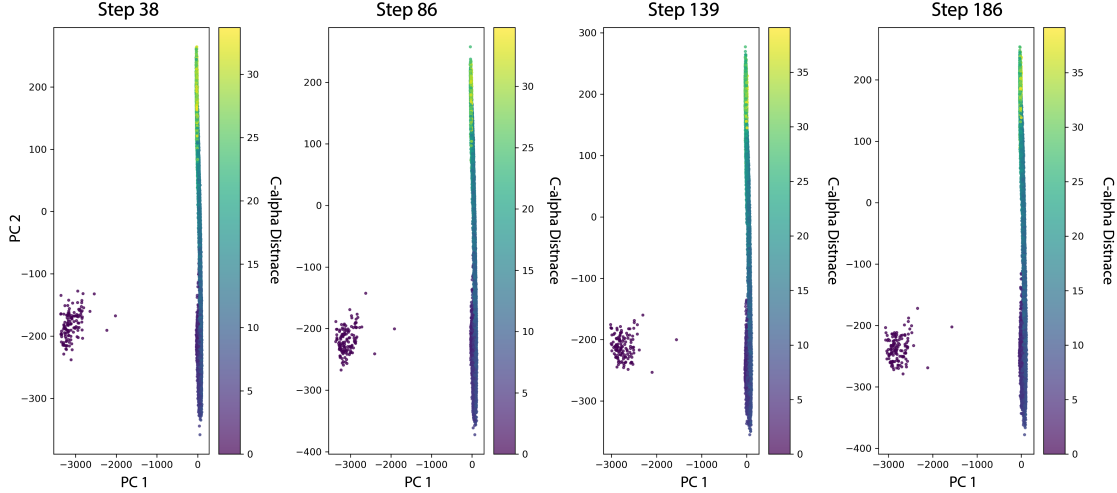


Figure 2: A visualization of the first two principal components of 1a9v AlphaFold Evoformer pair embeddings extracted at various blocks. One block, or step, was selected from each cycle for visualization. Each point represents an amino acid pair and is colored by the predicted distance between the two amino acids from the intermediate structure.

2.4.1 Individual Intermediate Embedding Subspaces

For our pair embeddings Z_k , $d_{\text{pair}} = 128$. Because this is impossible to visualize in its true dimensionality, we can leverage principal component analysis (PCA) to visualize the embeddings along a subspace that maximally captures the variance of the data in a lower dimension. The two-dimensional projection of our pair embedding $Z_k \in \mathbb{R}^{128}$,

$$Q_2 = (Z_k - \mathbf{1}\mu^T)[\mathbf{v}_1, \mathbf{v}_2]$$

where the first term centers the embedding using the mean μ and then, using the first two principal components - the eigenvectors corresponding to the largest eigenvalues - projects the data into $\mathbb{R}^2$.

Here, we take a sample protein in our dataset, 1A9V, and perform PCA on its pair-wise embedding representation at random blocks and cycles in the Evoformer stack [19]. Each point on the plot represents the distance between that pair of amino acids in the predicted structure at that intermediate step. Across all of the visualized embeddings, we see two clear clusters in the data: one distributed along the first principal component and another along the second.

The smaller cluster towards the bottom left solely consists of the elements in the main diagonal, $(Z_k)_{ii}$ for $i \in 1 : n$. While PC1 separates pairs where $i \neq j$ from the main diagonal, PC2 seems to capture variance among the remaining pairs. We can also see that this distribution of pairs is consistently stratified by the predicted distance between the two amino acids. Performing a similar analysis on the dimension reduction for the single embeddings, we see that similar amino acids tend to cluster towards one another.

Each of the principal components $\mathbf{v}_i$ has a corresponding eigenvalue λ_i that quantifies the variance in the original data explained by $\mathbf{v}_i$. When PCA is performed for k components, we

Guiding Protein Embeddings to Low-Energy States via Subspace Relaxation

have $\lambda_1, \dots, \lambda_k$, which we use to determine the variance captured by the top k components. In this embedding space, the top 2 components capture, on average, 75% of the embedding variance. Beyond the fourth PC, the explained variance values consistently remain less than two percent.

2.4.2 Comparison of Subspaces Across Blocks and Cycles

Visually, the dimensionality-reduced embeddings appear to look similar but may appear along different components. Towards the goal of manipulating the embedding spaces in the Subspace Relaxation Operator, directly quantifying the change between consecutive embedding spaces is important. Given two sets of orthogonal principal components for two different embedding spaces, we can measure the geodesic distance between the two spaces as a function of their canonical angles. Intuitively, smaller geodesic distances would decrease the challenge of predicting embedding perturbations as the SRO would not have to also account for the shift in underlying vector subspaces.

Definition 4

Given two embedding spaces $X \in \mathbb{R}^{n \times d}$ and $U \in \mathbb{R}^{n \times d}$, and the top- k principal component matrices $Q_X \in \mathbb{R}^{k \times d}$ and $Q_U \in \mathbb{R}^{k \times d}$,

$$M = Q_X^T Q_U$$

where M describes the relationship between two sets of orthogonal vectors. If $M_{ij} = 0$, then $(Q_X)_i \perp (Q_U)_j$. Taking the single value decomposition of M , we have,

$$M = V \Sigma W^T$$

where $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_k)$. Here, we define the **principal or canonical angles** of the matrix as:

$$\theta_i = \cos^{-1}(\sigma_i), \quad i \in 1 : k$$

The canonical angles collectively describe the rotation required to align the vector space spanned by Q_X and Q_U . Analyzed using geodesic or chordal distances, these angles provide an interpretable mechanism for measuring the difference between two subspaces. Here, we define the geodesic distance, the shortest path between two subspaces on a Grassmann manifold [20], as

$$d_{\text{geodesic}}(\theta_1, \dots, \theta_k) = \sqrt{\theta_1^2 + \theta_2^2 + \dots + \theta_k^2}$$

given the k canonical angles derived from the principal components of X and U . First, we randomly selected 100 proteins from the PDB dataset and extracted the intermediate pair embeddings from every step along each cycle. Then, between each consecutive step,

Guiding Protein Embeddings to Low-Energy States via Subspace Relaxation

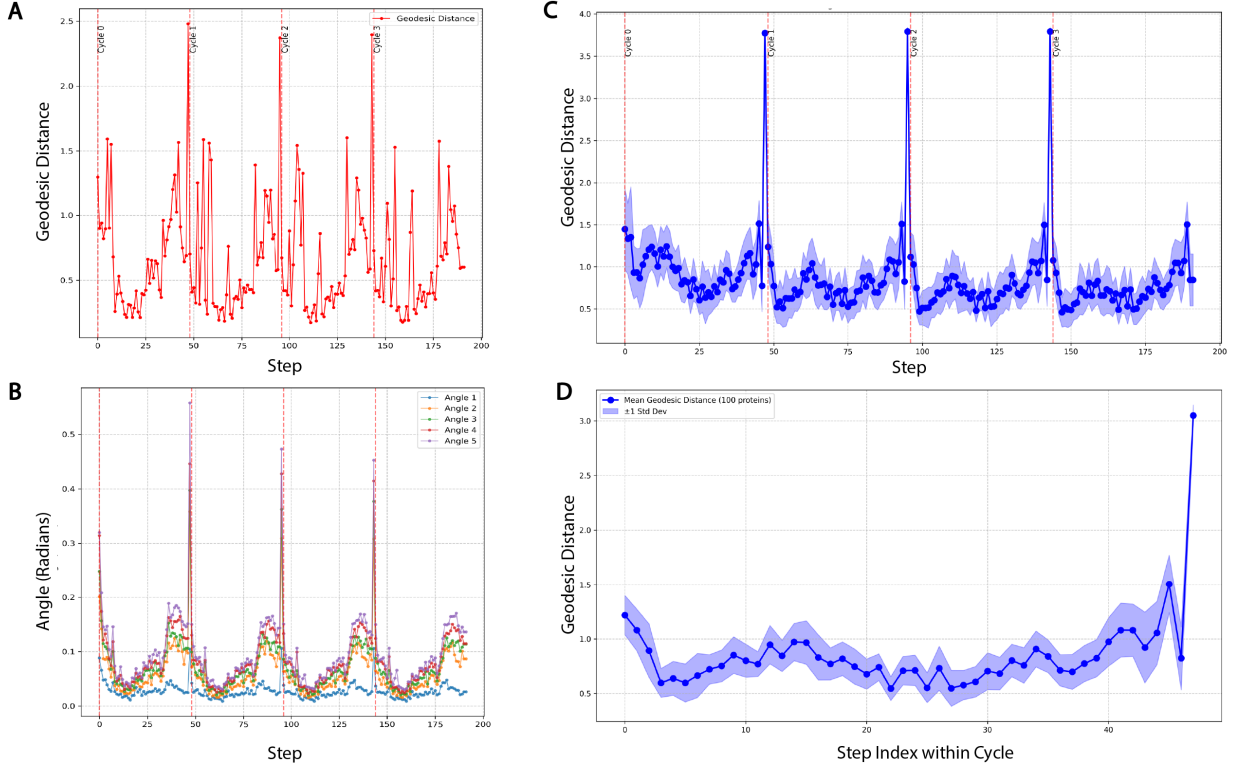


Figure 3: An analysis of consecutive and cyclical subspaces defined by the top ten principal components of Evoformer block embeddings. Across all plots, cycles are separated by a red dashed line. **A** The consecutive geodesic distances between consecutive blocks for the protein 1ASS Chain A. To the right are the canonical angles used to calculate these distances, with only the five largest angles plotted for each block. **B** An average of the geodesic distances between blocks across 100 proteins, with the 1st standard deviation shown above and below the lines. **C** The average distance between the same Evoformer blocks across the four cycles aggregated across 100 proteins.

we determined the canonical angles and geodesic distance between the respective embedding spaces from the Evoformer’s pair embeddings using the top ten principal components. Given the above formulation of d_{geodesic} , the maximum possible distance for 10 principal components is $d_{10} = \sqrt{\frac{\pi}{2} \times 10} \approx 4$.

In Figure 3A, we show the progression of the embedding spaces for the protein 1ASS Chain A. At both the end and beginning of a cycle, denoted by a red dotted line, there are large upwards shifts in both the geodesic distance and the individual angles. Similar to how the variation was primarily seen in the first three principal components, the first three canonical angles generally account for a majority of the observed distance between consecutive embedding spaces (Figure 3B).

To view a more general pattern, we aggregated the geodesic distances between steps and plotted the mean across the samples (Figure 3C). As we observed in the individual protein, there are large spikes that almost reach the maximum d_{10} in the geodesic distances towards the end of each cycle. Interestingly, while there seems to be a consistent plateau between these spikes for cycles two through four, there is more variation in the consecutive

Guiding Protein Embeddings to Low-Energy States via Subspace Relaxation

subspaces during the first cycle. This is potentially explained by the bulk of the protein structure prediction and movement of the embeddings occurring during the first cycle before refinement in subsequent cycles. Further, for a majority of sequential blocks, it appears that there is no significant variation in the underlying pair embedding space.

Next, we performed the geodesic distance analysis on embedding spaces produced by the same blocks across different cycles (Figure 3D). Generally, the distances were small, with a majority falling below 1. Similar to the sequential analysis, there was a spike in the distances towards the end of the cycle, which is to be expected if the produced structures from each cycle are different.

3 Subspace Relaxation Operator for Protein Structure Predictions

3.1 Approach

To guide AlphaFold towards more physical structural predictions, we introduce a **Subspace Relaxation Operator** (SRO). Given a pair embedding, Z_k and a corresponding structure prediction Y_k , $\text{SRO}(Z_k)$ produces an embedding Z'_k such that $E(Y'_k) < E(Y_k)$. This can alternatively be viewed as predicting an embedding that produces a more physical protein structure.

The generation of the physically altered ground truth will require a dataset of physically altered proteins to train towards. The ability to move embeddings towards a distinct class of protein structures would suggest that the AlphaFold embedding space is *operable*. This understanding could enable us to leverage the SRO in inference to steer the model’s predictions considering a particular solvent or energetic condition. Further, this would open using AlphaFold as a foundation model for tasks beyond ground-state structure prediction.

3.2 Dataset Generation

To train the SRO, we first must compile a collection of intermediate pair embeddings $Z_i \in \mathbb{R}^{n \times n \times d_z}$ with corresponding structure predictions $Y_i \in \mathbb{R}^{3n}$. Then, we must produce a corresponding set of physically operated structures $Y'_i \in \mathbb{R}^{3n}$ that will serve as our ground truth labels for training the SRO. Using the known Structure Module SM , the loss will be computed as a function of $SM(\text{SRO}(Z_i))$ and Y'_i . This process for generating the data is laid out in Figure 4. Ultimately, the goal is to use the physically altered structures to move the embeddings towards a state that would produce our desired lower energy structure.

First, we ran AlphaFold on a collection of PDB monomer protein sequences and saved 5 intermediate embeddings Z_i and structures Y_i from randomly selected blocks within each cycle [21, 13]. To generate the physically altered data, we took the extracted intermediate PDB files and solvated the proteins in a water box using GROMACS [22]. Recall, from the forces and molecular dynamics sections, that this is required to generate a proper force field to simulate the protein’s dynamics. Further, we protonated the protein to simulate a pH 5

Guiding Protein Embeddings to Low-Energy States via Subspace Relaxation

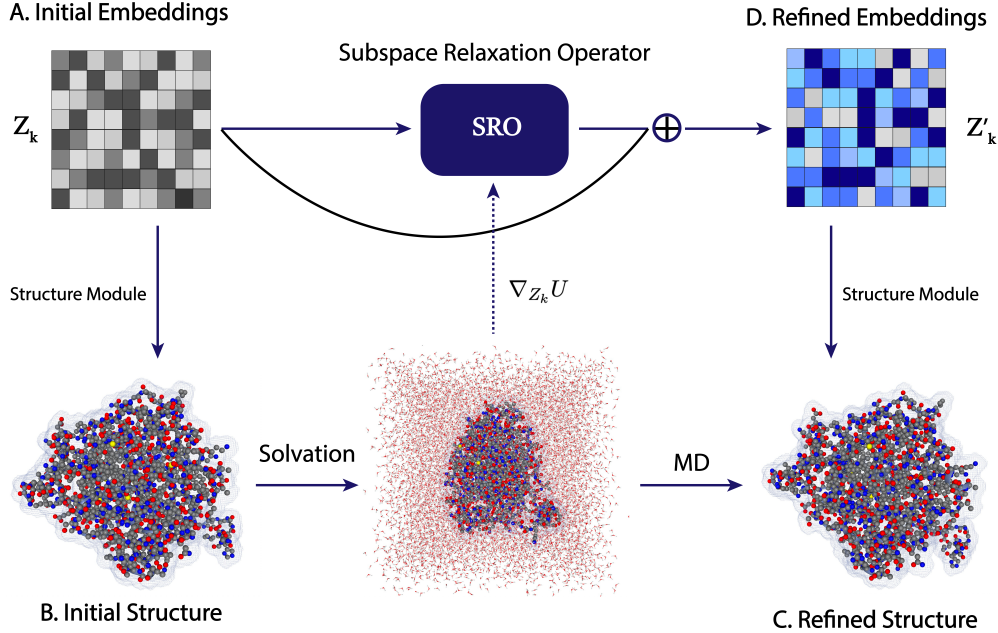


Figure 4: Data and training schematic for the Subspace Relaxation Operator. **A** We first extract a collection of intermediate pair embeddings from the Evoformer. **B** We then use the trained structure module from AlphaFold to produce the related initial structures corresponding to our embeddings. **C** After solvating our proteins under the desired condition (i.e. pH 5) we perform a short molecular dynamics simulation to lower the energy and improve the predicted structure. **D** The SRO has the objective of predicting the refined embedding that produces the refined structures produced from MD in **C**. This process is enabled by the integration of force gradients with respect to the initial embeddings.

condition using PDB2PQR and PROPKA [23, 24]. Then, at 300K, we ran 20000 steps of molecular dynamics simulations using 2 femtoseconds (2×10^{-15}) intervals [6]. We repeated this physical alteration under a pH 7 condition as well (Figure 4).

On average, this process saw a decrease in energy for the proteins across all sequence sizes. A vast majority of the sequences hovered around 500 amino acids, with a few exceptional proteins exceeding 1000 in length.

3.3 Architecture

3.3.1 Conditioning on Energy Gradients

Recall from 2.1.3 that we can take the gradient of the protein’s energy with respect to a protein’s predicted position Y as $f = \nabla_Y U$. Intuitively, this vector represents the directions in which the individual atoms will move to minimize the energy of the system. In AlphaFold’s construction, a predicted structure Y is a function of the related pair embedding Z from the Evoformer $Y = SM(Z, S, \dots)$. Then, using standard gradient backpropagation,

$$\nabla_Z U = \nabla_Y U \cdot \frac{\partial SM(Z)}{\partial Z} = f \cdot \frac{\partial SM(Z)}{\partial Z}$$

Guiding Protein Embeddings to Low-Energy States via Subspace Relaxation

where **SM** itself is made up of a series of differentiable functions. This calculation provides an idea of how the embeddings must shift to result in a decrease in energy. Here, the **SRO** is conditioned on $\nabla_Z U$, providing the model with energetic context in prediction. Later, we will show using ablation that this conditioning enables the model to generalize to unseen protein structures from training and dramatically reduces overfitting to the training data.

3.3.2 Architecture Components

In a first step, we must integrate the conditioning gradients with the input embeddings Z . A popular method to perform similar integration in the language modeling and protein folding domain are FiLM (Feature-wise Linear Modulation) layers [25]. FiLM layers modulate the features of an input on external information through feature-wise affine transformations. For $Z, \nabla_Z U \in \mathbb{R}^{n \times n \times d_z}$ as follows,

$$\hat{Z} = \gamma(\nabla_Z U) \odot Z + \beta(\nabla_Z U)$$

where $\gamma \in \mathbb{R}^{d_z \times d_z}$ and $\beta \in \mathbb{R}^{d_z \times d_z}$ are linear layers that scale and shift the gradients. This mechanism is effective at conditioning an input at the feature level, enabling smooth integration of the gradients into residual embedding prediction.

The model then proceeds to use a triangle multiplicative update and triangle attention (2.3.2) [2]. Similar in style to the triangle attention mechanism, the triangle multiplicative update takes individual pairs i, j and integrates every other pair along the i -th row and j -th column. In particular,

$$\begin{aligned} a_{ik} &= \sigma(W_a^{(g)} z_{ik}) \odot (W_a^{(p)} z_{ik}) \\ b_{kj} &= \sigma(W_b^{(g)} z_{kj}) \odot (W_b^{(p)} z_{kj}) \\ x_{ij} &= \sum_{k=1}^n a_{ik} b_{kj} \end{aligned}$$

where we first apply two separate gated linear projections to each pair-embedding before integrating them in x_{ij} . Finally, the update produces a residual $\hat{Z}$ that is added to our original input $\hat{Z}$,

$$\hat{Z} = \hat{Z} + (W_z x_{ij} \odot W_g z_{ij})$$

Importantly, this captures information in the direction of $i \rightarrow j \rightarrow k$ – the outgoing direction. Then, the same operation is again performed in the inverse direction $j \rightarrow k \rightarrow i$, to capture the information flowing into the residue i . Finally, the **SRO** applies a pair transition module, which can be understood as an MLP that operates over the individual embeddings before passing on to the loss function [2].

Combining these terms, we can build our pair refinement module that predicts the refined Z' . Instead of re-creating the entire Z , this module opts to predict the $\Delta Z = Z' - Z$. In doing so, we limit the possibility of interfering with the existing structure in Z . We can

Guiding Protein Embeddings to Low-Energy States via Subspace Relaxation

interpret this as predicting the residual required to move the embedding towards a desired structural prediction.

Algorithm 1 Pair Refinement Module Forward Pass

- 1: **Input:** $Z_k \in \mathbb{R}^{b \times n \times n \times d_z}$, $\nabla_{Z_k} U \in \mathbb{R}^{b \times n \times n \times d_z}$
 - 2: **Output:** $\Delta Z_k \in \mathbb{R}^{b \times n \times n \times d_z}$ (Predicted residual for pair embedding)
 - 3: *Step 1: Gradient Conditioning*
 - 4: Normalize the gradient: $\overline{\nabla_{Z_k} U} \leftarrow \text{LayerNorm}(\nabla_{Z_k} U)$
 - 5: Apply FiLM conditioning: $\Delta Z_k \leftarrow \gamma(\overline{\nabla_{Z_k} U}) \odot Z_k + \beta(\overline{\nabla_{Z_k} U})$
 - 6: *Step 2: Triangle Multiplication*
 - 7: Compute outgoing multiplication: $\Delta Z_k^{out} \leftarrow \text{TriangleMultiplicationOutgoing}(\Delta Z_k)$
 - 8: $\Delta Z_k \leftarrow \Delta Z_k + \Delta Z_k^{out}$
 - 9: Compute incoming multiplication: $\Delta Z_k^{in} \leftarrow \text{TriangleMultiplicationIncoming}(\Delta Z_k)$
 - 10: $\Delta Z_k \leftarrow \Delta Z_k + \Delta Z_k^{in}$
 - 11: *Step 3: Triangle Attention (Start and End)*
 - 12: Compute start node attention: $\Delta Z_k^{att_start} \leftarrow \text{TriangleAttentionStartingNode}(\Delta Z_k)$
 - 13: $\Delta Z_k \leftarrow \Delta Z_k + \Delta Z_k^{att_start}$
 - 14: Compute end node attention: $\Delta Z_k^{att_end} \leftarrow \text{TriangleAttentionEndingNode}(\Delta Z_k)$
 - 15: $\Delta Z_k \leftarrow \Delta Z_k + \Delta Z_k^{att_end}$
 - 16: *Step 4: Pair Transition*
 - 17: Predict residual update: $\Delta Z_k \leftarrow \text{PairTransition}(\Delta Z_k)$
 - 18: **Output:** Return ΔZ_k
-

3.3.3 Extending the AlphaFold Loss

Loss	Weight	Targets
Distogram	0.03	Accuracy of predicted inter-residue distance distributions.
FAPE	1.00	Error between predicted and true atomic frames.
pLDDT	0.03	AlphaFold confidence in local structure.
Supervised χ	1.00	Sidechain torsion (χ) angles.
Violation	1.00	Structural violations (e.g., bond-length violations).
RMSD	0.02	C_α root-mean-square deviation from ground truth positions.

Table 1: Loss functions used in training the SRO and their individual weights. These optimal weights were selected from a sweep of SRO models trained with varying loss terms and weights.

Guiding Protein Embeddings to Low-Energy States via Subspace Relaxation

In Table 1, we show the conglomerate of loss terms that the SRO optimizes over in training. These loss functions are a direct extension of those used in the original AlphaFold paper [2]. Next to each loss term, we also include a weight that dynamically scales the contribution of each loss function to balance them. For example, the raw outputs from the RMSD and pLDDT loss functions are much larger than the other functions. To prevent gradient descent from over-fixating on these, we significantly diminish their weights. The final loss over n loss functions and an individual sample pair (Y, Y') is then calculated,

$$L = \sum_i^n w_i \cdot L_i(Y, Y')$$

It should also be noted that we include the RMSD (root mean square deviation) and pLDDT in the loss calculations, which is not done in AlphaFold. The calculation of RMSD for two folds Y and Y' proceeds by first optimally superimposing protein structures [26]. With these superimposed structures, we define

$$\text{RMSD}(Y, Y') = \sqrt{\frac{1}{n} \sum_{i=1}^n \|Y_i - Y'_i\|^2}$$

The structure module contains auxiliary heads that it uses for specific loss calculations. In particular, using our embedding Z , it predicts a confidence score over the resulting structure, the predicted local distance difference test (pLDDT). This prediction is based upon the corresponding Local Distance Difference Test (LDDT), which intuitively measures the pairwise difference between individual atoms across two structures for a protein. For example, if a predicted alpha carbon i is 4 Å away from another alpha carbon j but is 2 Å in the true structure, then this pair would incur an error of 2 Å. Finally, these values are aggregated together to determine the accuracy of the structure.

pLDDT is a trained AlphaFold function that conveys the model’s confidence in a predicted structure. We include this weight to encourage the embeddings to be shifted in a way that AlphaFold still believes is a valid protein structure prediction. Further, because we are optimizing over existing trained embeddings, we introduce the RMSD to encourage the individual atoms to move towards our ground truth positions Y' .

3.3.4 Subspace Relaxation Model

Putting these distinct pieces together, we can now define the forward pass of the Subspace Relaxation Operator or SRO. First, the model must run through the structure module to produce $\nabla_Z U$ as described in 3.3.1. Then, in a multi-step process, the model utilizes the previously defined Pair Refinement Module to iteratively refine the embeddings. Similar to how AlphaFold’s structure module iteratively edits the pair embedding using decaying cycles, the SRO slowly diminishes the contribution of each step.

Guiding Protein Embeddings to Low-Energy States via Subspace Relaxation

Algorithm 2 Subspace Relaxation Operator Forward Pass

```
1: Input:  $Z_k \in \mathbb{R}^{b \times n \times n \times d_z}$ ,  $S_k \in \mathbb{R}^{b \times n \times d_s}$ 
2: Output:  $\Delta Z_k \in \mathbb{R}^{b \times n \times n \times d_z}$ 

3: Step 1: Backpropagate Energy Gradient and Generate Forces
4:  $\nabla E_{Z_k} \leftarrow \text{StructureModuleBackprop}(Z_k, S_k)$ 
5:  $\text{grad\_}Z_k \leftarrow \text{LayerNorm}(\nabla E_{Z_k})$ 

6: Step 2: Multi-Cycle Refinement Loop
7:  $Z'_k \leftarrow Z_k$ 
8: for cycle = 1 to num_cycles do
9:    $\Delta Z_k \leftarrow \text{PairRefinementModule}(Z'_k, \text{grad\_}Z_k)$ 

10:   Step 2d: Update Embeddings
11:   Update pair and single embeddings with predicted residuals:
12:    $Z'_k \leftarrow Z_k + \Delta Z_k$ 
13:
14: Output: Return  $\Delta P'_k$ 
```

3.4 Training Procedure and Infrastructure

3.4.1 GPU Infrastructure and Gradient Smoothing

The SRO, its entirety, is implemented in PyTorch [27] while numerous chemistry-specific packages such as OpenMM are leveraged for the auxiliary tasks the SRO relies on [7]. Training the SRO requires significant technical infrastructure due to the size and quantity of samples. In particular, each pair embedding is $n \times n \times 128$ bytes in size. For a protein of length $n = 512$, the pair embedding by itself takes around 33MB of storage.

When PyTorch creates its autograd graph for backpropagation, it has to store copies of the input embeddings from the forward pass to allow for rapid updates of weights. However, each additional instance of the pair embedding Z rapidly consumes memory on the GPU. In this project, we use NVIDIA GeForce RTX 3090 GPUs that each have a limit of 24GB of GPU memory. To prevent the exploding size that increasingly long proteins can cause, we first only train with a maximum sequence length of $n = 256$. For proteins where $n > 256$, we select a random crop of the protein of size 256 for training. Due to the significant memory costs of loading samples, each GPU is still only able to safely run a batch of size 2 through the forward and backwards passes.

If, however, we were to run training and weights optimization solely with batch size 2, this would result in nearly stochastic gradient descent. Considering the training dataset contains over 18000 protein structures, this would likely yield noisy and ineffective training. To combat this, we have set up the training such that we can train the model across four GPUs in parallel using PyTorch’s Distributed Data Parallel (DDP) framework [28]. This

Guiding Protein Embeddings to Low-Energy States via Subspace Relaxation

allows us to increase the effective batch size to $2N$ for N GPUs. Then, with each GPU holding a different data shard, GPU k computes its local loss and gradient with parameters θ and batch inputs x ,

$$L^{(k)} = \frac{1}{\mathcal{B}^{(k)}} \sum_{x \in \mathcal{B}^{(k)}} \text{Loss}(\theta; x)$$
$$g^{(k)} = \nabla_{\theta} L^{(k)}$$

DDP is then able to reduce and average these local gradients to produce an accumulated gradient across the GPUs,

$$\nabla_{\theta} L = \frac{1}{N} \sum_{k=1}^N g^{(k)}$$

Once the gradients are collected, the main process (rank = 0) performs the backwards pass, increments the weights, and sends the new weights to all N GPUs. In our setup, we use 4 GPUs, which results in a still small aggregate batch size of 8. To further smooth out our gradients, we also perform gradient accumulation, a similar process to aggregation across GPUs. Instead of proceeding with an optimization step for each batch, we aggregate for G steps and optimize based on the average of the gradients across the G batches.

To monitor training and GPU infrastructure, we utilized Weights and Biases. This allowed for easy comparison of different loss constructions and model architectures [29].

3.4.2 Hyperparameter Tuning and Weights

In training such a large model, there are a vast number of tunable hyperparameters that we need to explore. In preliminary testing, we found that alterations of parameters such as the learning rate or the maximum gradient norm did not have a significant impact on the training trajectory. Instead, we prioritize a sweep over the various possible combinations of weights that are assigned to each loss. In particular, an individual loss term can become over-prioritized in optimization processes, which would diminish the contributions of other loss terms. The above selection of loss terms in Table 1 was the best-performing set of weights in our preliminary grid search shown in Table 2.

Beyond architectural hyperparameters and weights, we generally limit training to around 3 epochs. Although we have a seemingly large dataset, the 18000 proteins we have in our training data are still a very small sample of the broader class of proteins that exist. Training beyond 3 epochs consistently results in overfitting and an increase in loss on the validation datasets.

Finally, while we have frozen the structure module and use it to produce our predicted structures, we have also experimented with initializing the attention mechanism using the weights from the trained Evoformer. In our tests, SROs initialized with these weights tend to outperform those trained completely from scratch. Here, we initialize our triangle attention mechanism using the triangle attention weights from Block 48 of AlphaFold’s trained Evoformer model.

Guiding Protein Embeddings to Low-Energy States via Subspace Relaxation

3.5 Results

3.5.1 Model Selection

To select our SRO for sample-specific validation, we performed a sweep over loss terms and selected the top-performing model based on validation RMSD. As previously mentioned in Table 1, we selected the weights shown in Model 1 (Table 2). In our sweep over loss terms, we omitted manipulating the weights for supervised χ , violation, and FAPE, which we fixed at 1 per the AlphaFold fine-tuning paradigm.

Model	Loss Weights			Best Val RMSD
	RMSD	pLDDT	Distogram	
Model 1	0.02	0.03	0.03	2.065 Å
Model 2	0	0.03	0.03	2.090 Å
Model 3	0.02	0	0.03	2.139 Å
Model 4	0.02	0.03	0	2.085 Å
Model 5	0.02	0	0	2.130 Å
Model 6	0	0.03	0	2.137 Å
Model 7	0	0	0.03	2.068 Å
Model 8	0	0	0	2.135 Å

Table 2: The best validation RMSD for a series of models trained with different loss weights. Loss weights for supervised χ , violation, and FAPE, were fixed at 1 as is done in the AlphaFold training protocol.

3.5.2 Force Conditioning Enables SRO Predictive Ability

Using the dataset presented in 3.2, we present a successfully trained SRO under the pH 5 solvent condition. In Figure 5C, the trained model resulted in a validation RMSD of 2.06 Å. In Figure 5D, we show the protein 1ATZ’s initial structure extracted from an intermediate step of AlphaFold inference. Running the SRO on Z_{1ATZ} , the energy of the molecule decreased from $2 \times 10^7 \frac{\text{kJ}}{\text{mol}}$ to $-2 \times 10^5 \frac{\text{kJ}}{\text{mol}}$. Figure 5D also highlights two regions where significant structural shifts occurred, such as the appearance of an alpha helix that was previously absent in the intermediate structure but present in the true and refined structures [30]. Further, in Figure 5E, we show another pair of initial and refined structures for 1UMG. Notably, the initial structure is lacking a connected backbone and is missing most secondary structures present in the true protein. After running the SRO, however, we see the appearance of a complete backbone and secondary structures reflective of the true protein [31]. Both of these examples are around 400 peptides in length, exceeding the training crop of 256. Other evaluated proteins beyond 500 peptides in length also saw a decrease in observed potential energy. These observations are indicative of the SRO model’s ability to operate in the AlphaFold embedding space to produce lower energy structures.

Guiding Protein Embeddings to Low-Energy States via Subspace Relaxation

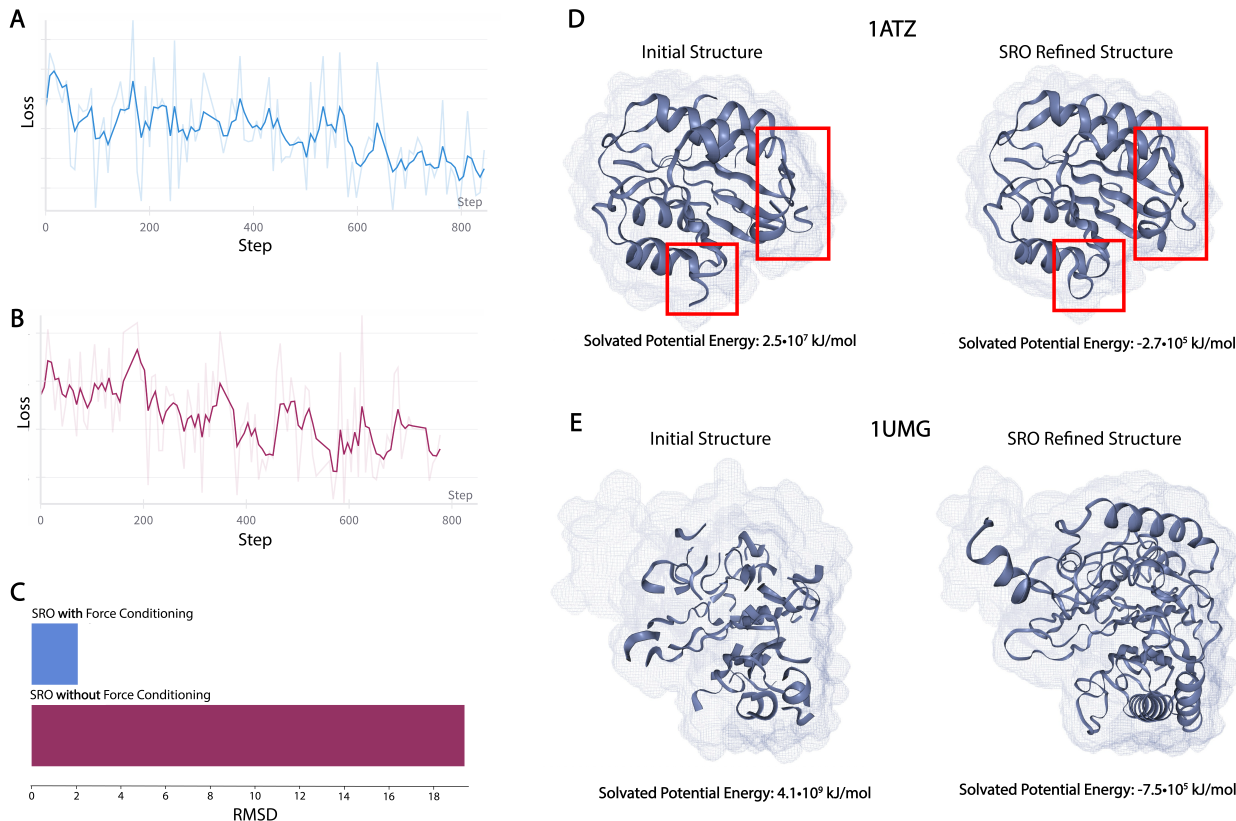


Figure 5: **A, B** Smoothed batch loss curves reported over 800 steps of model training for the SRO with force conditioning (blue) and without force conditioning (magenta). **C** Validation RMSD for both SRO models with and without force conditioning. **D, E** Initial structures and refined structures for 1ATZ and 1UMG from SRO with force conditioning with energies reported below each structure. In **D**, selected regions in the initial and refined structure show the introduction of new structural elements, including an alpha helix due to the SRO.

One key reason we can operate in this subspace is the inclusion of force conditioning. In Figure 5A and B, we show the training loss curves for two SRO models: one trained with force conditioning and one without forces. While both exhibit a similar decrease in loss throughout training, there is a stark difference in their ability to generalize to unseen data. In particular, without force conditioning, the validation RMSD explodes to nearly 20 Å. This suggests that the SRO is learning how to integrate physical information into its prediction of the residual embedding, but that it is difficult to extract that context from only the AlphaFold embedding.

Altogether, these results suggest that the SRO is a promising model for rapidly incorporating physical context to improve the energy of inferred structures. Further, we show that the ability to navigate or operate over the embedding space is enabled by differentiating the energy function we seek to minimize.

3.6 Future Integration into AlphaFold2 Inference

Here we introduce the SRO and show how it can steer embeddings towards lower energy states under the pH 5 condition. This process, first, can be extended to simulate protein folding under different conditions, such as pH 7. Once these steps have been completed, SRO₅ and SRO₇ can be integrated into the Evoformer stack.

There are two possible avenues for this integration. First, as we proposed in section 2.2, we can apply the structure module on intermediate structures, compute their energy, and apply the SRO if the predicted structure at a consecutive step is of higher energy. Alternatively, the procedure could be applied strategically throughout the stack regardless of the change in energy between two steps. One potential mechanism would be to evaluate the geodesic distance between subsequent embeddings and refine embeddings when there is not a significant difference in the underlying subspaces. Regardless, understanding AlphaFold’s behavior in the face of injected noise through the SRO would need to be investigated.

Further, it would be interesting to investigate whether an early perturbation of the embedding space results in a different folding trajectory. These resulting predicted structures, under conditions such as pH 5, can be validated against existing data to determine whether we improve structure prediction from the default AlphaFold inference. The ability to produce differential context-specific structures using the SRO would have significant impacts in the drug discovery domain.

4 Conclusion

In this study, we provide a comprehensive review of the computational methods required to approximate and sample protein conformations from high-dimensional energy surfaces. In particular, we discuss the challenges that exist in minimizing functions and how methods such as molecular dynamics and Metropolis-Markov Chain (MMC) can be applied to find such minima. However, these methods are hampered by the significant computational costs and time required for accurate sampling.

These findings motivated the introduction of AlphaFold, a deep learning model that revolutionized our ability to predict ground-state protein structure. To understand its ability, we examined its architectural components, with a particular focus on how a sequence’s multiple sequence alignment (MSA) and pair representations flow through the intermediate Evoformer blocks. In examining these constructions, we show that the AlphaFold model has an over-reliance on homology information and can lead the model to produce non-physical results.

To combat this, we proposed the **Subspace Relaxation Operator** (SRO) to take intermediate embeddings and guide them towards a refined embedding related to a structure of lower potential energy. To train this model, we collected a large dataset of intermediate structures and generated target structures of lower energy using molecular dynamics in a solvent environment of pH 5. Training over this dataset, we found that the SRO successfully operates over the AlphaFold embedding space to produce more physical results of lower

Guiding Protein Embeddings to Low-Energy States via Subspace Relaxation

energy. Using an ablation study, we also show that the SRO is only able to perform this task when provided with atomic forces of the structure in relation to the embeddings. This successful training of the refinement model opens the door to further investigations into whether the SRO can guide embeddings to different structures under varying environmental conditions, such as pH. Further, we can investigate whether the SRO can enable AlphaFold trajectories that infer conformational states aside from the ground-state structure the model is designed to predict.

The successful training of the SRO opens the door to many other courses of study. First, we must investigate how integrating the SRO into successive Evoformer blocks can alter folding trajectories. With these results, we can examine whether SROs trained under different conditions yield context-specific structural conformations. If these context-specific conformations align with validated data, this would have broad implications in the biomedical domain, where simulating different environments is crucial for therapeutic development. More importantly, these results are a first analysis of the intermediate structures of AlphaFold. Further, the outlined method is an easily reproducible pipeline for integrating physical context into structural prediction, treating AlphaFold as a foundation model for domain-specific tasks.

In a broader sense, the ability to accurately predict protein structures is crucial for drug discovery and biomedical science. While existing methods are geared towards the prediction of a protein’s ground state, these structures are often not reflective of the conformations that proteins take in varying solvents. Here, we show how the integration of physical context and AlphaFold embeddings through the SRO can help us find context-specific minima over highly complex energy functions.

5 Acknowledgments

This thesis project is a culmination of my time at Brown and would not have been possible without the support of so many of my friends, mentors, and collaborators.

First, I would like to thank Professor Sorin Istrail, my advisor for this project and my mentor over the past three years. When I first applied to Brown over four years ago, I wrote about working with Sorin, "The Open Curriculum will support an interdisciplinary experience that will prepare me to participate in research at the Istrail Lab, for example." Little did I know that taking his class in the fall of 2022 would lead me down a three-year journey that fulfilled that very goal (though I did imagine us working on evolutionary histories, not protein folding). I am constantly in awe of Sorin’s passion for computational biology and his desire to have a truly rigorous understanding of a problem. Beyond the science, though, Sorin has always been our biggest cheerleader. Whether it’s by giving us inquisitive books or international chocolates, he has constantly pushed us to be the students we can be. As much as he calls us star students, he is certainly a star professor!

Second, I would like to thank my first research advisors, Professor Erica Larschan and Dr. Mukulika Ray (now also a professor at UMass Boston!). I first met Erica as a freshman taking genetics. When I had applied to Brown, I was super excited by the idea of combining computer science and biology, but I had no idea of what that entailed. Erica and Mukulika

Guiding Protein Embeddings to Low-Energy States via Subspace Relaxation

are the best mentors I could have asked for. At the bench, Mukulika was excited to teach me wet lab techniques even though I was a computational student. I will never forget the 15+ gels that I tried (and failed to run) and the joy I felt after we figured out what was going wrong. Even though the domain was completely different, it was one of my first exposures to the innate struggles and challenges that exist in research. Mukulika, more than anyone, taught me that those failures are okay (accidentally *almost* freezing a batch of fruit fly samples was not okay, however). Outside of the lab, Erica helped me navigate countless applications and has always been my strongest advocate. Without her, I know that I would not have found a love for research, and I am eternally grateful for her guidance.

Towards the end of my sophomore year, Erica introduced me to Professor Ritambhara Singh, my second reader on this thesis. Though I started as a student in her class, I have been her Head TA for the 400+ student edition of Deep Learning, done research with her lab, and am now helping her make a new class at Brown. I am so honored to have had the opportunity to collaborate with Ritambhara across so many projects. Even when we were struggling to handle the swarm of students in DL (not quite unlike a swarm of bees...), she consistently showed her immense passion for the field and her care for those who worked with her (thank you for all the pizza!). Her work ethic and ability to comprehensively understand so many concepts are truly inspirational. More than that, conversations and meetings with her are often fun, something that I find is often missing in computer science. She is an incredible mentor, scientist, and professor, and I am very grateful to have been able to work with her at Brown.

Next, I would like to thank Ananya Pavuluri, Kaitlyn Cortez, Justin Currie, Michal Golovanevsky, Jasmine Shum, and the many other lab members who have shaped my time at Brown. I also would not be here without the endless love and support from my friends and family. My parents have wholeheartedly supported me for as long as I can remember. I am quite positive they would have never expected me to write a document that has so much math in it, given how much I would cheat my way out of the math problems they tried to make me do. Alas, the apple doesn't fall too far from the tree! Throughout this thesis, my sister has always been a pillar of support. In many ways, she is a role model in my life, and in a classic younger-sibling way, her decision to pursue research probably subconsciously pushed me to do the same.

Last, but not least, I would like to thank my collaborator on this thesis, Colin Baker. When we first started last year, I knew next to nothing about protein folding, but that was no problem for Colin; no question was a stupid question. Working and reflecting on this project so far, we have learned so much together: how annoying setting up a conda environment can be, that I am better than him at flappy golf and foosball, and many less comical but important technical skills. But beyond all that, Colin was my number one supporter throughout my chronic health issues this year. He motivated me through challenging periods and made me take rest when he knew I was not taking enough breaks. Without him, I would not have been able to write a senior thesis. Working with him on this project has been one of the best experiences of my time at Brown. Thank you, Colin!

References

- [1] Martin Karplus. The levinthal paradox: yesterday and today. *Folding and design*, 2:S69–S75, 1997.
- [2] 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.
- [3] Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J Ballard, Joshua Bambrick, et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, 630(8016):493–500, 2024.
- [4] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.
- [5] Ivet Bahar, Robert L Jernigan, and Ken A Dill. *Protein actions: Principles and modeling*. Garland Science, 2017.
- [6] V. Hornak, R. Abel, A. Okur, B. Strockbine, A. Roitberg, and C. Simmerling. Comparison of multiple AMBER force fields and development of improved protein backbone parameters. *Proteins: Structure, Function, and Bioinformatics*, 65(3):712–725, 2006.
- [7] Peter Eastman, Jason Swails, John D. Chodera, Robert T. McGibbon, Yutong Zhao, Kyle A. Beauchamp, Lee-Ping Wang, Andrew C. Simmonett, Matthew P. Harrigan, Cassia D. Stern, Rafal P. Wiewiora, Bernard R. Brooks, and Vijay S. Pande. Openmm 7: Rapid development of high performance algorithms for molecular dynamics. *PLoS Computational Biology*, 13(7):e1005659, 2017.
- [8] Alan E Gelfand and Adrian FM Smith. Sampling-based approaches to calculating marginal densities. *Journal of the American statistical association*, 85(410):398–409, 1990.
- [9] 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.
- [10] Martin Steinegger and Johannes Söding. Clustering huge protein sequence sets in linear time. *Nature communications*, 9(1):2542, 2018.
- [11] Lorna Richardson, Ben Allen, Germana Baldi, Martin Beracochea, Maxwell L Bileschi, Tony Burdett, Josephine Burgin, Juan Caballero-Pérez, Guy Cochrane, Lucy J Colwell, et al. Mgnify: the microbiome sequence data analysis resource in 2023. *Nucleic acids research*, 51(D1):D753–D759, 2023.

Guiding Protein Embeddings to Low-Energy States via Subspace Relaxation

- [12] Zhihong Li, Xiaoshuai Zhang, Chunran Li, Andrey Kovalevsky, and Qun Wan. Studying the role of a single mutation of a family 11 glycoside hydrolase using high-resolution x-ray crystallography. *The Protein Journal*, 39:671–680, 2020.
- [13] G. Ahdrizt and et.al. Openfold: retraining alphafold2 yields new insights into its learning mechanisms and capacity for generalization. *Nature Methods*, 21:1514–1524, 2024.
- [14] Jack Lanchantin, Ritambhara Singh, Beilun Wang, and Yanjun Qi. Deep motif dashboard: visualizing and understanding genomic sequences using deep neural networks. In *Pacific symposium on biocomputing 2017*, pages 254–265. World Scientific, 2017.
- [15] John Boaz Lee, Ryan A Rossi, Xiangnan Kong, Sungchul Kim, Eunye Koh, and Anup Rao. Graph convolutional networks with motif-based attention. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 499–508, 2019.
- [16] Xiang Li, Minglei Li, Pengfei Yan, Guanyi Li, Yuchen Jiang, Hao Luo, and Shen Yin. Deep learning attention mechanism in medical image analysis: Basics and beyonds. *International Journal of Network Dynamics and Intelligence*, pages 93–116, 2023.
- [17] Richard Jensen and Qiang Shen. Semantics-preserving dimensionality reduction: rough and fuzzy-rough-based approaches. *IEEE Transactions on knowledge and data engineering*, 16(12):1457–1471, 2004.
- [18] Mingyang Hu, Fajie Yuan, Kevin Yang, Fusong Ju, Jin Su, Hui Wang, Fei Yang, and Qiuyang Ding. Exploring evolution-aware &-free protein language models as protein function predictors. *Advances in Neural Information Processing Systems*, 35:38873–38884, 2022.
- [19] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.
- [20] E. Batzies, K. Hüper, L. Machado, and F. Silva Leite. Geometric mean and geodesic regression on grassmannians. *Linear Algebra and its Applications*, 466:83–101, 2015.
- [21] Helen M Berman, John Westbrook, Zukang Feng, Gary Gilliland, Talapady N Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne. The protein data bank. *Nucleic acids research*, 28(1):235–242, 2000.
- [22] Mark James Abraham, Teemu Murtola, Roland Schulz, Szilárd Páll, Jeremy C Smith, Berk Hess, and Erik Lindahl. Gromacs: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX*, 1–2:19–25, 2015.

Guiding Protein Embeddings to Low-Energy States via Subspace Relaxation

- [23] TJ Dolinsky, JE Nielsen, JA McCammon, and NA Baker. Pdb2pqr: an automated pipeline for the setup of poisson-boltzmann electrostatics calculations. *Nucleic Acids Research*, 32(suppl_2):W665–W667, 2004.
- [24] M. H. M. Olsson, C. R. Søndergaard, M. Rostkowski, and J. H. Jensen. Propka3: Consistent treatment of internal and surface residues in empirical pka predictions. *Journal of Chemical Theory and Computation*, 7(6):525–537, 2011.
- [25] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.
- [26] Feng Zu-Kang and Manfred J Sippl. Optimum superimposition of protein structures: ambiguities and implications. *Folding and Design*, 1(2):123–132, 1996.
- [27] Adam Paszke and et.al. Pytorch: An imperative style, high-performance deep learning library, 2019.
- [28] Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, et al. Pytorch distributed: Experiences on accelerating data parallel training. *arXiv preprint arXiv:2006.15704*, 2020.
- [29] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.
- [30] Eric G Huizinga, R Martijn van der Plas, Jan Kroon, Jan J Sixma, and Piet Gros. Crystal structure of the a3 domain of human von willebrand factor: implications for collagen binding. *Structure*, 5(9):1147–1156, 1997.
- [31] Hiroshi Nishimasu, Shinya Fushinobu, Hirofumi Shoun, and Takayoshi Wakagi. The first crystal structure of the novel class of fructose-1, 6-bisphosphatase present in thermophilic archaea. *Structure*, 12(6):949–959, 2004.