

Article

Protein Design Using Physics Informed Neural Networks

Sara Ibrahim Omar ¹, Chen Keasar ², Ariel J. Ben-Sasson ^{3,*} and Eldad Haber ^{4,*}¹ Proteic Bioscience Inc., Vancouver, BC V7T 1C0, Canada² Department of Computer Science, Ben Gurion University of the Negev, Be'er Sheva 84105, Israel³ Independent Researcher, Haifa 3436301, Israel⁴ Department of Earth Ocean and Atmospheric Sciences, University of British Columbia, Vancouver, BC V6T 1Z4, Canada

* Correspondence: arielbs10@gmail.com (A.J.B.-S.); ehaber@eoas.ubc.ca (E.H.)

Abstract: The inverse protein folding problem, also known as protein sequence design, seeks to predict an amino acid sequence that folds into a specific structure and performs a specific function. Recent advancements in machine learning techniques have been successful in generating functional sequences, outperforming previous energy function-based methods. However, these machine learning methods are limited in their interoperability and robustness, especially when designing proteins that must function under non-ambient conditions, such as high temperature, extreme pH, or in various ionic solvents. To address this issue, we propose a new Physics-Informed Neural Networks (PINNs)-based protein sequence design approach. Our approach combines all-atom molecular dynamics simulations, a PINNs MD surrogate model, and a relaxation of binary programming to solve the protein design task while optimizing both energy and the structural stability of proteins. We demonstrate the effectiveness of our design framework in designing proteins that can function under non-ambient conditions.

Keywords: protein design; physics-informed neural networks; binary optimization



Citation: Omar, S.I.; Keasar, C.; Ben-Sasson, A.J.; Haber, E. Protein Design Using Physics Informed Neural Networks. *Biomolecules* **2023**, *13*, 457. <https://doi.org/10.3390/biom13030457>

Academic Editors: Dongqing Wei, Gilles Peslherbe, Gurudeeban Selvaraj and Yanjing Wang

Received: 18 January 2023
Revised: 16 February 2023
Accepted: 27 February 2023
Published: 1 March 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Protein sequence design is the process of generating a sequence of amino acids that folds into a desired shape to perform a certain function. To achieve this, several approaches have been developed which can be divided into traditional-physics-based e.g., [1,2], and Machine Learning (ML)-based techniques e.g., [3–5].

Traditional physics-based methods led to the initial steps in protein design and have contributed significantly to our understanding of protein structure and function. Yet, these methods suffer from several limitations, including model complexity, sensitivity to model accuracy, computational cost, and oftentimes a lack of flexibility. Traditional algorithms often presume a fixed or almost fixed backbone. Such an assumption consequently means that the protein structure is weakly sequence dependent; thus, changing the sequence does not significantly change the structure [2,6–8]. This approximation can work only for small changes in the original sequence, or when changes in the original sequence do not break or create any significant new interactions. Indeed, if we arbitrarily change every residue in the original sequence, the structure is almost surely bound to change. Therefore, while this approach can lead to elegant optimization algorithms, it lacks robustness as the number of mutations in the sequence increases. Furthermore, such an approach necessitates a reasonable initial guess, namely a protein that already closely folds into the desired shape, making the use of this approach for *de novo* design difficult.

Machine Learning methods have proven to be effective tools for predicting protein structure and function, as well as designing new proteins [4,9–11], yet they avoid energy calculations all-together. Since ML methods do not use energy calculations and since they often lack interpretability and have limited explanatory power, there is no guarantee that

the generated structure-sequence model optimizes some target function (e.g., stability), even approximately. Furthermore, even if a sequence does fold into a desired structure it may be unstable or suboptimal.

Physics-based and deep learning-based methods rely on empirical energy functions and machine learning models, respectively. Both are parameterized by the known protein structures, regardless of the optimal conditions for their folding and operations. For most of them, however, these conditions are ambient. Thus, interpolating these functions and models to specific domains of non-ambient conditions (e.g., high temperature) is speculative, and likely wrong.

In this work, we tackle the problem of protein design from a different perspective. Rather than using the commonly used energy functions on a static structure, we use molecular dynamics (MD) simulations to compute the protein energy landscape. In this way, we both avoid the fixed backbone assumption and infer an energy landscape that is MD-derived for a target under specific environmental conditions. Thus, this method should allow us to design proteins that approximately fold into a target structure, optimize their properties even at non-ambient conditions, and account for their flexibility, under the assumption that molecular dynamics simulations faithfully describe protein interactions (see [12–14] and reference within).

A similar idea was used in [12]; however, its usability is prohibitively restricted by the computational complexity. Since the energy, structure and stability of the protein are determined by MD, computing them a number of times might not be realistic. We, therefore, extend previous work and combine MD with a deep learning algorithm that approximates the quantities obtained from MD. Replacing MD with ML is a recent practice (see e.g., [15,16] and references within). Unlike many recent papers that aim to replace MD with deep learning to approximate the entire MD trajectory, our model is designed to approximate only the average energy and deviation from the target design, requiring only a small amount of computational effort. Such an approximation is sometimes referred to as a Physics-Informed Neural Network (PINN) or surrogate model (see [15] and references within). Using PINNs as surrogate models is an inexpensive approximation that replaces complex simulations and is now commonly carried out in many fields (see, e.g., [17–20]), ranging from flow to chemical engineering and hardware testing. We, therefore, find it appropriate to extend this methodology to protein design.

We set out to construct PINNs which, given a sequence and MD data, predicts the protein energy, metrics that measure the deviation of the structure from the initial design as well as stability. Once PINNs learn a certain model, they can replace MD and be used to efficiently solve the protein design task as an optimization problem (see algorithm description in Figure 1).

To solve the optimization problem that is associated with the design, we use binary programming where the energy calculations are conducted inexpensively. We generate new, low-energy sequences as we solve the optimization problem. These sequences are simulated by MD and their energy and structure data are input into the networks to be re-trained. The algorithm exploits the neural network surrogate to MD while exploring the space by using random perturbations.

Note that a fundamental aspect of this work is that we aim to generate predictions that are as good as MD predictions. Clearly, if the simulated system is too far from equilibrium or the computed energy is incorrect then our model will fail to reproduce experimental results. However, the calculation of energies based on MD trajectories has been thoroughly studied and therefore can be considered a reliable alternative compared to many ad hoc calculations [12,21–24].

The rest of this paper is structured as follows. In Section 2, we mathematically define the protein design problem. In Section 3, we describe the surrogate model used to approximate the energy. In Section 4, we describe the optimization procedure used to design new proteins. Section 5 describes a case study and in Section 6 we summarize the paper.

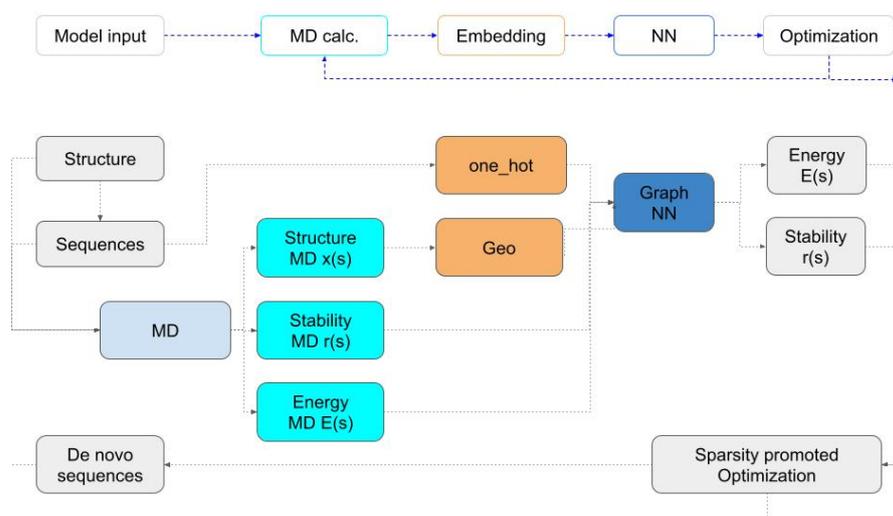


Figure 1. Design workflow. Input model structure and sequences are processed by MD simulations to generate time trajectories of structure, stability, and energy. All metrics are used to establish PINNs, followed by a sparsity-promoted optimization algorithm. The output is either used to retrain the model or outputs the suggested optimized sequences.

2. Mathematical Definition of the Protein Design Problem

In this section, we discuss the mathematical framework for the solution of the protein design problem. While this framework bears some resemblance to earlier studies in this field, there are some substantial differences that make our formulation unique.

Similar to other classical work, the driving force behind our protein design algorithm is the energy of the system. Therefore, a key requirement is to be able to compute the energy for every given sequence. Common codes assume fixed coordinates and compute the energy while others use heuristics to do so. Here, we attempt, for the first time, to introduce protein dynamics in solution under the desired environmental conditions while avoiding common assumptions that could lead to unrealistic designs.

Assume that we have a protein or a number of proteins that interact through some interface. The interface may be intra-chain, or between different proteins in a complex. We represent the system with a sequence, \mathbf{s} , and the coordinates of its full atom model, \mathbf{x} .

Assume that we have a target structure \mathbf{x}_{tar} , to which we aim the protein to fold into. This structure may be fully known, or alternatively, only a portion of the region involved in the interaction might be defined (protein in painting [25,26]). The latter may be the case when considering protein–protein interactions where only the interaction site is known. Our goal is to find sequences $\hat{\mathbf{s}}$ with low energies and low structural deviations from \mathbf{x}_{tar} .

There are a number of challenges when considering the binding energy and structural stability of a complex. First, there is no unique value to the binding energy. Proteins are flexible and may even change conformation. Second, taking into account the energy alone may not be sufficient. A low energy protein may be structurally unstable/disordered if its energy landscape has several local minima or is very flat, allowing for a high level of coordinate perturbation. We thus turn to a different approach that takes these limitations into consideration. Let us assume that we have a sequence \mathbf{s} and its all-atom model. In order to compute the energy of this sequence, we simulate the protein. MD trajectories can be analyzed to obtain the energy of the protein and extract its coordinates for every timestep of the simulation. Since proteins in solution are in constant motion and fluctuate around a minimum at equilibrium, we define the following quantities

$$\mathbf{r}(\mathbf{s}) = \frac{1}{T} \int_0^T \|\mathbf{x}(t) - \mathbf{x}_{\text{tar}}\|^2 dt \quad (1a)$$

$$E^*(\mathbf{s}) = \frac{1}{T} \int_0^T E(\mathbf{x}(t), \mathbf{s}) dt \quad (1b)$$

Note that these quantities depend only on \mathbf{s} and \mathbf{x}_{tar} as the coordinates $\mathbf{x}(t)$ are integrated out. The first quantity measures the average deviation of the coordinates from the desired fold, the Root Mean Square Deviation (RMSD), from the target. This deviation is small in stable proteins with a fold, which is similar to the desired design. However, for unstable proteins or ones that undergo large conformational changes, this quantity may be rather high. We refer to the RMSD as stability. The second component, $E^*(s)$, is the average energy of the structure.

Let us now consider the design of a new protein. In this case, we want to design a protein with a particular structure of interest \mathbf{x}_{tar} . Our goal is to find an optimal sequence $\hat{\mathbf{s}}$ with two characteristics. First, we require that the energy of the protein $E^*(\hat{\mathbf{s}})$ is low. Second, we require that its structure is stable around the required fold, i.e., $\mathbf{r}(\hat{\mathbf{s}})$ is small. This strategy permits coordinate motion (no fixed backbone assumption) and also addresses the problem of structural instability in the designed structure. These requirements lead to the following optimization problem

$$\begin{aligned} \min_{\mathbf{s}} \quad & E^*(\mathbf{s}) && \text{energy minimization} && (2a) \\ \text{s.t.} \quad & \mathbf{r}(\mathbf{s}) \leq \delta_{\text{stab}} && \text{structural stability} && (2b) \end{aligned}$$

where δ is some acceptable deviation from the target structure.

The optimization problem (2a) and (2b) is challenging to solve. The difficulty stems from two main issues. First, the energy and stability are computed by expensive simulations. While this looks initially difficult, deep learning approximations have been developed to replace expensive MD at a fraction of the cost in recent years. We thus use MD simulations to train a graph neural network that can approximate those simulations, yielding the desired quantities for the solution of the problem. Note that our network is different from networks that recover the complete trajectory. Our demands are substantially modest as we aim to approximate only the average energy of the protein and its coordinate deviation from the target design.

A major challenge in such an optimization task is the discrete nature of the sequence \mathbf{s} , which can be solved as a binary programming problem. To address this, we utilize a sparsity-promoting optimization method that employs convex relaxation to replace the binary optimization problem. This approach has proven to be effective for several binary optimization problems [27–29]. Although the method may require intensive sampling, it is highly efficient when the objective function evaluation is inexpensive. As a result, our approach of combining this method with surrogate deep learning models can be considered an effective solution for the protein design problem.

3. Deriving a Surrogate Neural Network

Since solving the optimization problem requires the MD simulation of every given sequence-structure set, and since each MD simulation is computationally intensive, we opt for a so-called surrogate deep learning model to replace the MD simulation. The goal of the surrogate is to predict the binding energy $E^*(\mathbf{s})$, and the structural stability $\|\mathbf{r}(\mathbf{s})\|$ for a given protein. These quantities can then be used directly in the optimization problem (2a) and (2b), saving on the time costs of MD simulations and significantly reducing the cost of the optimization problem.

Protein folding is influenced by both local and global interactions between residues, making it a system that can be described well using a graph. We experimented with graph neural networks (GNNs) [9] and selected the one proposed in [30] as it has an interpretation of energy.

The input to the network is the sequence (one hot encoded) and the coordinates of the alpha carbon (C_{α}) atom of each residue. Given the coordinates, we compute the distance matrix (that is, the distance between every pair of C_{α} atoms) and use the distance map as a soft adjacency matrix. From the coordinates, we further derive a number of invariant features. To this end, we use the Frenet framework [31] that uniquely defines any curve in

3D space up to rotation and translation. We compute the curvature of each protein, $\kappa(\mathbf{x})$, and its torsion $\tau(\mathbf{x})$. We use these features, together with one hot embedding, as inputs to our network.

The network can be written as

$$\mathbf{Y}_0 = \mathbf{K}_0 \mathbf{X} \quad (3)$$

$$\mathbf{Y}_{j+1} = \mathbf{Y}_j + \sigma(\mathbf{K}_j \mathbf{Y}_j \mathbf{L}) \quad j = 1, \dots, N \quad (4)$$

$$\mathbf{X}_{\text{out}} = \mathbf{K}_{\text{out}} \mathbf{Y}_N$$

Here, \mathbf{X} , which represents the sequence, is a matrix of $k \times n$, where k is the number of possible amino acids (typically 20), n is the length of the protein, \mathbf{L} is the $n \times n$ graph Laplacian matrix, the weights \mathbf{K}_j are $s \times s$ and \mathbf{K}_{out} is a $2 \times s$ matrix. We choose $\sigma(\cdot)$ to be the leaky Relu function.

The output of the network, \mathbf{X}_{out} , contains two channels. The first is the binding energy of each residue and the second is the RMSD of each residue. When we train the network, we minimize the difference between \mathbf{X}_{out} to the per node energy and structural deviation obtained by the MD simulation.

Thus, if the network is trained successfully, it can approximate both the binding energy and the structural stability of a given sequence; it can therefore be used for protein design without the computational cost of MD simulations.

4. Binary Programming Using Group Sparsity Relaxation

The core of the protein design problem requires solving an optimization problem based on a specific energy. The sequence \mathbf{s} is converted into a $s \times n$ "one hot" matrix \mathbf{S} , where each column represents a residue and each entry is binary. Techniques for solving binary optimization include relaxation and random searches [27–29]. Here, we propose a modification of the concept of group sparsity to approximately solve the optimization problem, as it has been successful in relaxing non-convex problems in the past (see [32–34] and reference within).

To this end, rather than solving for a binary matrix \mathbf{S} that has a single non-zero entry at each column we define the convex set, \mathcal{S} and require the following three properties for every matrix $\mathbf{S} \in \mathcal{S}$:

$$0 \leq \mathbf{S}_{ij} \leq 1 \quad (5)$$

$$\sum_i \mathbf{S}_{ij} = 1 \quad (6)$$

$$\|\mathbf{S}\|_{1,2}^2 = \sum_j \left(\sum_i \mathbf{S}_{ij} \right)^2 \leq n \quad (7)$$

The first two properties simply imply that each element in the matrix is between 0 and 1 and that the sum of each column of the matrix is also 1. The last term in the set is the so called $\ell_{1,2}$ norm and its role is to promote sparsity in each column of the matrix. Since the matrix \mathbf{S} is sparse, holding exactly n non-zero entries, group sparsity yields a structure of \mathbf{S} that is similar to the one obtained from binary programming.

The set \mathcal{S} is convex; therefore, the projection is well defined (see [34] for details). To this end, we define the projection operator as

$$P_{\mathcal{S}}(\mathbf{S}^*) = \arg \min_{\mathbf{S} \in \mathcal{S}} \frac{1}{2} \|\mathbf{S} - \mathbf{S}^*\|^2 \quad (8)$$

The projection takes a matrix \mathbf{S}^* that is outside the convex set \mathcal{S} and solves for a matrix \mathbf{S} that belongs to the set \mathcal{S} . As shown in [34], this can be achieved using simple thresholding. Given the projection $P_{\mathcal{S}}$, we can now use the gradient projection algorithm [35] for the

design problem. The algorithm takes a steepest descent step to minimize the energy followed by a projection to the set \mathcal{S} (see Algorithm 1).

Convergence occurs when \mathbf{S} is a fixed point of the iteration. However, we use another important criterion to terminate the algorithm and exit. Recall that the energy we use in this algorithm is not the true energy but rather a surrogate function. When training the network, we keep track of our validation accuracy and the number of mutations in \mathbf{S} . The network is trained until the number of mutations exceeds a preassigned number. Here, we allow up to 10 mutations in the protein and exit the algorithm when it exceeds this number. While there is no proof that such an algorithm converges to an exact solution, we have found that after a small number of iterations the algorithm does not yield significant improvement in the energy and yields acceptable solutions. These findings are consistent with other applications [36] where similar algorithms have been used. Notably, each iteration of the algorithm requires an evaluation of the energy function, which is prohibitively expensive if MD is used, but feasible with the surrogate model. Thus, practical design is obtained in linear time with respect to the length of the protein.

Algorithm 1 Group Sparsity Design

Require: $\mathbf{S}, \mathbf{X}, niter, \delta_{stab}$
 $n = length(X)$
 $E \leftarrow \infty$
 set $\mathbf{S} \leftarrow P_{\mathcal{S}}(\mathbf{S})$
for $i = 1 \dots niter$ **do**
 Using the network compute $E(\mathbf{S}, \mathbf{X})$ and $\nabla_{\mathbf{S}} E$
 $\mathbf{S} \leftarrow P_{\mathcal{S}}(\mathbf{S} - \mu \nabla_{\mathbf{S}} E)$
 Check for convergence or exit conditions.
end for
 return \mathbf{S}

5. Example: Designing Cardiac Troponin Binders

Cardiac troponin I (cTpnI) is a protein found in heart muscles [37]. Upon damage to cardiac cells such as in acute myocardial infarction (AMI) (commonly known as heart attack), cTpnI is released in the blood [38]. Hence, the detection of cTpnI makes it an excellent biomarker for the diagnosis of AMI [39–41]. In this section, we describe cTpnI binders design using our PINNS approach.

In cardiac cells, cTpnI forms a complex with the following two regulatory proteins: troponin C (TpnC) and troponin T (TpnT) [38]. The crystal structure of the ternary complex is available (Figure 2a) [42]. In blood, however, the biomarker can be found in a complex with either TpnC only or both TpnC and TpnT. Our goal is to design peptides that can specifically bind to cTpnI complexed with TpnC, herein referred to as the target complex. Such a binder can be used for the detection of cTpnI to test for AMI.

5.1. Choice of the Template Peptide

This study is based on the experimentally-resolved structure of the cTpnI-TpnC-TpnT complex, PDB ID: 1J1D [42]. Our first goal was to choose the fragment of TpnT that would be the starting template for the designed binder. An ideal binder would have high affinity and selectivity to cTpnI. Therefore, we aim to find the highest affinity fragment of TpnT to cTpnI-TpnC. Specifically, we ran MD simulations of the troponin complex. The average energy contribution of each residue of TpnT to the total binding energy was calculated to decide the highest affinity fragment to be used as a design template.

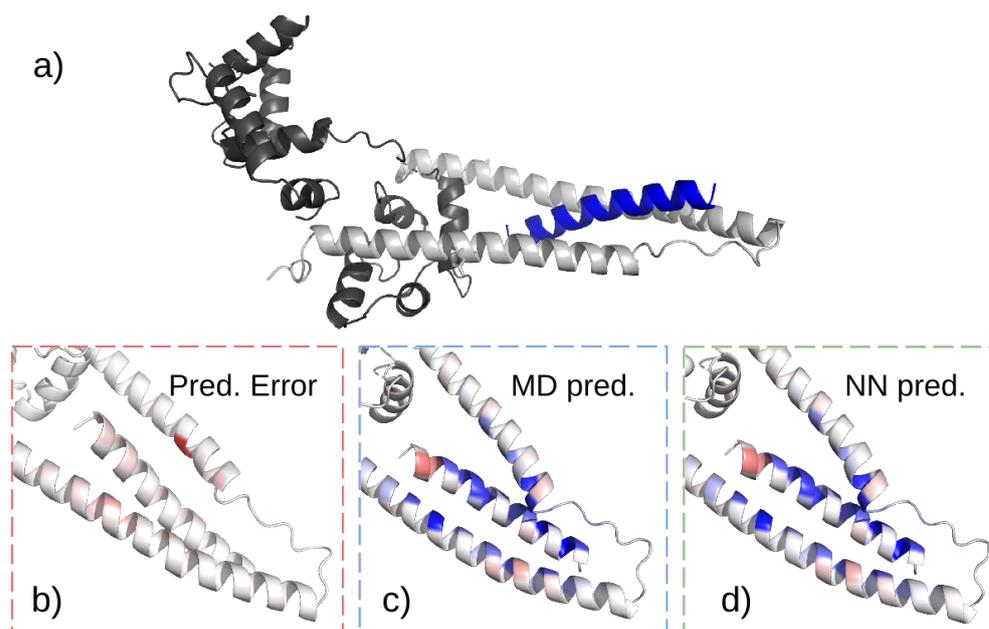


Figure 2. Structure of the Troponin complex. (a) Ribbon representation of a modified Troponin complex. Using a fragment of Troponin T (blue) as a template, the design of a binder for Troponin I (light gray)-Troponin C (dark gray) complex can be optimized. (b) Mean normalized energy prediction error (white_red spectrum, red indicates large errors) between MD prediction (panel (c)) and NN prediction (panel (d)), (both panels are blue_white_red spectrum, blue indicates negative energy values). Note that the overall error is low, as indicated from the minor color difference representing the energy predictions even where the difference (normalized) is the most significant. This means the model can be used for optimization.

To prepare the complex for MD simulations, we first protonated the 1J1D structure using PDB2PQR [43,44] with the Amberff14SB forcefield [45]. This complex was neutralized and solvated using the tleap program of Ambergtools21 [46], in TIP3P 0.15 M NaCl water box. The solvated system was parameterized using the Amberff14SB forcefield [45] and simulated using the NAMD simulations package [47]. First, water molecules and ions were minimized for 1000 steps while the protein complex was restrained. This was followed by 1000 step minimization of the unrestrained system. Next, the backbone-restrained system was gradually heated from 0 to 300 K. The restraints were then gradually decreased until they were completely removed. The non-restrained system was finally simulated for 500 ns. The MMPBSA.py program [48] was used to calculate the Molecular Mechanics Generalized Born Surface Area (MMGBSA) binding energy (BE) of TpnT to the target complex. Generally, the change in Gibbs free energy (ΔG_{bind}) can be approximated as follows:

$$\Delta G_{bind} = \Delta H - T\Delta S \approx \Delta E_{MM} + \Delta G_{sol} - T\Delta S \quad (9)$$

in which ΔH is the change in enthalpy, T is the temperature, ΔS is the change in entropy, ΔE_{MM} is the molecular mechanics energy in the gas phase and ΔG_{sol} is the solvation free energy. BE is computed using MMPBSA.py and is calculated as follows:

$$BE = \Delta E_{MM} + \Delta G_{sol} \quad (10)$$

$$\Delta E_{MM} = \Delta E_{inter} + \Delta E_{elec} + \Delta E_{vdW} \quad (11)$$

BE is a sum of E_{inter} , which is the internal energy of the molecule bond distances, angles and dihedrals, as well as ΔE_{elec} and ΔE_{vdW} , which are the electrostatic and van der Waals energies. Note that the entropic change is not accounted for in this calculation since such a calculation is computationally demanding and inaccurate.

BE was decomposed to each residue of TpnT and it was decided that residues 104–130 would constitute the starting template for our peptide design, herein referred to as the TpnT-template.

5.2. Generating Training Data

While it is possible to randomly sample peptide structures and compute their trajectories, it is better to sample structures that are considered feasible, that is, those structures that do not diverge when running molecular dynamics. As explained in the introduction, there are many software packages that can provide an initial design using either empirical energy or machine learning. Here, we use the popular Rosetta package and its protein complex generation process [7].

The proteins cTpnI and TpnC complexed with residues 104–130 of TpnT were first repacked and minimized based on the Beta_nov16 scoring function [49]. This was followed by Monte-Carlo based mutagenesis of the design peptide; in this step, the electrostatic contribution of the Beta_nov16 scoring function was up-weighted by a factor of 1.5. This modification to the scoring function biases mutagenesis towards residues that would form more hydrogen bonds, as this would increase the designed peptide specificity. The mutant fragment was then redocked to the target complex and minimized before the final complex structure was output. A total of 15,000 design attempts were made using Rosetta. Only about 2600 designs passed the following initial screening criteria: general design score of -190 Rosetta Energy Units (REUs), a binding energy of -50 REU, a shape complementarity score of 0.75 and no more than 3 buried unsatisfied hydrogen bonds. These 2600 structures were further filtered using MD simulations. Each designed peptide complexed with cTpnI-TpnC was prepared for MD simulations, as described in Section 5.1. Each complex was neutralized and solvated in a TIP3P 0.15 M NaCl water box. The backbone-restrained system was minimized for 1000 steps followed by another 1000 minimization steps without restraints. The backbone-restrained system was then heated from 0 to 300 K; then, the restraints were incrementally removed at 300 K. The unrestrained complex was simulated for 5 ns. Data from 5 ns simulations were extracted for the training of our algorithm.

Three pieces of data were used to train our algorithm.

- Complex coordinates were extracted from the 5 ns simulations at regular intervals for a total of 50 coordinate sets per designed peptide. These were used to compute the center of mass coordinates, $\mathbf{x}^*(\mathbf{s})$, as well as standard coordinate deviation during simulations, $\mathbf{r}(\mathbf{s})$, of the trajectory.
- The average MMGBSA binding energies of the designed peptides to the target complexes, representing $E^*(s)$, were computed using MMPBSA.py [48].
- The contribution of each individual residue to the average MMGBSA binding energy for each complex was also calculated.

As part of trajectory analysis, we also computed the MMGBSA binding energies for the 2600 complexes over the 5 ns simulation time. While more accurate methods for calculating binding energies exist, such as thermodynamic integration [50,51] and free energy perturbation [52], these methods are computationally demanding and would be impractical for calculating the binding energy of thousands of complexes. Furthermore, previous studies have shown that the Molecular Mechanics Poisson Boltzmann Surface Area (MMPBSA) method for calculating the binding energy was comparable to MMGBSA, with the latter being less computationally expensive [53,54].

We also calculated the RMSD of the design atomic coordinates over the simulation time. In Figure 3, we plot the RMSD as a function of time for 10 random designs. The RMSD of some structures continued increasing, which implying that they were unstable. For other structures, however, we observe an initial structural drift followed by a plateau in RMSD values. It is important to note that 5 ns is a short time to deem whether RMSD plateauing determines structural stability. However, such short simulation times provide a means of filtering initial designs within reasonable computational costs. RMSD values were condensed to the vector \mathbf{r} with Equation (1a), and were predicted using the network.

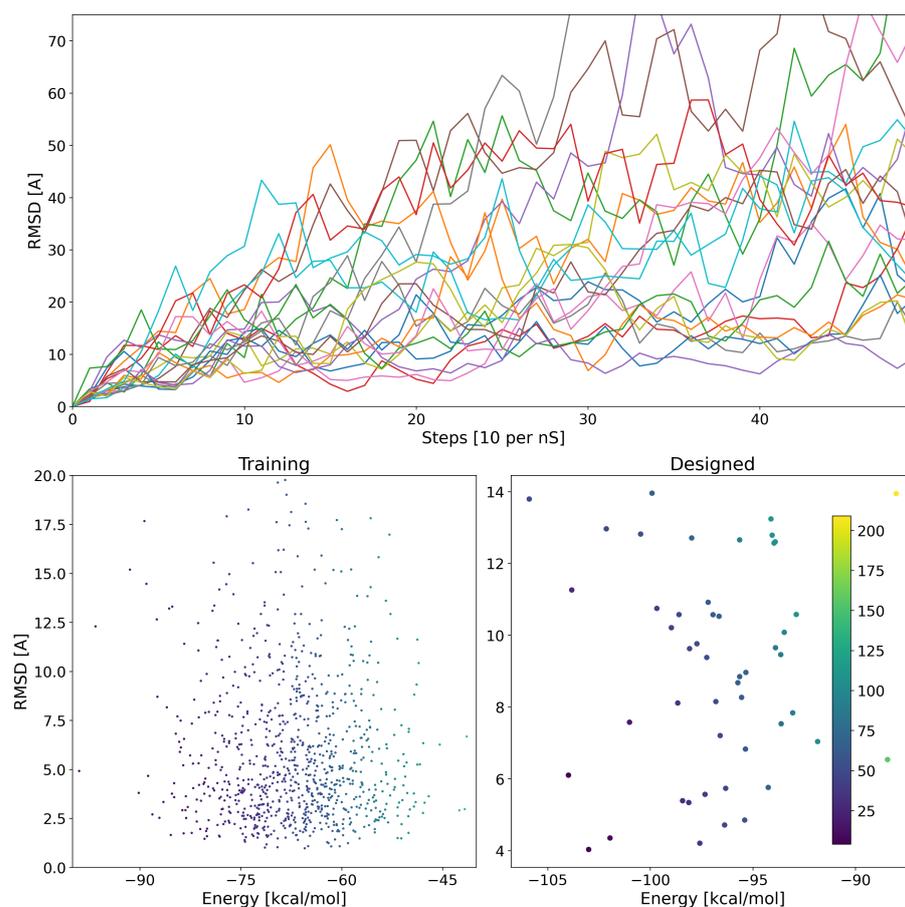


Figure 3. Top panel: The trajectories of RMSD of 10 random complexes over 5ns. It is evident that some designs are more stable and remain within their original conformation from design (low RMSD), whereas others deviate greatly from their original design coordinates (high RMSD). Bottom panel: The binding energy of different complexes and their corresponding coordinate RMSD. The best designs (low energy and stable) are plotted in blue while less reliable ones become more yellow.

An ideal design would have a low binding energy and would not considerably deviate from the original design coordinates. We plot the MMGBSA binding energy of 950 structures versus their RMSD in Figure 3. It can be observed that there is no correlation between a designed peptide average MMGBSA binding energy and its structural stability during the simulation. Hence, it is important to compute and evaluate both attributes in the complex to assess design success.

5.3. Training the Model

Given the training data, we trained the graph neural network described in Section 3. In the first step, we take the energy and the stability data, subtract the mean and divide by the standard deviation. This is a common practice in most training procedures [55]. The network has a total of roughly 1.3 M parameters, and we use 5 layers of the graph network. The opening layer, K_o , increases the size of the network from 23 channels (20 for the one hot sequence and 3 for the geometrical features) to 512 channels. Using this architecture, we manage to reduce the error of the validation data set to 9.8×10^{-3} . The maximal error in training is plotted in Figure 2.

5.4. Using the Deep Learning Model for Protein Design

Given the initial 950 sequences, we chose the best 50 sequences and used the optimization Algorithm 1. We used a total of 5 mutations in each protein before the optimization

algorithm returned. A graph of the energy as a function of iterations as well as the number of total mutations (the number of mutations in all designs) is presented in Figure 4.

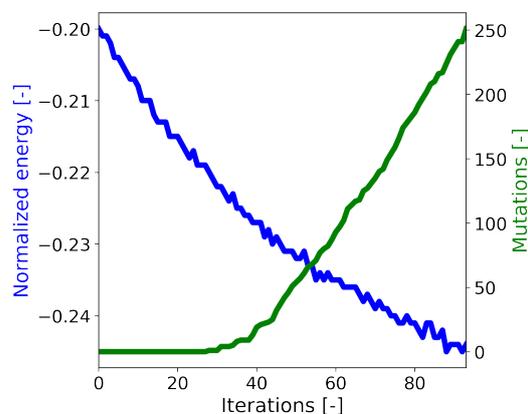


Figure 4. Convergence of the optimization algorithm and the number of mutations observed by using Algorithm 1. Note that the presented energy is the normalized one (de-biased). The code is terminated when 5 mutations are proposed on average, leading the overall mutations over 50 designs to amount to 250.

The initial 50 sequences and the designs obtained by using the algorithm are plotted in Figure 5. We observe a much larger variability of designs compared with the initial designs.

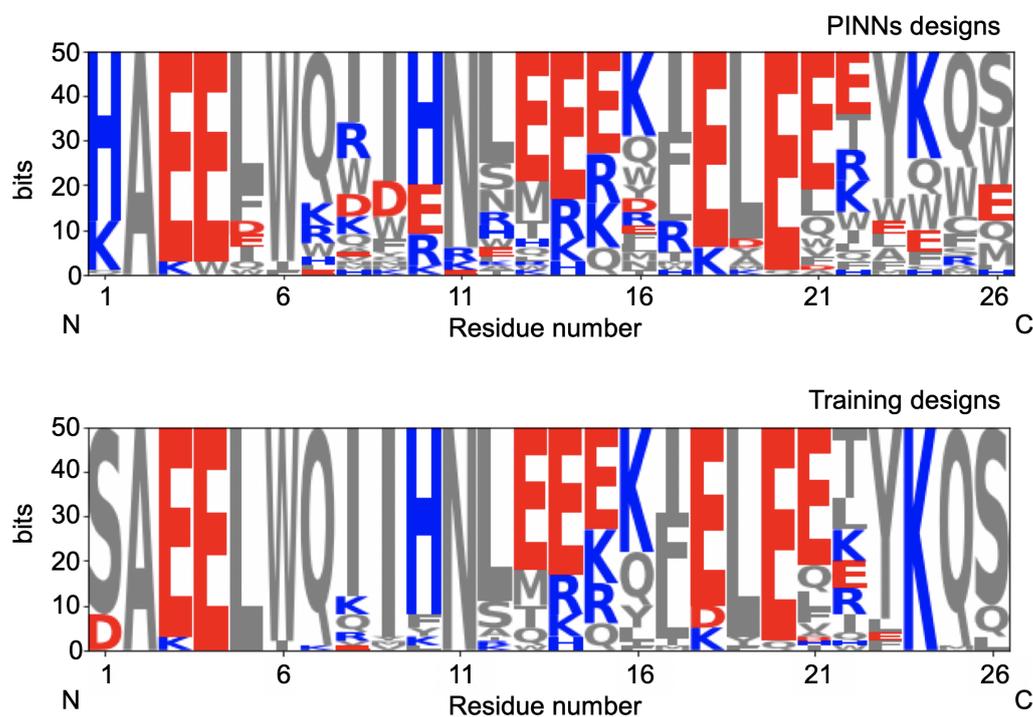


Figure 5. Design sequence logos of the 950 designs from the training set (**bottom**) and the PINNs-generated designs (**top**). Blue, red and grey colors indicate that the residues are positively charged, negatively charges or uncharged, respectively. It is evident that the PINNs designs are more diverse compared to the initial designs from the training set. Since the problem is highly non-convex, this allows us to generate many different sequences with different starting points.

The newly designed proteins are then used as validation data. The energy of the designed proteins as well as their structural stability is plotted in Figure 3. We see an overall agreement between the predicted and observed energies and stability.

6. Summary

In this work, we proposed a new framework that combines MD, deep learning, and binary programming optimization that leverage the strengths of each to solve the protein design task. Our goal is to use the most accurate energy model possible in order to achieve a stable design. To this end, we use MD to account for the protein environment and flexibility, since proteins are in constant motion. The MD trajectories are used for the calculation of an average binding energy. To assess the structural stability of the protein, we also record the deviation of the designed protein from the desired design. The average binding energy and RMSD of each sequence are then approximated using a graph neural network. The goal of the network is to recover the average binding energy and the average deviation of the design, which is more achievable than recovering the whole simulation. This approach is commonly referred to as PINNS and has gained popularity in many science and engineering problems. We train a graph neural network on data that are generated using MD. The resulting network is then used for solving the protein sequence design problem. For optimization, we use a convex relaxation of the problem with a sparsity-promoting norm. This yields sparse designs that are used to probe the energy surface. The code was tested on a true design of a binder to troponin. We have shown that we are able to find designs that indeed have a low binding energy (as computed by MD) given the trained data. Our designs reduce the energy by an average of 20% while keeping the protein stable. Our results yield a more diverse set of sequences compared with the initial designs, which will facilitate the laboratory testing of many more variants compared with the initial variants.

Our method heavily relies on MD, which accounts for protein flexibility. It is worth noting that simulation conditions can be set to facilitate the evaluation of designed proteins at different temperatures. Furthermore, different pH values may be considered by adjusting residue protonation. We assumed in this procedure that if the structure for a given sequence stably binds to the target protein with high affinity then it will manifest this prediction in the lab. On the other hand, if the simulations were not reliable our design would lead to unreliable binder sequence predictions. Clearly, any final design needs to be validated in the laboratory. It is important to note that 5 ns simulations are too short to assess the structural stability of peptide–protein complexes. Longer simulation times could be employed to address this issue although this would be computationally expensive. Alternatively, longer simulations could be obtained for filtered designs that have an initially low RMSD. Validating MD results against laboratory experiments is beyond the scope of this paper. However, recent computational techniques are proposed to obtain an even higher accuracy for MD when solving the full Schrödinger system. Our approach, which can be used with any energy function, is designed with the aim of reducing the need for laboratory experiments by improving the accuracy of protein design simulations.

Author Contributions: Conceptualization, E.H., A.J.B.-S. and S.I.O.; methodology, E.H. and S.I.O.; software, S.I.O., E.H. and A.J.B.-S.; validation, E.H., S.I.O., A.J.B.-S. and C.K.; formal analysis, E.H., S.I.O. and A.J.B.-S.; investigation, E.H., S.I.O., A.J.B.-S. and C.K.; resources, E.H.; data curation, S.I.O. and E.H.; writing—original draft preparation, E.H., S.I.O. and A.J.B.-S.; writing—review and editing, E.H., S.I.O., C.K. and A.J.B.-S.; visualization, E.H. and A.J.B.-S.; supervision, E.H.; funding acquisition, E.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by NSERC Discovery Grant, and grant no. 696/22 from the Israel Science Foundation (ISF).

Data Availability Statement: Data is available on a Github repository.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Dahiyat, B.I.; Mayo, S.L. De novo protein design: Fully automated sequence selection. *Science* **1997**, *278*, 82–87. [[CrossRef](#)] [[PubMed](#)]
2. Pan, X.; Kortemme, T. Recent advances in de novo protein design: Principles, methods, and applications. *J. Biol. Chem.* **2021**, *296*. [[CrossRef](#)] [[PubMed](#)]
3. Hsu, C.; Verkuil, R.; Liu, J.; Lin, Z.; Hie, B.; Sercu, T.; Lerer, A.; Rives, A. Learning inverse folding from millions of predicted structures. *bioRxiv* **2022**, preprint. [[CrossRef](#)]
4. Dauparas, J.; Anishchenko, I.; Bennett, N.; Bai, H.; Ragotte, R.J.; Milles, L.F.; Wicky, B.I.; Courbet, A.; de Haas, R.J.; Bethel, N.; et al. Robust deep learning-based protein sequence design using ProteinMPNN. *Science* **2022**, *378*, 49–56. [[CrossRef](#)] [[PubMed](#)]
5. Ingraham, J.; Garg, V.; Barzilay, R.; Jaakkola, T. Generative Models for Graph-Based Protein Design. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R., Eds.; Curran Associates, Inc.: New York, NY, USA, 2019; Volume 32.
6. Lippow, S.M.; Tidor, B. Progress in computational protein design. *Curr. Opin. Biotechnol.* **2007**, *18*, 305–311. [[CrossRef](#)]
7. Loshbaugh, A.L.; Kortemme, T. Comparison of Rosetta flexible-backbone computational protein design methods on binding interactions. *Proteins Struct. Funct. Bioinform.* **2020**, *88*, 206–226. [[CrossRef](#)]
8. Baker, D.; Sali, A. Protein structure prediction and structural genomics. *Science* **2001**, *294*, 93–96. [[CrossRef](#)]
9. Eliasof, M.; Boesen, T.; Haber, E.; Keasar, C.; Treister, E. Mimetic neural networks: A unified framework for protein design and folding. *arXiv* **2021**, arXiv:2102.03881.
10. Ferruz, N.; Schmidt, S.; Höcker, B. ProtGPT2 is a deep unsupervised language model for protein design. *Nat. Commun.* **2022**, *13*, 4348. [[CrossRef](#)]
11. Baker, D. What has de novo protein design taught us about protein folding and biophysics? *Protein Sci.* **2019**, *28*, 678–683. [[CrossRef](#)]
12. Pikkemaat, M.G.; Linssen, A.B.; Berendsen, H.J.; Janssen, D.B. Molecular dynamics simulations as a tool for improving protein stability. *Protein Eng.* **2002**, *15*, 185–192. [[CrossRef](#)]
13. Lindorff-Larsen, K.; Maragakis, P.; Piana, S.; Eastwood, M.P.; Dror, R.O.; Shaw, D.E. Systematic validation of protein force fields against experimental data. *PLoS ONE* **2012**, *7*, e32131. [[CrossRef](#)]
14. Piana, S.; Robustelli, P.; Tan, D.; Chen, S.; Shaw, D.E. Development of a force field for the simulation of single-chain proteins and protein–protein complexes. *J. Chem. Theory Comput.* **2020**, *16*, 2494–2507. [[CrossRef](#)]
15. Cai, D. Physics-informed distribution transformers via molecular dynamics and deep neural networks. *J. Comput. Phys.* **2022**, *468*, 111511. [[CrossRef](#)]
16. Wang, Y.; Ribeiro, J.M.L.; Tiwary, P. Machine learning approaches for analyzing and enhancing molecular dynamics simulations. *Curr. Opin. Struct. Biol.* **2020**, *61*, 139–145. [[CrossRef](#)]
17. Cozad, A.; Sahinidis, N.V.; Miller, D.C. Learning surrogate models for simulation-based optimization. *AIChE J.* **2014**, *60*, 2211–2227. [[CrossRef](#)]
18. Trincherro, R.; Larbi, M.; Torun, H.M.; Canavero, F.G.; Swaminathan, M. Machine learning and uncertainty quantification for surrogate models of integrated devices with a large number of parameters. *IEEE Access* **2018**, *7*, 4056–4066. [[CrossRef](#)]
19. Tang, M.; Liu, Y.; Durlofsky, L.J. A deep-learning-based surrogate model for data assimilation in dynamic subsurface flow problems. *J. Comput. Phys.* **2020**, *413*, 109456. [[CrossRef](#)]
20. Gal, R.; Haber, E.; Ziv, A. Using dnns and smart sampling for coverage closure acceleration. In Proceedings of the 2020 ACM/IEEE Workshop on Machine Learning for CAD, Online, 16–20 November 2020; pp. 15–20.
21. Karplus, M.; McCammon, J.A. Molecular dynamics simulations of biomolecules. *Nat. Struct. Biol.* **2002**, *9*, 646–652. [[CrossRef](#)]
22. Klepeis, J.L.; Lindorff-Larsen, K.; Dror, R.O.; Shaw, D.E. Long-timescale molecular dynamics simulations of protein structure and function. *Curr. Opin. Struct. Biol.* **2009**, *19*, 120–127. [[CrossRef](#)]
23. Plattner, N.; Doerr, S.; De Fabritiis, G.; Noé, F. Complete protein–protein association kinetics in atomic detail revealed by molecular dynamics simulations and Markov modelling. *Nat. Chem.* **2017**, *9*, 1005–1011. [[CrossRef](#)] [[PubMed](#)]
24. Swope, W.C.; Pitera, J.W.; Suits, F. Describing protein folding kinetics by molecular dynamics simulations. 1. Theory. *J. Phys. Chem. B* **2004**, *108*, 6571–6581. [[CrossRef](#)]
25. Wang, J.; Lisanza, S.; Juergens, D.; Tischer, D.; Watson, J.L.; Castro, K.M.; Ragotte, R.; Saragovi, A.; Milles, L.F.; Baek, M.; et al. Scaffolding protein functional sites using deep learning. *Science* **2022**, *377*, 387–394. [[CrossRef](#)] [[PubMed](#)]
26. Strokach, A.; Becerra, D.; Corbi-Verge, C.; Perez-Riba, A.; Kim, P.M. Fast and flexible protein design using deep graph neural networks. *Cell Syst.* **2020**, *11*, 402–411. [[CrossRef](#)] [[PubMed](#)]
27. Theunissen, T. Binary programming and test design. *Psychometrika* **1985**, *50*, 411–420. [[CrossRef](#)]
28. Al-Tashi, Q.; Kadir, S.J.A.; Rais, H.M.; Mirjalili, S.; Alhussian, H. Binary optimization using hybrid grey wolf optimization for feature selection. *IEEE Access* **2019**, *7*, 39496–39508. [[CrossRef](#)]
29. Okuyama, T.; Sonobe, T.; Kawarabayashi, K.i.; Yamaoka, M. Binary optimization by momentum annealing. *Phys. Rev. E* **2019**, *100*, 012111. [[CrossRef](#)]
30. Eliasof, M.; Haber, E.; Treister, E. Pde-gcn: Novel architectures for graph neural networks motivated by partial differential equations. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 3836–3849.

31. Spivak, M. *A Comprehensive Introduction to Differential Geometry*; Publish or Perish, Incorporated; Houston, TX, USA, 1975; Volume 4.
32. Ghosh, A.; McCann, M.T.; Mitchell, M.; Ravishankar, S. Learning Sparsity-Promoting Regularizers using Bilevel Optimization. *arXiv* **2022**, arXiv:2207.08939.
33. Chen, P.Y.; Selesnick, I.W. Group-sparse signal denoising: Non-convex regularization, convex optimization. *IEEE Trans. Signal Process.* **2014**, *62*, 3464–3478. [[CrossRef](#)]
34. Deng, W.; Yin, W.; Zhang, Y. Group sparse optimization by alternating direction method. In Proceedings of the Wavelets and Sparsity XV, San Diego, CA, USA, 26 September 2013; SPIE: Bellingham, WA, USA, 2013; Volume 8858, pp. 242–256.
35. Nocedal, J.; Wright, S. *Numerical Optimization*; Springer: New York, NY, USA, 1999.
36. Donoho, D.L.; Elad, M. Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ_1 minimization. *Proc. Natl. Acad. Sci. USA* **2003**, *100*, 2197–2202. [[CrossRef](#)]
37. Sarko, J.; Pollack Jr, C.V. Cardiac troponins. *J. Emerg. Med.* **2002**, *23*, 57–65. [[CrossRef](#)]
38. Practice Guidelines (CPG), E.C.; Netherlands), J.J.B.C.C.; (Germany), H.B.; (Italy), C.C.; (France), V.D.; (UK), C.D.; (Belgium), R.F.; (France), C.F.B.; (Israel), D.H.; Netherlands), A.H.T.; et al. Third universal definition of myocardial infarction. *J. Am. Coll. Cardiol.* **2012**, *60*, 1581–1598.
39. Fathil, M.; Arshad, M.M.; Gopinath, S.C.; Hashim, U.; Adzhri, R.; Ayub, R.; Ruslinda, A.; Nuzaihan, M.; Azman, A.; Zaki, M.; et al. Diagnostics on acute myocardial infarction: Cardiac troponin biomarkers. *Biosens. Bioelectron.* **2015**, *70*, 209–220. [[CrossRef](#)]
40. Babuin, L.; Jaffe, A.S. Troponin: The biomarker of choice for the detection of cardiac injury. *Cmaj* **2005**, *173*, 1191–1202. [[CrossRef](#)]
41. Cullen, L.A.; Mills, N.L.; Mahler, S.; Body, R. Early rule-out and rule-in strategies for myocardial infarction. *Clin. Chem.* **2017**, *63*, 129–139. [[CrossRef](#)]
42. Takeda, S.; Yamashita, A.; Maeda, K.; Maéda, Y. Structure of the core domain of human cardiac troponin in the Ca²⁺-saturated form. *Nature* **2003**, *424*, 35–41. [[CrossRef](#)]
43. Dolinsky, T.J.; Nielsen, J.E.; McCammon, J.A.; Baker, N.A. PDB2PQR: An automated pipeline for the setup of Poisson–Boltzmann electrostatics calculations. *Nucleic Acids Res.* **2004**, *32*, W665–W667. [[CrossRef](#)]
44. Dolinsky, T.J.; Czodrowski, P.; Li, H.; Nielsen, J.E.; Jensen, J.H.; Klebe, G.; Baker, N.A. PDB2PQR: Expanding and upgrading automated preparation of biomolecular structures for molecular simulations. *Nucleic Acids Res.* **2007**, *35*, W522–W525. [[CrossRef](#)]
45. Maier, J.A.; Martinez, C.; Kasavajhala, K.; Wickstrom, L.; Hauser, K.E.; Simmerling, C. ff14SB: Improving the accuracy of protein side chain and backbone parameters from ff99SB. *J. Chem. Theory Comput.* **2015**, *11*, 3696–3713. [[CrossRef](#)]
46. Case, D.A.; Aktulga, H.M.; Belfon, K.; Ben-Shalom, I.; Brozell, S.R.; Cerutti, D.S.; Cheatham, T.E., III; Cruzeiro, V.W.D.; Darden, T.A.; Duke, R.E.; et al. *Amber 2021*; University of California: San Francisco, CA, USA, 2021.
47. Phillips, J.C.; Hardy, D.J.; Maia, J.D.; Stone, J.E.; Ribeiro, J.V.; Bernardi, R.C.; Buch, R.; Fiorin, G.; Hénin, J.; Jiang, W.; et al. Scalable molecular dynamics on CPU and GPU architectures with NAMD. *J. Chem. Phys.* **2020**, *153*, 044130. [[CrossRef](#)] [[PubMed](#)]
48. Miller III, B.R.; McGee Jr, T.D.; Swails, J.M.; Homeyer, N.; Gohlke, H.; Roitberg, A.E. MMPBSA.py: An efficient program for end-state free energy calculations. *J. Chem. Theory Comput.* **2012**, *8*, 3314–3321. [[CrossRef](#)] [[PubMed](#)]
49. Pavlovicz, R.E.; Park, H.; DiMaio, F. Efficient consideration of coordinated water molecules improves computational protein-protein and protein-ligand docking discrimination. *PLoS Comput. Biol.* **2020**, *16*, e1008103. [[CrossRef](#)] [[PubMed](#)]
50. Beveridge, D.L.; Dicapua, F.M. Free energy via molecular simulation: Applications to chemical and biomolecular systems. *Annu. Rev. Biophys. Chem.* **1989**, *18*, 431–492. [[CrossRef](#)] [[PubMed](#)]
51. Straatsma, T.; McCammon, J. Computational alchemy. *Annu. Rev. Phys. Chem.* **1992**, *43*, 407–435. [[CrossRef](#)]
52. Zwanzig, R.W. High-temperature equation of state by a perturbation method. I. Nonpolar gases. *J. Chem. Phys.* **1954**, *22*, 1420–1426. [[CrossRef](#)]
53. Hou, T.; Wang, J.; Li, Y.; Wang, W. Assessing the performance of the MM/PBSA and MM/GBSA methods. 1. The accuracy of binding free energy calculations based on molecular dynamics simulations. *J. Chem. Inf. Model.* **2011**, *51*, 69–82. [[CrossRef](#)]
54. Rastelli, G.; Rio, A.D.; Degliesposti, G.; Sgobba, M. Fast and accurate predictions of binding free energies using MM-PBSA and MM-GBSA. *J. Comput. Chem.* **2010**, *31*, 797–810. [[CrossRef](#)]
55. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.