Differentiable molecular simulation can learn all the parameters in a coarse-grained force field for proteins

Joe G Greener, David T Jones

Department of Computer Science, University College London,
Gower Street, London WC1E 6BT, UK

Correspondence: j.greener@ucl.ac.uk

Abstract

Finding optimal parameters for force fields used in molecular simulation is a challenging and time-consuming task, partly due to the difficulty of tuning multiple parameters at once. Automatic differentiation presents a general solution: run a simulation, obtain gradients of a loss function with respect to all the parameters, and use these to improve the force field. This approach takes advantage of the deep learning revolution whilst retaining the interpretability and efficiency of existing force fields. We demonstrate that this is possible by parameterising a simple coarse-grained force field for proteins, based on training simulations of up to 2,000 steps learning to keep the native structure stable. The learned potential matches chemical knowledge and PDB data, can fold and reproduce the dynamics of small proteins, and shows ability in protein design and model scoring applications. Problems in applying differentiable molecular simulation to all-atom models of proteins are discussed along with possible solutions. The learned potential, simulation scripts and training code are made available at `https://github.com/psipred/cgdms`.

Introduction

Molecular simulation has been successful in making predictions and understanding experimental data [1, 2]. Treating the system with an appropriate level of complexity, usually all-atom molecular mechanics or residue-level coarse-graining when simulation proteins, is necessary to access the timescales required for the property under investigation [3, 4]. It is generally agreed that force fields are not optimally parameterised [5, 6], for example significant effort has gone into modifying a handful of parameters in standard force fields to better represent both ordered and disordered proteins [7, 8, 9]. Such efforts have improved the force fields without changing their functional form, an attractive proposition when the alternative is adding complexity that restricts the timescales available for study.

Meanwhile, deep learning has had a major impact on many areas of biology, achieving state of the art performance in fields such as protein structure prediction [10]. A number of groups have applied these advances to molecular simulations [11, 12, 13] including learning coarse-grained potentials [14, 15, 16, 17, 18], learning quantum mechanical potentials [19, 20, 21, 22], improving sampling [23], and improving atom typing [24]. Whilst promising, many of these approaches show limited success when used on systems they were not trained on. Methods trained on trajectory data also suffer from a lack of standardised simulations across many systems due to the high cost of obtaining such trajectories. Other groups have used inventive machine learning strategies for end-to-end protein structure prediction [25] and to score static conformations [26, 27].

The idea of differentiating through the numerical solution of Newton's equations of motion to obtain gradients that can be used to improve a learned force field has been discussed [28], but is yet to produce generally useful force fields. Conceptually the idea is related to recent work on neural differential equations [29, 30]. The idea is appealing because a large number of parameters can be improved at once, rather than the small numbers currently modified. The gradients are also exact, at least with respect to the limitations of the numerical integration and the loss function used. Only recently has the hardware and software been available to run such simulations.

Previous studies utilising differentiable molecular simulation (DMS) have trained neural network potentials, ranging from graph neural networks [28] to ambitious protein folding simulators running Langevin dynamics [31]. Whilst training neural networks may be an effective solution, as discussed above there is much room for improvement in existing force fields. It makes sense to try and improve these where possible rather than moving to a new functional form, which has the additional advantage of retaining the physical interpretability of conventional force fields. Neural networks are also slower to run than existing force fields, meaning that to avoid reducing available simulation time the network has to be trained to jump multiple time steps, a rather different problem to learning the instantaneous potential. This work is also influenced by a number of studies that compare native and training ensembles to improve force fields for protein folding using maximum likelihood and contrastive divergence approaches [32, 33, 34, 35, 36]. These methods are able to modify many parameters at once, but they generally involve comparing conformations to distinguish ensembles rather than obtaining gradients of some loss function through the simulation.

In this study we use automatic differentiation (AD) [37], the procedure used to train neural networks where it is called backpropagation, to learn all the parameters from scratch in a simple coarse-grained force field for proteins. This learned potential matches potentials derived from chemical knowledge and Protein Data Bank (PDB) statistics, reproduces native flexibility when used in simulation, is able to fold a set of small proteins not used for training, and shows promise for protein design and model scoring applications. It adds to existing coarse-grained potentials used for simulation [38, 39] and model scoring approaches [40, 41, 42]. More broadly, it points to DMS as a useful technology falling under the banner of differentiable programming [43], an expansion of the principles of deep learning to the concept of taking gradients through arbitrary algorithms.

Results

Differentiable molecular simulation

In this study a coarse-grained potential for proteins is learned in which a protein is represented by 4 point particles per residue (N, Cα, C and sidechain centroid) with no explicit solvent. The potential consists of 3 components: pairwise distance potentials (including covalent bonds), bond angle potentials, and torsion angle potentials associated with the predicted secondary structure type of a residue. Overall there are 29,360 individual potentials and 4.1m learnable parameters. The high number of parameters indicates some redundancy due to the nature of the potentials, but also demonstrates that DMS can be used to learn a large number of parameters at once. See the methods for further details on the functional form of the potentials, how forces are calculated and how the model is trained. Proteins presented in the results do not have homologs present in the training set and single sequence secondary structure prediction is used throughout the study, so the results are not due to overtraining or direct learning of evolutionary information. Assessing the method on proteins not used for training distinguishes this approach from many approaches used to date for machine learning of molecular simulations.

During training, proteins are simulated using the velocity Verlet integrator in the NVE ensemble, i.e. with no thermostat. The starting conformation is the native structure and up to 2,000 steps are run. Due to shared parameters at each time step this can be thought of as analogous to a recurrent neural network (RNN) running on a sequence of length 2,000, as shown in Figure 1A. In particular, implementing the simulation in the AD framework PyTorch [44] allows the gradients of a given loss function with respect to each learned parameter to be calculated, allowing an optimiser to change the parameters to reduce the loss function. However, the learned parameters are those of the force field rather than the weights and biases of a standard neural network. $\log(1+R_f)$ is used as the loss function, where R_f is the root-mean-square deviation (RMSD) across all atoms in the coarse-grained model between the conformation at the end of the simulation and the native structure. At the start of learning the potentials are flat, the forces are zero and the proteins distort according to the randomised starting velocities, as shown in Figure 1B. Over the course of training, the R_f values decrease as the potential learns to stabilise the proteins in the training set. The training and validation R_f values throughout training are shown in Figure S1.

After training on a dataset of 2,004 diverse proteins up to 100 residues long the potentials resemble those derived from chemical knowledge and PDB statistics, as shown in Figure 2. Due to the coarse-grained nature of the simulation the energy values, along with other properties such as the time step and the temperature used later in the thermostat, cannot be assigned standard units. Covalent bond distance potentials have strong minima at the correct distance and steep barriers preventing steric clashing. The steep drops at the edges are an artifact of training and do not affect simulations or energy scoring, since these values are never occu-

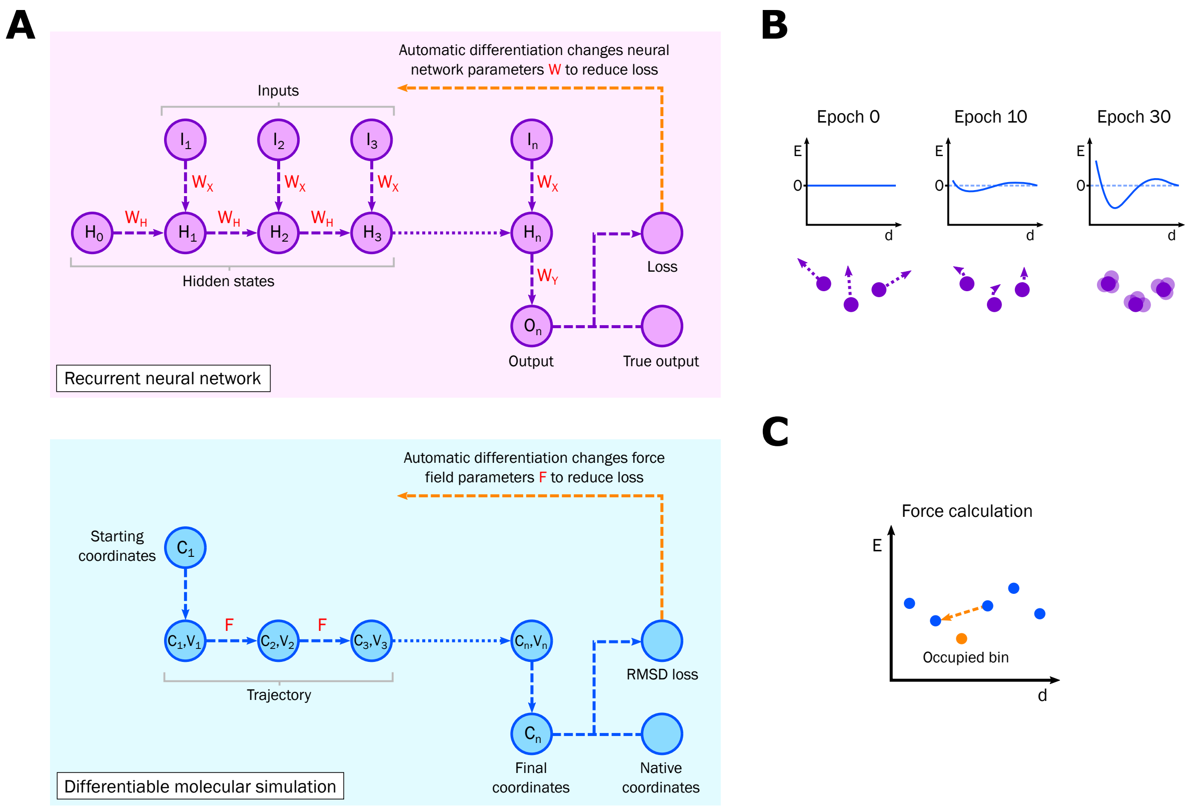


Figure 1 Differentiable molecular simulation. (A) The analogy between a RNN and DMS. Learnable parameters are shown in red. The same parameters are used at each step. There are many variants of RNNs; the architecture shown here has a single output for a variable length input, which could for example represent sentiment classification of a series of input words. (B) Learning the potential. A representation of a component of the potential is shown with energy E plotted against inter-atomic distance d. At the start of training the potential is flat and the atoms deform according to their starting velocities. During training the potential learns to stabilise the native structures of the training set. (C) Force calculation from the potential. Adjacent bins to the occupied bin are used to derive the force using finite differences. In this case the force acts to reduce the distance d. See the methods for more details.

pied when using the trained potential. Bond angle potentials have minima at the true values with a few degrees of tolerance allowed either side. Same residue Cα-sidechain distance potentials indicate that different rotamer conformations have been learned; Figure 2C shows that the energy minima for isoleucine agree with the two minima found in the PDB distance distributions, which correspond to different rotamers. Other cases showing different rotamer conformations include glutamic acid, glutamine, lysine, methionine and tryptophan. These can be seen in the complete set of such potentials shown in Figure S2. The torsion angle potentials show different preferences for residues predicted as α-helical, β-sheet and coiled, and these agree with the true Ramachandran distributions. The glycine torsion angle potentials indicate lower energy regions in the positive φ space and the proline potentials display a minimum at $0°$ for the ω angle corresponding to cis-proline (data not shown). The potentials most important for the tertiary structure are the general distance potentials. As shown in Figure 2E these match potentials of mean force (PMFs) derived from the PDB in many cases, with minima for many pairs around 6 Å driving hydrophobic packing. The steep energy barriers to steric clashing do not generally extend below 4 Å because extending these is not required to improve R_f during training. Steric clashing is not seen during simulations provided that a suitable time step is used. The complete set of potentials for general sidechain-sidechain distances are shown in Figure S3.

We find that the learned potential has surprising local detail, despite being trained only to minimise the final RMSD. The potential energy when modifying the φ and ψ torsion angles of alanine dipeptide is shown in Figure 3A. Also shown is the free energy calculated from an all-atom simulation in Wang et al. 2019 [15]. The learned potential matches the major low energy conformations of the all-atom model.

Protein structure and dynamics

A learned potential can be used to run a simulation of arbitrary length since gradients are not recorded. Here we study four small, fast-folding proteins investigated with molecular simulation previously [45, 8, 46]. They all have NMR ensembles or crystal structures available from experiments [47, 48]. Details on all proteins presented in the results are in Table S1. First we investigate whether the learned potential can keep the native structures stable and reproduce residue-level flexibility. As shown in Figure 4B, the native structures are stable under simulation in the NVT ensemble using the Andersen thermostat, with Cα RMSDs less than 4 Å in all cases. Figure 3B shows that the root-mean-square fluctuation (RMSF) of the Cα atom of each residue over the simulation generally matches that of the native NMR ensembles, with an expected increase in flexibility for terminal residues. Chignolin displays more flexility under simulation than in the NMR ensemble, likely due to the lack of explicit hydrogen bonding in the coarse-grained model to keep the beta-turn structure rigid. For villin HP36, we see general agreement between the residue RMSF and the crystal temperature factors.

Next, we ask whether we can fold small proteins and peptides in the NVT ensemble. We find that the $(AAQAA)_3$ repeat peptide folds into an α-helix over 12m steps when started from a random conformation, accompanied by a reduction in the energy under the learned potential.

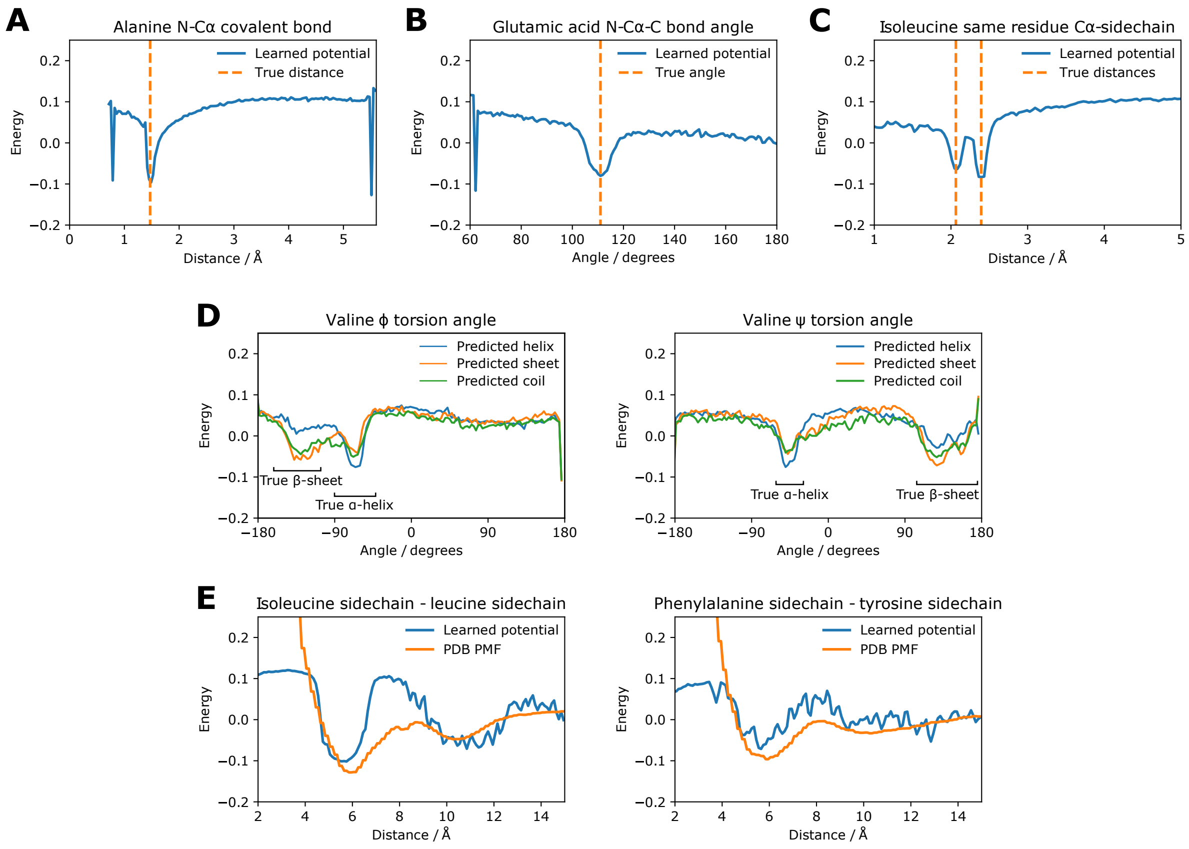


Figure 2 Components of the learned coarse-grained potential compared to true values. The energy scale is the same for each plot. Each individual potential consists of discrete energy values for 140 evenly-spaced bins. (A) Distance potential between N and Cα in the same alanine residue, i.e. a covalent bond. (B) Bond angle potential for the N-Cα-C angle in glutamic acid. (C) Distance potential between Cα and the sidechain centroid on the same residue in isoleucine. (D) Torsion angle potentials in valine. There are different potentials for residues predicted as α-helical, β-sheet and coiled. The true ranges of α-helices and β-sheets are shown. (E) Distance potentials between sidechains for two pairs of amino acids. A PMF calculated from the PDB is also shown for comparison.

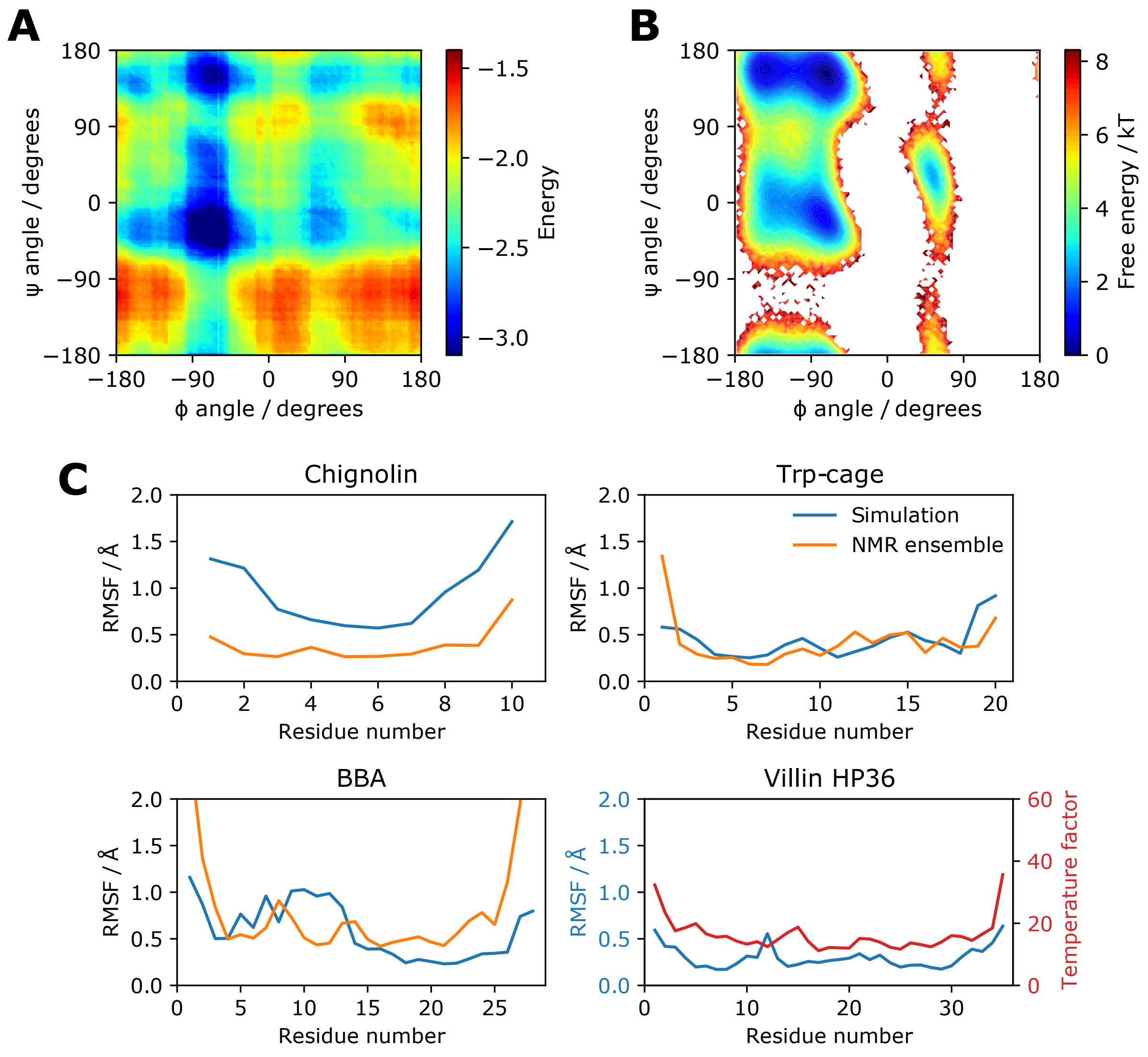


Figure 3 Protein energy and dynamics. (A) The energy of alanine dipeptide in the learned potential for different φ and ψ torsion angles. Conformations are generated with PeptideBuilder [49] at intervals of 2° and scored. (B) Free energy for different φ and ψ torsion angles for all-atom alanine dipeptide. The energy is obtained from direct histogram estimation from all-atom simulations. The data is taken from Wang et al. 2019 [15] and provided by the authors. (C) Cα atom RMSF values for simulations of 6m steps using the learned potential starting from the native structure. An initial burn-in period of 6m steps was discarded. The RMSF values from the NMR ensembles are also shown, or the crystal temperature factors in the case of villin HP36.

This is shown in Figure 4A. We do find that longer sequences are able to form the correct secondary structure, and often the correct tertiary structure given enough simulation time. However to better explore tertiary structure formation with available compute resources we started the proteins from extended conformations containing predicted secondary structure, i.e. α-helical φ/ψ angles for predicted α-helical residues and extended φ/ψ angles for residues predicted β-sheet or coiled. The Cα RMSD distributions of the trajectories after a burn-in period, the starting Cα RMSDs and representative models from the trajectories are shown in Figure 4B. All the proteins fold to a native-like structure over 12m steps, taking about 36 hours on a single graphics processing unit (GPU) or about 3 times as long on the central processing unit (CPU). Chignolin adopts approximately the correct turn structure; the minimum sampled Cα RMSD of 2.15 Å is comparable to the Cα RMSD of 1.82 Å between the first model in the NMR ensemble and the crystal structure. Trp-cage lacks native helix formation in the middle of the protein but the overall shape is correct. BBA forms a native-like structure with a minimum Cα RMSD during the simulation of 3.47 Å. For villin HP36 the N-terminal helix faces the wrong direction but the location of the turns and the rest of the structure is correct.

Models are also shown from web servers implementing two popular methods that carry out coarse-grained protein folding: UNRES [50, 38], which has two interacting sites per protein, and CABS-fold [51, 39], which uses a lattice model. Both make use of predicted secondary structure, and are given the same single sequence prediction used here. Performance of the learned potential is comparable to these established methods across the proteins tested, with the learned potential able to break secondary structure elements and add turns in the correct locations. The comparison to these methods is not exact; they provide less compute resources on their servers than the 36 hours of GPU time per protein used here, whereas this method does not employ the replica exchange algorithms used to enhance sampling in UNRES and CABS-fold. Enhanced sampling approaches to predict the structures of larger proteins with the learned potential is a topic of future work. We note that larger proteins remain close to their native structure with low energy when simulated, suggesting that the native structure is in an energy minimum that can be accessed with appropriate sampling. Another point of note is that the same simulation parameters are used when simulating all four proteins (temperature 0.015, coupling constant 25). More accurate models can likely be obtained by optimising these for each protein but we did not want to risk overfitting. The $(AAQAA)_3$ repeat peptide helix formation simulations were, however, carried out at a higher temperature (temperature 0.022, coupling constant 100) to faster explore the conformational space needed to form the helix. We do notice some success in folding from an extended chain with these higher temperature parameters. For example, villin HP36 reaches a minimum Cα RMSD of 4.19 Å over 12m steps, with the orientation of all the helices correct.

Protein design and model scoring

In order to see whether native sequences are optimal for native structures in the learned potential, we thread sequences with varying fractions of native amino acids onto the native backbone and calculate the energy. Non-native residues are drawn randomly from the distribution

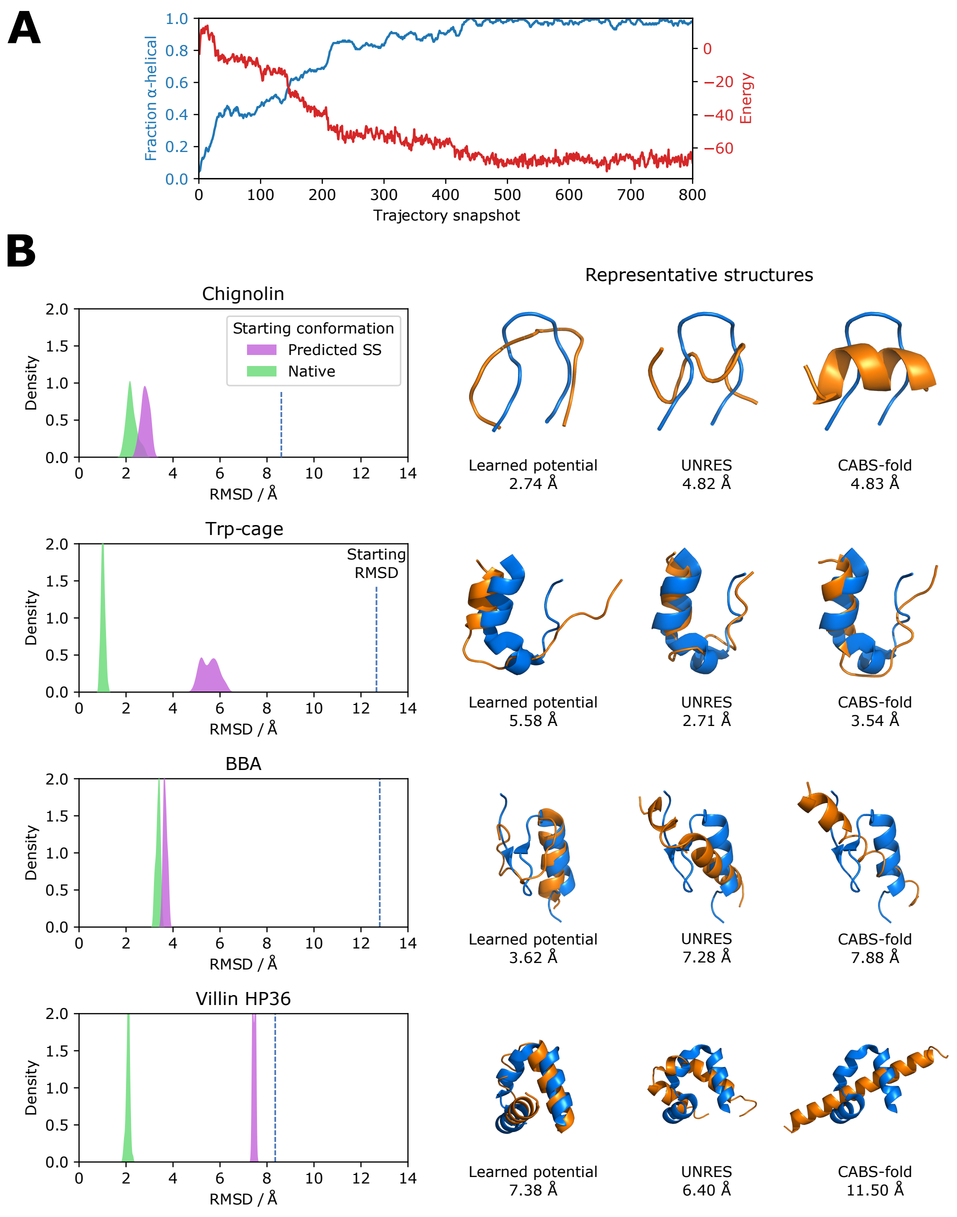


Figure 4 Folding small proteins and peptides. (A) The $(AAQAA)_3$ repeat peptide folds from a random starting conformation into an α-helix over 12m steps. The α-helical fraction and energy in the learned potential are shown with a snapshot taken every 15,000 steps. (B) Cα RMSD distributions from simulations of 3m steps for 4 proteins starting from predicted secondary structure and native conformations. An initial burn-in period of 9m steps was discarded. The starting Cα RMSD for the predicted secondary structure conformation is also shown. A representative structure found with MDAnalysis [52] is shown (orange) along with models generated from the web servers of UNRES [50] and CABS-fold [51]. The Cα RMSD to the native structure (blue) is given for each model.

of amino acids in the PDB. As shown in Figure 5A we find that an increased fraction of native amino acids gives a lower energy, with the native sequence lower in energy than most 90% native sequences and considerably lower in energy than less native sequences. For chignolin we are able to recover a native-like sequence by starting from a random sequence and making mutations. At each trial a residue is mutated and the mutation is accepted or rejected based on the energy change when threaded onto the native structure and the distance through the trial process (see the methods). After 1,000 trials the sequence obtained is FYDPSTGVSK, which has 5 of 10 residues identical to the native sequence of YYDPETGTWY.

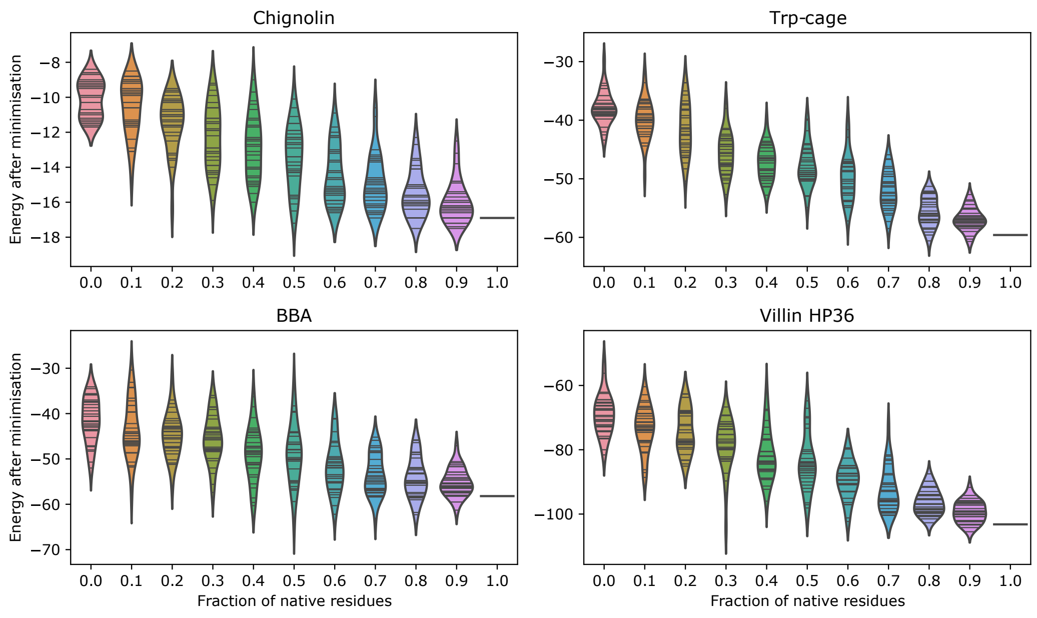
We also investigate whether the potential is able to distinguish between native structures and close decoys. Models up to 12 Å from the native structure were obtained using 3DRobot [53], which generates diverse and well-packed decoys using fragment assembly and energy minimisation. As shown in Figure 5B, the native structure has low energy compared to the decoys for Trp-cage and villin HP36. Chignolin was too small to run 3DRobot on. For BBA many decoys are lower in energy than the native structure. A number of these lower energy decoys are topological mirrors in which the β-turn faces the other way, but the structure of the protein is otherwise native-like. The problem of mirror topologies has been discussed previously for protein structure determination [54] and de novo protein structure prediction [55].

Discussion

The purpose of learning a coarse-grained force field for proteins is to demonstrate that DMS can be used to learn all the parameters from scratch in simple, interpretable force fields. It is notable that running training simulations from the native structure, reaching up to 4 Å RMSD to the native structure over 2,000 steps during training, is sufficient to learn a potential that can fold proteins from an extended chain of secondary structure elements over a few million steps. The same potential can also be used for model scoring, despite the energy not being explicitly used at all beyond force calculation during training. This particular force field may be useful for exploring the conformational space of proteins, discovering folding pathways, assessing flexible or disordered regions of proteins, or predicting structure in combination with co-evolutionary or experimental constraints [56].

Here we have used RMSD as a simple loss function, but there are a variety of possible loss functions for DMS suitable for other systems. Examples include RMSD over the course of a simulation, the radial distribution function, the flexibility of a set of atoms during the simulation, the distance between two molecules, supramolecular geometry (e.g. assembly of molecules into fibres), the correlation of different particle velocities, the energy of a system, the temperature of a system, a measure of phase change, steered molecular dynamics, or some combination of the above. Many of the possible properties are based on static reference structures, thermodynamic observables or chemical knowledge, meaning that expensive trajectory data is not necessarily required. Complex constraints that might be difficult to use in the simulation itself can be targeted via the loss function. Any property that can be computed from the system with a meaningful gradient can be used to guide a learned force field to reproduce

A



B

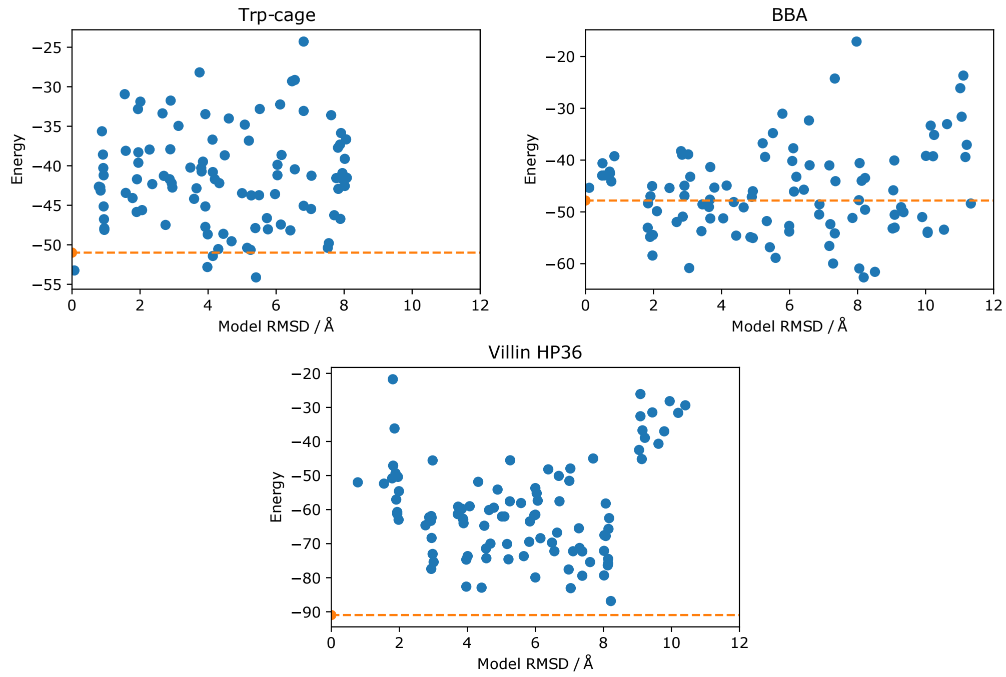


Figure 5 Protein design and model scoring. (A) The energy of sequences with varying fractions of native residues threaded onto the native structure. 30 sequences are scored for each fraction. A short minimisation of 100 steps is carried out for each sequence and the energy is recorded at the end of the minimisation. (B) The energy of well-packed decoys generated using 3DRobot [53] is compared to the Cα RMSD of the decoys (blue dots) and the energy of the native structure (orange line). Chignolin was too small to run 3DRobot on.

desired behaviour. This sets DMS apart from contrastive divergence approaches that train entire force fields by teaching them to distinguish native and non-native ensembles. Possible applications include combining DMS with learned atom typing [24] to study protein-ligand binding, improving torsion angle potentials to balance bonded and non-bonded terms [57], and exploring whether multi-body terms can make molecular mechanics potentials more accurate. The development of software packages appropriate for DMS such as JAX MD [58], TorchMD [18], DeePMD-kit [59], SchNetPack [60], and DiffTaichi [61], along with the effort to make programming languages such as Julia differentiable by default [62], will assist in the development of DMS.

Other coarse-grained systems may be immediately amenable to DMS, for example protein aggregation [63]. For DMS to be used to parameterise all-atom molecular mechanics force fields - either from scratch, by tuning existing parameters, or by adding new atom types - a few issues need to be addressed. Algorithms such as particle mesh Ewald for long-range electrostatic interactions [64] will have to be implemented in differentiable frameworks. The best form for temperature control during training will have to be considered, as stochasticity will likely affect the gradients. In this study, the thermostat was not used during training. Even the form of the numerical integration will have to be explored, as it is unclear that velocity Verlet integration with a standard time step is best-suited to DMS. It has been shown that discrete time steps can lead to incorrect gradients [61]. Ensuring continuous potentials should prevent this, and the standard all-atom potentials - harmonic, cosine, Coulomb and Lennard-Jones - are all continuous. However, cutoffs for short-range forces and neighbour lists will have to be carefully considered. Finding the parameters for a simple water model would be an ideal system to start with [18].

DMS, like deep learning, appears to be quite sensitive to hyperparameters, and generally appropriate choices for these will have to be discovered. Learned potentials being amenable to physical interpretation helps when investigating such issues, as well as alleviating problems of under-specification and shortcut learning identified for neural networks [65]. For example, increasing the learning rate in this study gives jagged potentials, whereas decreasing it leads to prohibitively slow training.

As with analogous developments in deep learning, a limitation of DMS is the GPU memory required. Most deep learning software frameworks are geared towards reverse-mode AD, in which intermediate results of the forward pass of the network are stored and used during the reverse pass to calculate gradients. This approach is efficient when the number of trainable parameters is large, as is usually the case with neural networks. However, the requirement to store intermediate results means that it scales linearly in memory with the number of steps for DMS. By contrast, forward-mode AD does not store intermediate results because the gradients are calculated in tandem with the forward pass. The memory required does not therefore increase with the number of steps, though the number of learned parameters does affect the computation speed. Use of forward-mode AD may provide a way for DMS to use the large number of steps during training that would be required to learn molecular mechanics force fields. Advances in hardware, GPU parallelism and algorithmic techniques such as gradient checkpointing, invertible simulations [66] and offloading compute to the CPU [67] also present solutions to the issue of GPU memory.

Though applications of deep learning in biology have been impressive, the models used have been largely taken off the shelf from other fields. DMS provides a general approach to improving and expanding the force fields that have been crucial for biological understanding. It uses the techniques and frameworks of neural network training but relies on proven, interpretable functional forms rather than deep neural networks themselves.

Methods

Dataset

The PDB [47] was searched for protein chains with 20 to 100 residues, no internal missing residues and resolution 2.5 Å or better. These chains were clustered at 30% sequence identity to reduce redundancy. Proteins homologous to those used in the results were removed from the dataset by eliminating overlap at the same ECOD T-level [68] and removing hits when searching the PDB using BLAST with an E-value of 1.0. The resulting chains were randomly split into a training set of 2,004 chains and a validation set of 200 chains used to monitor training. Single sequence secondary structure prediction was carried out using PSIPRED [69]. Details of the proteins used in the results are given in Table S1. Dataset collection and other aspects of the work were carried out using BioStructures.jl [70] and the Bio.PDB module of Biopython [71].

Molecular simulation

The system was implemented in PyTorch [44]. A protein is represented in a coarse-grained manner with 4 point particles for each residue corresponding to backbone nitrogen, backbone Cα, backbone carbonyl carbon and the centroid of the sidechain heavy atoms. There is no explicit solvent, no periodic boundary conditions and no neighbour list. The masses are set to 15 for N (includes amide H), 13 for Cα (includes H) and 28 for C (includes carbonyl O). The sidechain mass for each amino acid is the sum of the atom masses in the all-atom sidechain, with glycine set artificially heavier at a mass of 10.

The overall learned potential used to obtain the forces at each time step consists of 3 components, each consisting of many individual potentials. Each individual potential consists of a number of energy values corresponding to 140 bins evenly distributed over a specified distance or angle range. Overall there are 29,360 individual potentials and 4,111,000 learnable parameters.

1. *Pairwise distance potentials*. There are 80 atom types (4 atoms for 20 amino acids) and each pair has a distance potential. In addition there are separate potentials for each atom pair on residues close in sequence with residue separations $i \rightarrow i$ to $i \rightarrow i+4$, allowing the model to learn local constraints separately from global preferences. Covalent bond interactions are included implicitly in the potentials for atom pairs on the same residue,

and the $i \rightarrow i+1$ potentials in the case of the C-N backbone covalent bond. There are 28,960 pairwise distance potentials in total (3,240 general, 120 same residue, 6,400 $\times$ 4 close residues). Each distance potential has 140 bins distributed between 1 Å and 15 Å (0.1 Å width per bin). For the close residue potentials the bins are distributed between 0.7 Å and 14.7 Å (0.1 Å width per bin), and for same residue pairs between 0.7 Å and 5.6 Å (0.035 Å width per bin).

2. *Bond angle potentials*. There are 5 bond angles in the model - 3 in the backbone, 2 to the sidechain centroid - and there is a separate potential for each of these for each amino acid. There are 100 bond angle potentials in total (5×20). Each angle potential has 140 bins distributed between 60° and 180° (0.86° width per bin).

3. *Torsion angle potentials*. There are 5 torsion angles in the model - 3 in the backbone, 2 to the sidechain centroid - and there is a separate potential for each of these for each amino acid. There are also separate potentials for residues predicted as α-helical, β-sheet or coiled to help in secondary structure formation. There are 300 torsion angle potentials in total ($5 \times 20 \times 3$). Each torsion angle potential has 140 bins distributed between -180° and 180° (2.57° width per bin) with an extra bin on either end to allow the derived force to be periodic.

At each simulation time step the force is calculated as the negative gradient of the potential using the following finite differencing procedure for each individual potential:

1. Calculate the current value of the property, e.g. the distance between two atoms or a bond angle.

2. Find the bin b_i with the closest bin centre to the value, excluding the first and last bins. For distances this means that all distances over the maximum bin distance (e.g. 15 Å) are treated as being in the penultimate bin.

3. Calculate $F = \frac{1}{2}(E(b_{i-1}) - E(b_{i+1}))$, where $E(b_i)$ is the energy of bin b_i.

4. Multiply F by the appropriate vector on each atom to apply the force [72].

This is shown in Figure 1C. The sum of the resulting forces on each atom is divided by the atomic masses to get the accelerations. This approach is differentiable since it can be implemented in a vectorised manner in PyTorch. Whilst more sophisticated methods such as a sum of Gaussians or spline fitting could be adopted to obtain the force from the potential, this finite differencing procedure was found to be memory-efficient and effective. It was also found to be more effective than learning force values directly, an approach that does not immediately provide an interpretable potential that can be used to calculate energies.

The coordinates and velocities are updated at every time step using the velocity Verlet integrator. If $\mathbf{x}(t)$ is the coordinates at time t, $\mathbf{v}(t)$ is the velocities at time t, $\mathbf{a}(t)$ is the accelerations at time t, and Δt is the time step, then the procedure is:

1. Calculate $\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \mathbf{v}(t)\Delta t + \frac{1}{2}\mathbf{a}(t)\Delta t^2$.

2. Obtain $\mathbf{a}(t+\Delta t)$ from the learned potential as described above using $\mathbf{x}(t+\Delta t)$.
3. Calculate $\mathbf{v}(t+\Delta t) = \mathbf{v}(t) + \frac{1}{2}(\mathbf{a}(t) + \mathbf{a}(t+\Delta t))\Delta t$.
4. Update $t+\Delta t$ to t and go to step 1.

No thermostat was used during training as it was not found to improve performance. Training therefore takes place in the NVE ensemble. A time step of 0.02 was used for training.

During production runs used to obtain results, which consisted of many more steps than during training, the Andersen thermostat was used to keep temperature constant [73]. Production runs therefore take place in the NVT ensemble. At each step, each atom is given a new velocity with probability equal to the time step divided by a coupling constant. The new velocities are drawn from a normal distribution with mean 0 and standard deviation equal to a temperature value. Simulations were run with temperature 0.015 and coupling constant 25, apart from the $(AAQAA)_3$ repeat peptide helix formation simulations which were run with temperature 0.022 and coupling constant 100. A lower time step of 0.004 was used during production runs to ensure stability of the simulations. We did try using Langevin dynamics for production runs but found it did not improve the results.

Training

During training, simulations are started from the native structure. Starting velocities are drawn from a normal distribution with mean 0 and standard deviation 0.1. At the end of each simulation R_f is calculated as the RMSD across all atoms in the coarse-grained model of the final conformation compared to the native structure using the Kabsch algorithm. $\log(1+R_f)$ was used as the loss function, which was found to give better performance than using R_f. Each parameter in the learned potential then has an associated gradient from AD in PyTorch, which the optimiser uses to modify the potential to reduce the R_f from future simulations. At the start of training all values in the potential are set to zero. We did try starting the potentials from PMFs derived from the PDB but this did not improve results. The Adam optimiser [74] was used with a learning rate of 10^{-4}. Gradients are accumulated for 100 proteins before the optimiser updates the gradients. At epoch 38 the ADAM optimiser was reset with a lower learning rate of 5×10^{-5}.

One epoch of training consists of simulating each protein in the training set in a random order. A batch size of 1 was used due to memory constraints. The number of steps in the simulation was increased over the course of training, starting at 250 for the first epoch and increasing by 250 every 5 epochs to a maximum of 2,000 at epoch 36 and beyond. This allowed the model to access lower R_f values during early epochs to learn basic chemical principles such as steric clashing and covalent bonding. Further increasing the step number was prevented by memory limitations of the GPU (32 GB); during training the memory required scales linearly with the number of steps due to the requirements of storing intermediate computations for reverse mode AD. This limitation is not present for production runs, which use memory constant in the number of steps and typically less than 1 GB. Training was carried out on a NVIDIA Tesla V100 for 45 epochs, which took around 2 months. This can likely be sped up using multiple GPUs. The training and validation R_f values throughout training are shown in Figure S1. The protein folding simulations shown in Figure 4B took around 36 hours for 12m steps, equating to around 10 ms per time step. This was constant across protein sizes tested due to the vectorised operations used. Simulation time is around 3 times slower on the CPU depending on the hardware used.

Analysis

Throughout this study we analyse only coarse-grained models, however all-atom models can be generated using software such as PULCHRA [75] if required. Alanine dipeptide conformations were generated using PeptideBuilder [49] at intervals of 2°. The $(AAQAA)_3$ repeat peptide simulation was started from a random conformation where each residue was given a φ angle between -180° and -30° and a ψ angle between -180° and 180°. α-helical fraction was determined by the fraction of non-terminal residues where the φ angle was between -120° and -30° and the ψ angle was between -60° and 30°. The α-helical fraction was averaged over a window of 5 snapshots either side of the snapshot in question for ease of visualisation. For simulations starting from predicted secondary structure, a residue was given an α-helical starting conformation (φ -60°, ψ -60°) if predicted α-helical and an extended starting conformation (φ -120°, ψ 140°) if predicted β-sheet or coiled. The representative structures in Figure 4 were found using MDAnalysis [52] and were visualised in PyMOL [76] after being run through PULCHRA. UNRES models were generated on the web server [50] using the parameters for the MREMD structure prediction example in the tutorial. The number of steps was increased to the maximum of 10^7. CABS-fold models were generated on the web server [51] with default de novo parameters. In both cases we use the same single sequence secondary structure prediction as for our method.

The fraction of native sequence results involved threading a sequence onto the native structure. Sequences were chosen by mutating a given fraction of residues to amino acids taken from the distribution of amino acids in the PDB. 30 sequences are generated for each fraction. Each sidechain centroid is placed at a distance from the Cα atom corresponding to the minimum in the learned potential for that amino acid, along the vector linking the Cα atom and the native sidechain centroid. A brief energy minimisation of 100 steps in the learned potential is carried out and the final energy is used as the energy of the sequence. For the chignolin design task 1,000 trial mutations were made. Each trial involved mutating one residue to an amino acid taken from the PDB distribution, calculating the new energy by threading the sequence and running energy minimisation as above, and accepting or rejecting the mutation. Mutations resulting in lower energy are always accepted, and mutations resulting in a higher energy of 10 or more energy units are always rejected. Mutations resulting in an energy increase up to 10 energy units are accepted with a probability that changes linearly from 0.25 at the start of the simulation to 0.0 at 500 trials, and remains at 0.0 for the remaining trials.

3DRobot models were generated using the 3DRobot standalone software with default parameters [53]. Plots throughout were produced with Matplotlib [77], seaborn [78] and PyEMMA [79] for Figure 3B.

Data availability

The learned potential, simulation scripts and training code are made available under a permissive license at `https://github.com/psipred/cgdms`.

Acknowledgements

We thank Nick Charron and Cecilia Clementi for providing the all-atom alanine dipeptide data from Wang et al. 2019 [15]. We thank the UCL Bioinformatics Group for useful discussions. This work was supported by the European Research Council Advanced Grant "ProCovar" (project ID 695558).

References

[1] S A Hollingsworth and R O Dror. Molecular Dynamics Simulation for All. *Neuron*, 99(6): 1129–1143, 2018.

[2] E Brini, C Simmerling, and K Dill. Protein storytelling through physics. *Science*, 370:eaaz3041, 2020.

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

[4] S Kmiecik, D Gront, M Kolinski, L Wieteska, A E Dawid, and A Kolinski. Coarse-Grained Protein Models and Their Applications. *Chem Rev*, 116(14):7898–7936, 2016.

[5] S Piana, K Lindorff-Larsen, and D E Shaw. How robust are protein folding simulations with respect to force field parameterization? *Biophys J*, 100(9):L47–L49, 2011.

[6] L P Wang, T J Martinez, and V S Pande. Building Force Fields: An Automatic, Systematic, and Reproducible Approach. *J Phys Chem Lett*, 5(11):1885–1891, 2014.

[7] R B Best, W Zheng, and J Mittal. Balanced Protein-Water Interactions Improve Properties of Disordered Proteins and Non-Specific Protein Association. *J Chem Theory Comput*, 10(11): 5113–5124, 2014.

[8] P Robustelli, S Piana, and D E Shaw. Developing a molecular dynamics force field for both folded and disordered protein states. *Proc Natl Acad Sci USA*, 115(21):E4758–E4766, 2018.

[9] M Liu, A K Das, J Lincoff, S Sasmal, S Y Cheng, R Vernon, J Forman-Kay, and T Head-Gordon. Configurational Entropy of Folded Proteins and its Importance for Intrinsically Disordered Proteins. *arXiv*, 2007.06150, 2020.

[10] A W Senior, R Evans, J Jumper, J Kirkpatrick, L Sifre, T Green, C Qin, A Žídek, A W R Nelson, A Bridgland, H Penedones, S Petersen, K Simonyan, S Crossan, P Kohli, D T Jones, D Silver, K Kavukcuoglu, and D Hassabis. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, 2020.

[11] F Noé, G De Fabritiis, and C Clementi. Machine learning for protein folding and dynamics. *Curr Opin Struct Biol*, 60:77–84, 2020.

[12] F Noé, A Tkatchenko, K R Müller, and C Clementi. Machine Learning for Molecular Simulation. *Annu Rev Phys Chem*, 71:361–390, 2020.

[13] P Gkeka, G Stoltz, A Barati Farimani, Z Belkacemi, M Ceriotti, J D Chodera, A R Dinner, A L Ferguson, J B Maillet, H Minoux, C Peter, F Pietrucci, A Silveira, A Tkatchenko, Z Trstanova, R Wiewiora, and T Lelièvre. Machine Learning Force Fields and Coarse-Grained Variables in Molecular Dynamics: Application to Materials and Biological Systems. *J Chem Theory Comput*, 16(8):4757–4775, 2020.

[14] L Zhang, J Han, H Wang, R Car, and W E. DeePCG: Constructing coarse-grained models via deep neural networks. *J Chem Phys*, 149(3):034101, 2018.

[15] J Wang, S Olsson, C Wehmeyer, A Pérez, N E Charron, G De Fabritiis, F Noé, and C Clementi. Machine Learning of Coarse-Grained Molecular Dynamics Force Fields. *ACS Cent Sci*, 5(5): 755–767, 2019.

[16] B E Husic, N E Charron, D Lemm, J Wang, A Pérez, M Majewski, A Krämer, Y Chen, S Olsson, G De Fabritiis, F Noé, and C Clementi. Coarse graining molecular dynamics with graph neural networks. *J Chem Phys*, 153(19):194101, 2020.

[17] W Wang and R Gómez-Bombarelli. Coarse-graining auto-encoders for molecular dynamics. *npj Computational Materials*, 5(125), 2019.

[18] S Doerr, M Majewsk, A Pérez, A Krämer, C Clementi, F Noé, T Giorgino, and G De Fabritiis. TorchMD: A deep learning framework for molecular simulations. *arXiv*, 2012.12106, 2020.

[19] S Chmiela, A Tkatchenko, H E Sauceda, I Poltavsky, K T Schütt, and K R Müller. Machine learning of accurate energy-conserving molecular force fields. *Sci Adv*, 3(5):e1603015, 2017.

[20] M Bogojeski, L Vogt-Maranto, M E Tuckerman, K R Müller, and K Burke. Quantum chemical accuracy from density functional approximations via machine learning. *Nat Commun*, 11:5223, 2020.

[21] J Hermann, Z Schätzle, and F Noé. Deep-neural-network solution of the electronic Schrödinger equation. *Nat Chem*, 12(10):891–897, 2020.

[22] S Batzner, T E Smidt, L Sun, J P Mailoa, M Kornbluth, N Molinari, and B Kozinsky. SE(3)-Equivariant Graph Neural Networks for Data-Efficient and Accurate Interatomic Potentials. *arXiv*, 2101.03164, 2021.

[23] F Noé, S Olsson, J Köhler, and H Wu. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365:eaaw1147, 2019.

[24] Y Wang, J Fass, and J D Chodera. End-to-End Differentiable Molecular Mechanics Force Field Construction. *arXiv*, 2010.01196, 2020.

[25] M AlQuraishi. End-to-End Differentiable Learning of Protein Structure. *Cell Systems*, 8(4): 292–301.e3, 2019.

[26] M Ragoza, J Hochuli, E Idrobo, J Sunseri, and D R Koes. Protein-Ligand Scoring with Convolutional Neural Networks. *J Chem Inf Model*, 57(4):942–957, 2017.

[27] D A Rufa, H E Bruce Macdonald, J Fass, M Wieder, P B Grinaway, A E Roitberg, O Isayev, and J D Chodera. Towards chemical accuracy for alchemical free energy calculations with hybrid physics-based machine learning / molecular mechanics potentials. *bioRxiv*, `https://www.biorxiv.org/content/10.1101/2020.07.29.227959v1`, 2020.

[28] W Wang, S Axelrod, and R Gómez-Bombarelli. Differentiable Molecular Simulations for Control and Learning. *arXiv*, 2003.00868, 2020.

[29] C Rackauckas, Y Ma, J Martensen, C Warner, K Zubov, R Supekar, D Skinner, A Ramadhan, and A Edelman. Universal Differential Equations for Scientific Machine Learning. *arXiv*, 2001.04385, 2020.

[30] P Holl, V Koltun, and N Thuerey. Learning to Control PDEs with Differentiable Physics. *arXiv*, 2001.07457, 2020.

[31] J Ingraham, A Riesselman, C Sander, and D Marks. Learning Protein Structure with a Differentiable Simulator. *ICLR*, 2019.

[32] J M Jumper, N F Faruk, K F Freed, and T R Sosnick. Trajectory-based training enables protein simulations with accurate folding and Boltzmann ensembles in cpu-hours. *PLoS Comput Biol*, 14(12):e1006578, 2018.

[33] C Várnai, N S Burkoff, and D L Wild. Efficient Parameter Estimation of Generalizable Coarse-Grained Protein Force Fields Using Contrastive Divergence: A Maximum Likelihood Approach. *J Chem Theory Comput*, 9(12):5718–5733, 2013.

[34] A A Podtelezhnikov, Z Ghahramani, and D L Wild. Learning about protein hydrogen bonding by minimizing contrastive divergence. *Proteins*, 66(3):588–599, 2007.

[35] B Zaborowski, D Jagieła, C Czaplewski, A Hałabis, A Lewandowska, W Żmudzińska, S Ołdziej, A Karczyńska, C Omieczynski, T Wirecki, and A Liwo. A Maximum-Likelihood Approach to Force-Field Calibration. *J Chem Inf Model*, 55(9):2050–2070, 2015.

[36] P Krupa, A Hałabis, W Żmudzińska, S Ołdziej, H A Scheraga, and A Liwo. Maximum Likelihood Calibration of the UNRES Force Field for Simulation of Protein Structure and Dynamics. *J Chem Inf Model*, 57(9):2364–2377, 2017.

[37] A G Baydin, B A Pearlmutter, A A Radul, and J M Siskind. Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research*, 18(1):1–43, 2018.

[38] A Liwo, M Khalili, and H A Scheraga. Ab initio simulations of protein-folding pathways by molecular dynamics with the united-residue model of polypeptide chains. *Proc Natl Acad Sci USA*, 102(7):2362–2367, 2005.

[39] A Kolinski. Protein modeling and structure prediction with a reduced representation. *Acta Biochim Pol*, 51(2):349–371, 2004.

[40] J Maupetit, P Tuffery, and P Derreumaux. A coarse-grained protein force field for folding and structure prediction. *Proteins*, 69(2):394–408, 2007.

[41] H Zhou and Y Zhou. Distance-scaled, finite ideal-gas reference state improves structure-derived

potentials of mean force for structure selection and stability prediction. *Protein Sci*, 11(11): 2714–2726, 2002.

[42] M Y Shen and A Sali. Statistical potential for assessment and prediction of protein structures. *Protein Sci*, 15(11):2507–2524, 2006.

[43] M Innes, A Edelman, K Fischer, C Rackauckas, E Saba, V B Shah, and W Tebbutt. A Differentiable Programming System to Bridge Machine Learning and Scientific Computing. *arXiv*, 1907.07587, 2019.

[44] A Paszke, S Gross, F Massa, A Lerer, J Bradbury, G Chanan, T Killeen, Z Lin, N Gimelshein, L Antiga, A Desmaison, A Kopf, E Yang, Z DeVito, M Raison, A Tejani, S Chilamkurthy, B Steiner, L Fang, J Bai, and S Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems 32*, pages 8024–8035, 2019.

[45] K Lindorff-Larsen, S Piana, R O Dror, and D E Shaw. How fast-folding proteins fold. *Science*, 334(6055):517–520, 2011.

[46] H Nguyen, J Maier, H Huang, V Perrone, and C Simmerling. Folding simulations for proteins with diverse topologies are accessible in days with a physics-based force field and implicit solvent. *J Am Chem Soc*, 136(40):13959–13962, 2014.

[47] H M Berman, J Westbrook, Z Feng, G Gilliland, T N Bhat, H Weissig, I N Shindyalov, and P E Bourne. The Protein Data Bank. *Nucleic Acids Res*, 28(1):235–242, 2000.

[48] S Honda, T Akiba, Y S Kato, Y Sawada, M Sekijima, M Ishimura, A Ooishi, H Watanabe, T Odahara, and K Harata. Crystal structure of a ten-amino acid protein. *J Am Chem Soc*, 130 (46):15327–15331, 2008.

[49] M Z Tien, D K Sydykova, A G Meyer, and C O Wilke. PeptideBuilder: A simple Python library to generate model peptides. *PeerJ*, 1:e80, 2013.

[50] C Czaplewski, A Karczynska, A K Sieradzan, and A Liwo. UNRES server for physics-based coarse-grained simulations and prediction of protein structure, dynamics and thermodynamics. *Nucleic Acids Res*, 46(W1):W304–W309, 2018.

[51] M Blaszczyk, M Jamroz, S Kmiecik, and A Kolinski. CABS-fold: Server for the de novo and consensus-based prediction of protein structure. *Nucleic Acids Res*, 41(W1):W406–W411, 2013.

[52] R J Gowers, M Linke, J Barnoud, T J E Reddy, M N Melo, S L Seyler, J Domański, D L Dotson, S Buchoux, I M Kenney, and O Beckstein. MDAnalysis: A Python Package for the Rapid Analysis of Molecular Dynamics Simulations. *Proceedings of the 15th Python in Science Conference*, pages 98–105, 2016.

[53] H Deng, Y Jia, and Y Zhang. 3DRobot: automated generation of diverse and well-packed protein structure decoys. *Bioinformatics*, 32(3):378–387, 2016.

[54] A Pastore, R A Atkinson, V Saudek, and R J Williams. Topological mirror images in protein structure computation: an underestimated problem. *Proteins*, 10(1):22–32, 1991.

[55] J G Greener, S M Kandathil, and D T Jones. Deep learning extends de novo protein modelling coverage of genomes using iteratively predicted structural constraints. *Nat Commun*, 10(1):3977, 2019.

[56] N J Cheung and W Yu. De novo protein structure prediction using ultra-fast molecular dynamics simulation. *PLoS One*, 13(11):e0205819, 2018.

[57] P S Nerenberg and T Head-Gordon. New developments in force fields for biomolecular simulations. *Curr Opin Struct Biol*, 49:129–138, 2018.

[58] S S Schoenholz and E D Cubuk. JAX, M.D.: A Framework for Differentiable Physics. *arXiv*, 1912.04232, 2019.

[59] H Wang, L Zhang, J Han, and W E. DeePMD-kit: A deep learning package for many-body potential energy representation and molecular dynamics. *Computer Physics Communications*, 228:178–184, 2018.

[60] K T Schütt, P Kessel, M Gastegger, K A Nicoli, A Tkatchenko, and K R Müller. SchNetPack: A Deep Learning Toolbox For Atomistic Systems. *J Chem Theory Comput*, 15(1):448–455, 2019.

[61] Y Hu, L Anderson, T-M Li, Q Sun, N Carr, J Ragan-Kelley, and F Durand. DiffTaichi: Differentiable Programming for Physical Simulation. *arXiv*, 1910.00935, 2019.

[62] M Innes. Don't Unroll Adjoint: Differentiating SSA-Form Programs. *arXiv*, 1810.07951, 2018.

[63] A Šarić, Y C Chebaro, T P Knowles, and D Frenkel. Crucial role of nonspecific interactions in amyloid nucleation. *Proc Natl Acad Sci USA*, 111(50):17869–17874, 2014.

[64] T Darden, D York, and L Pedersen. Particle mesh Ewald: An $N \cdot \log(N)$ method for Ewald sums in large systems. *J Chem Phys*, 98(12):10089, 1993.

[65] R Geirhos, J H Jacobsen, C Michaelis, R Zemel, W Brendel, M Bethge, and F A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2:665–673, 2020.

[66] L Ardizzone, J Kruse, S Wirkert, D Rahner, E W Pellegrini, R S Klessen, L Maier-Hein, C Rother, and U Köthe. Analyzing inverse problems with invertible neural networks. *ICLR*, 2019.

[67] J Ren, S Rajbhandari, R Y Aminabadi, O Ruwase, S Yang, M Zhang, D Li, and Y He. ZeRO-Offload: Democratizing Billion-Scale Model Training. *arXiv*, 2101.06840, 2021.

[68] H Cheng, R D Schaeffer, Y Liao, L N Kinch, J Pei, S Shi, B H Kim, and N V Grishin. ECOD: an evolutionary classification of protein domains. *PLoS Comput Biol*, 10(12):e1003926, 2014.

[69] D T Jones. Protein secondary structure prediction based on position-specific scoring matrices. *J Mol Biol*, 292(2):195–202, 1999.

[70] J G Greener, J Selvaraj, and B J Ward. BioStructures.jl: read, write and manipulate macromolecular structures in Julia. *Bioinformatics*, 36(14):4206–4207, 2020.

[71] T Hamelryck and B Manderick. PDB file parser and structure class implemented in Python. *Bioinformatics*, 19(17):2308–2310, 2003.

[72] B Monasse and F Boussinot. Determination of Forces from a Potential in Molecular Dynamics. *arXiv*, 1401.1181, 2014.

[73] H C Andersen. Molecular dynamics simulations at constant pressure and/or temperature. *The Journal of Chemical Physics*, 72(4):2384–2393, 1980.

[74] D P Kingma and J L Ba. Adam: A Method for Stochastic Optimization. *ICLR*, 2015.

[75] P Rotkiewicz and J Skolnick. Fast procedure for reconstruction of full-atom protein models from reduced representations. *J Comput Chem*, 29(9):1460–1465, 2008.

[76] Schrödinger, LLC. The PyMOL Molecular Graphics System. Version 2.5, 2020.

[77] J D Hunter. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9 (3):90–95, 2007.

[78] M Waskom and the seaborn development team. mwaskom/seaborn. *Zenodo*, `https://doi.org/10.5281/zenodo.592845`, 2020.

[79] M K Scherer, B Trendelkamp-Schroer, F Paul, G Pérez-Hernández, M Hoffmann, N Plattner, C Wehmeyer, J. H Prinz, and F Noé. PyEMMA 2: A Software Package for Estimation, Validation, and Analysis of Markov Models. *J Chem Theory Comput*, 11(11):5525–5542, 2015.

Supplementary Data

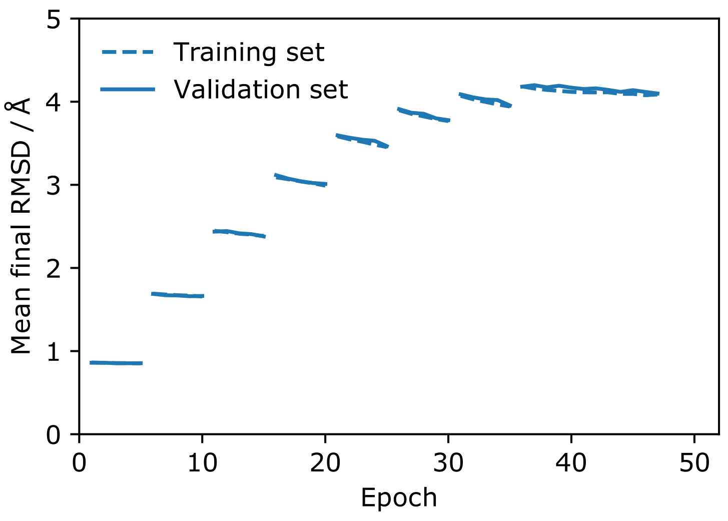


Figure S1 Training progress. The mean final RMSD across all atoms in the coarse-grained model for each epoch is shown for the training set and the validation set. As outlined in the methods the number of steps simulated starts at 250 and increases by 250 every 5 epochs, reaching a maximum of 2,000 steps at epoch 36. At epoch 38 the ADAM optimiser was reset with a lower learning rate. The parameters after epoch 45 are used in the results.

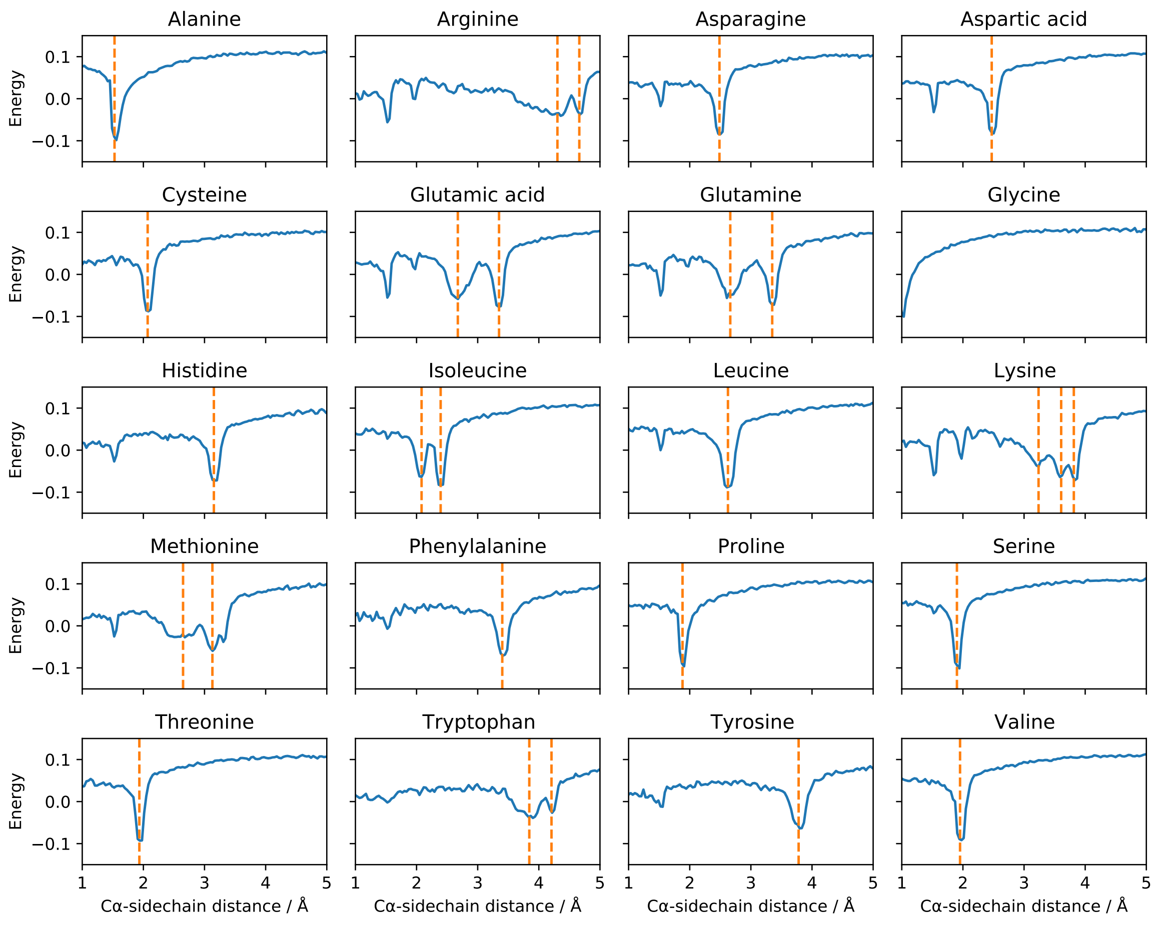


Figure S2 Learned distance potentials between the Cα atom and the sidechain centroid for each amino acid. The orange lines indicate minima in the PDB distributions of these distances, corresponding to different rotamers.

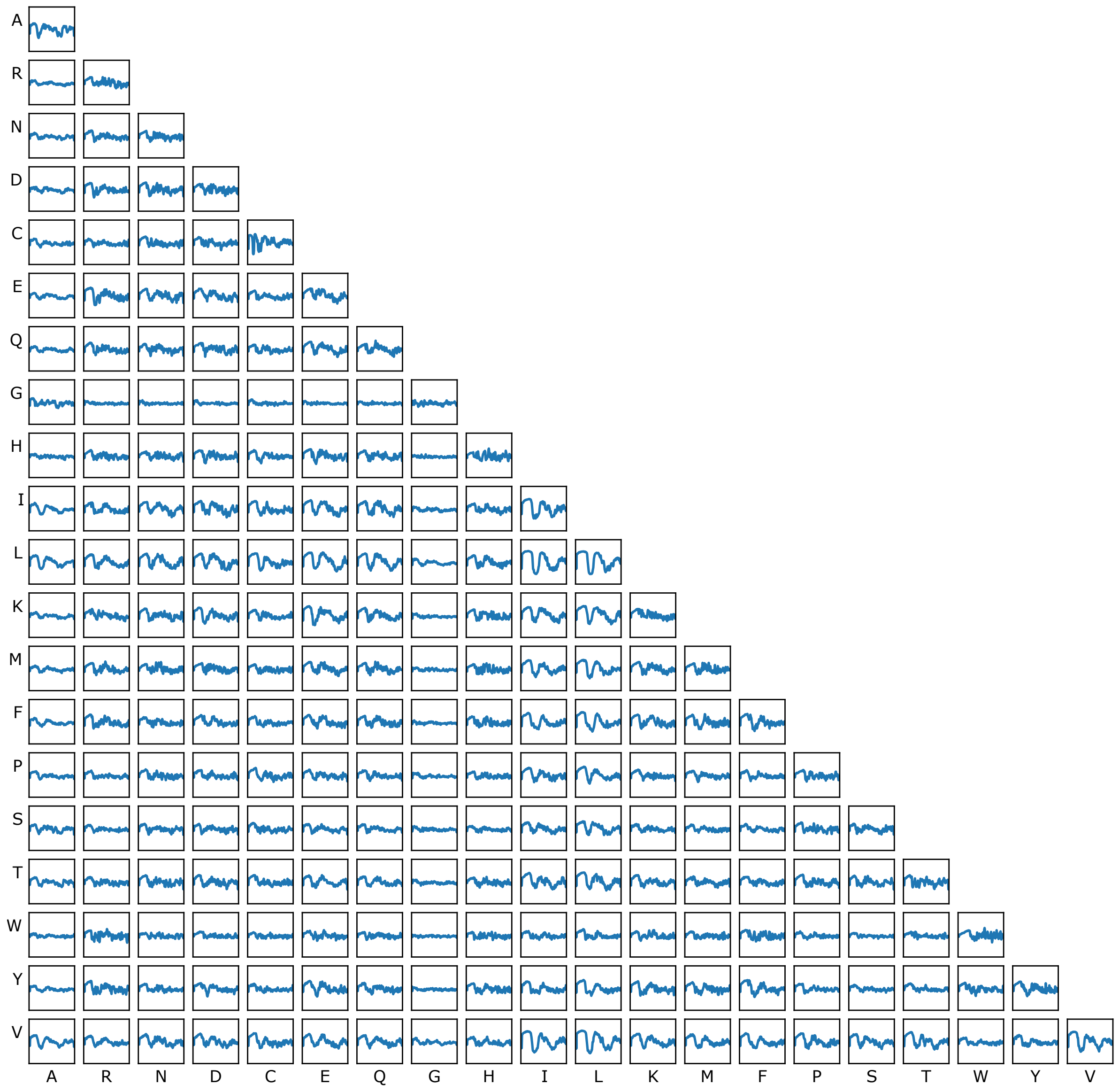


Figure S3 Learned distance potentials between the sidechain centroids for each amino acid pair. On each plot the *x*-axis is the distance between the sidechain centroids and runs from 1 Å to 15 Å. The *y*-axis is the energy in the learned potential and runs from -0.2 to 0.23. There are separate potentials for residues close in sequence, not shown here.

Protein or peptide	**PDB ID**	**Sequence length**	**Sequence**	**Predicted secondary structure**
Alanine dipeptide	-	2	AA	CC
$(AAQAA)_3$ repeat peptide	-	15	AAQAAAAQAAAAQAA	CHHHHHHHHHHHHHC
Chignolin	-	10	YYDPETGTWY	CCCCCCCCCC
Trp-cage	2JOF_A	20	DAYAQWLADGGPSSGRPPPS	CHHHHHHCCCCCCCCCCCCC
BBA	1FME_A	28	EQYTAKYKGRTFRNEKELRD FIEKFKGR	CCCCCCCCCCCCCCHHHHHH HHHHHCCC
Villin HP36	2F4K_A	35	LSDEDFKAVFGMTRSAFANL PLWLQQHLLKEKGLF	CCHHHHHHHHHCCHHHHHCH HHHHHHHHHHHCCCC

Table S1 Details of the proteins and peptides used. The PDB ID and chain ID are given along with the amino acid sequence used for modelling and the PSIPRED single sequence secondary structure prediction [69]. The chignolin structure was taken from the supplementary data of Honda et al. 2008 [48]. In the cases where the model is an NMR ensemble (chignolin, Trp-cage and BBA), the first model in the ensemble was used when calculating RMSD values.