PLOS ONE

RESEARCH ARTICLE

Predicting improved protein conformations with a temporal deep recurrent neural network

Erik Pfeiffenberger, Paul A. Bates*

Biomolecular Modelling Laboratory, The Francis Crick Institute, 1 Midland Road, London NW1 1AT, United Kingdom

* paul.bates@crick.ac.uk

OPEN ACCESS

Citation: Pfeiffenberger E, Bates PA (2018) Predicting improved protein conformations with a temporal deep recurrent neural network. PLoS ONE 13(9): e0202652. https://doi.org/10.1371/journal.pone.0202652

Editor: Attila Gursoy, Koc Universitesi, TURKEY

Received: April 10, 2018

Accepted: August 7, 2018

Published: September 4, 2018

Data Availability Statement: The source code is available from https://github.com/OneAngstrom/DeepTrajectory. The data used in this work is available for download from https://zenodo.org/record/1183354.

Funding: This work was supported by the Francis Crick Institute, which receives its core funding from Cancer Research UK (FC001003), the UK Medical Research Council (FC001003), and the Wellcome Trust (FC001003). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Abstract

Accurate protein structure prediction from amino acid sequence is still an unsolved problem. The most reliable methods centre on template based modelling. However, the accuracy of these models entirely depends on the availability of experimentally resolved homologous template structures. In order to generate more accurate models, extensive physics based molecular dynamics (MD) refinement simulations are performed to sample many different conformations to find improved conformational states. In this study, we propose a deep recurrent network model, called DeepTrajectory, that is able to identify these improved conformational states, with high precision, from a variety of different MD based sampling protocols. The proposed model learns the temporal patterns of features computed from MD trajectory data in order to classify whether each recorded simulation snapshot is an improved quality conformational state, decreased quality conformational state or whether there is no perceivable change in state with respect to the starting conformation. The model was trained and tested on 904 trajectories from 42 different protein systems with a cumulative number of more than 1.7 million snapshots. We show that our model outperforms other state of the art machine-learning algorithms that do not consider temporal dependencies. To our knowledge, DeepTrajectory is the first implementation of a time-dependent deep-learning protocol that is re-trainable and able to adapt to any new MD based sampling procedure, thereby demonstrating how a neural network can be used to learn the latter part of the protein folding funnel.

Introduction

Protein structure prediction from amino acid sequence tries to overcome the limitations of experimental structure determination, which are often time consuming and infeasible for certain types of proteins. Furthermore, construction of protein models seems to be the only practical solution for structural genomics where a high rate of newly discovered protein sequences demands for automated structure determination [1]. Current state-of-the-art methods which make use of template based modelling (TBM) are partially successful [2–8]. However, the

Additionally, the NVIDIA Corporation provided a donation of materials (i.e. GPUs) to the authors.

Competing interests: The NVIDIA Corporation provided a donation of materials (i.e. GPUs) to the authors.

quality of these TBMs is completely dependent on the presence of homologous proteins where the structure has been experimentally determined. A useful extension to TBM is so-called physics-based refinement methods that further try to improve the initial models by extensively sampling new conformations; essentially, emulating the later part of the protein folding pathway. Methods which make use of conformational sampling with molecular dynamics (MD) simulations, employing an all-atom physical force field, have proven to be successful in sampling improved conformational states. Currently, the most successful refinement simulations are based on multiple replicated simulations in the nanosecond scale with position restraints on parts of the protein to prevent drifts [9–11]. Yet, the most challenging problem is the reliable identification of improved quality configurations from this time-series trajectory data, from millions of possible solutions [12–15].

Continued progress in deep-learning research has demonstrated success for a number of noisy sequence or time-series problems [16–18]. In this work, a temporal deep-learning model for MD snapshot classification has been formalized that makes explicit use of the time-dependent nature of MD based trajectory data. The used trajectory data contains the information of a protein's conformational changes, such as unfolding or folding events, as a function of simulation time. In particular, the interest lies in whether it is possible to identify when, or if, improved quality conformations of a protein are reached, from a variety of starting model qualities. Progress in this area is important for high accuracy model building that is, for example, required for biomolecular understanding of protein function and *in-silico* rational drug design. From the generated trajectory of conformational snapshots, predictions about a protein's conformational states are based on energies and distance metrics in time. To this end, a deep recurrent neural network (RNN) [19] with gated recurrent units (GRUs) [20] is trained to classify each snapshot into one of three classes: improved quality, no-change in quality, and decreased quality. The change of quality is defined as an increase or decrease in the global distance test total score (GDTTS) [21, 22] from the starting configuration, as measured with respect to the reference crystal structure.

The results show that it is possible to train a RNN model that identifies improved and decreased quality states in different MD based refinement protocols with nanosecond timescale. Furthermore, the proposed model outperforms classic machine learning models and deep learning models that do not consider temporal dependencies during their training task. To be precise, our model achieves a mean cross-validation precision on the improved state assignment of 41.5% compared to 14.0%, 12.1% and 0.0% for random forest (RF) [23], k-nearest neighbours (KNN) [24], and logistic regression (LR) [25], respectively. The results also show that a deep representation and temporal patterns learned by the RNN are important and contribute to a higher precision of identifying improved quality snapshots.

Results

The DeepTrajectory model

The DeepTrajectory model is summarized in Fig 1. The model takes as input the computed features from sampled MD trajectory data and tries to predict the state change relative to the starting configuration for each trajectory snapshot of a refinement simulation of a protein. The three classes learned by the classifier are improved state (*I*), no-change state (*N*) and decreased state (*D*), and reflect whether a more accurate conformation of the protein is reached with respect to the reference crystal structure (See Fig 1A). The trajectory is quantified via 19 metrics. These are 17 different potential energy terms and scoring functions (quantifying the energetics of the protein structure); and 2 distance-metrics known as root mean square deviation (RMSD) and GDTTS (measuring the deviation to the starting configuration at time-step 0).

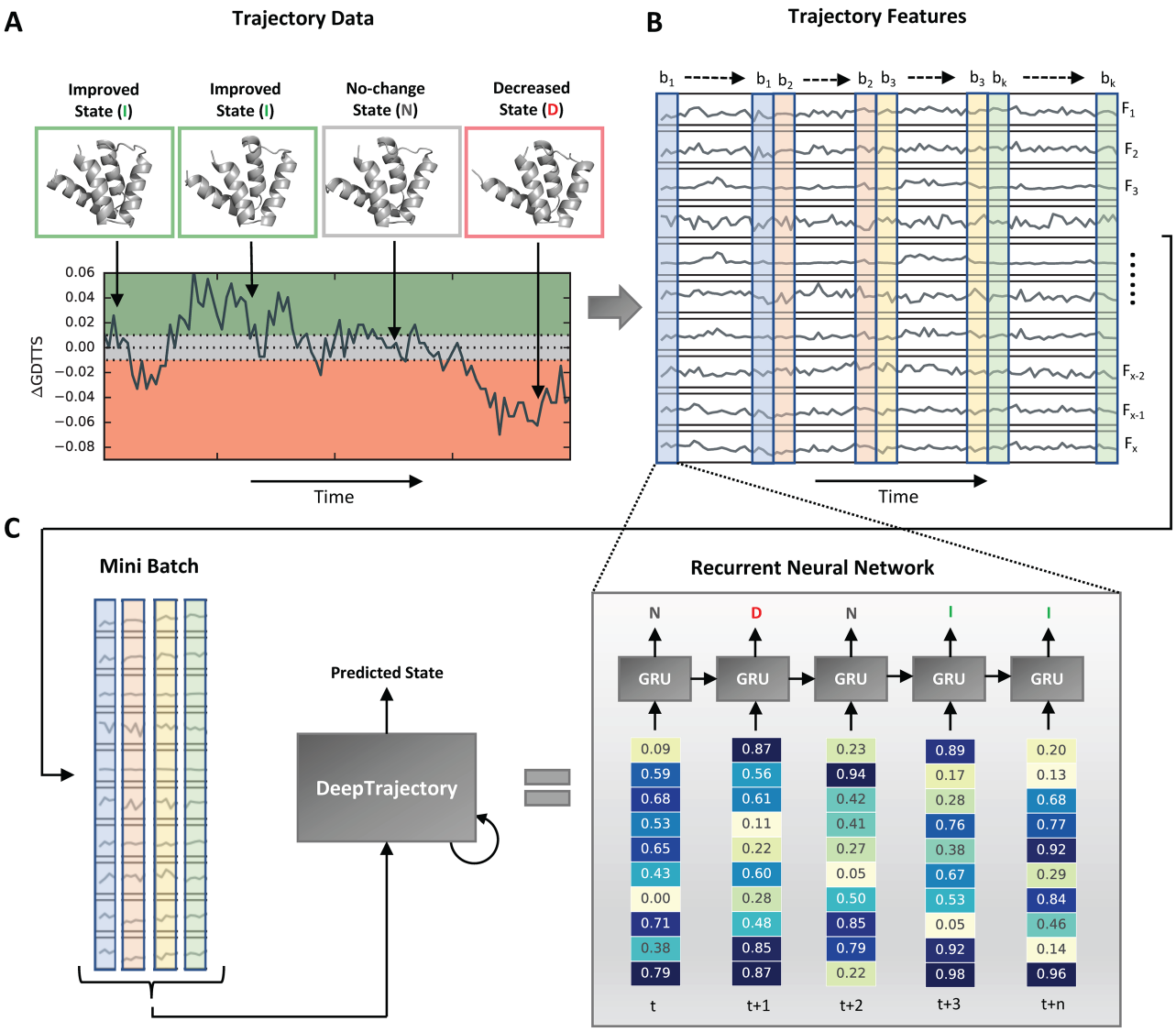

Fig 1. DeepTrajectory method overview. The method predicts improved conformational states, that more closely resemble the crystal structure observed conformation, in molecular dynamics (MD) trajectory data of template based models. (A) During sampling of the initially folded proteins with different MD based sampling protocols, transitions to conformations can be observed that represent an improved conformational state ($\Delta GDTTS \geq 0.01$), decreased conformational state ($\Delta GDTTS \leq -0.01$) and no-change in conformational state ($|\Delta GDTTS| < 0.01$). Every 2 ps a snapshot of this trajectory is saved. (B) The trajectory of snapshots of different sampled conformational states is quantified by 19 features ($F_1, \cdots, F_x$) that measure different energetic contributions or distance metrics of the protein structure at a particular time-point. (C) These temporal features are used to perform supervised mini-batch training ($b_1, \cdots, b_k$) to train an RNN with several layers of GRUs that is able to classify each snapshot of the trajectory into three classes: improved state (I), no-change state (N), decreased state (D). Predictions for the trajectory of a new protein system, for which this state change assignment is unknown, are assigned by applying the trained DeepTrajectory model to every snapshot. Full details of the model and training procedure are available in the methods section.

https://doi.org/10.1371/journal.pone.0202652.g001

The use of a set of diverse features in our model is motivated by learning non-linear relationships in order to improve the predictive power and is common practice in applied machine learning for structural bioinformatics [26–28]. The use of distance metrics as features was inspired by work of Mirjalili et al. that showed that these combined with an energy function can be beneficial [9]. Details about the features are summarized in the materials and methods section. Essentially, our DeepTrajectory model learns the temporal patterns of input features

in order to distinguish correct fold state changes from incorrect fold state changes. The trajectory data from the different protein systems and sampling runs used for training is presented to the network in mini-batches of 60 ps (30 steps) that continue after each iteration of training until the end of the training data is reached (Fig 1B). The DeepTrajectory model is an RNN with GRUs (Fig 1C). Each 2 ps time-step is represented by a standardized feature vector containing the values of the 19 features. The predicted state assignment for every snapshot by the RNN, i.e. *I*, *N* and *D*, is expressed as a probability distribution and the class with the highest probability is used as the final prediction.

Data-set and performance criteria

The performance of DeepTrajectory was compared to a RF classifier, a KNN classifier and a LR classifier. The data used for training and testing accumulates to 904 trajectories and a total simulation time of 3419 ns from 42 different protein monomers. The used protein systems and their starting model quality were collected from the refinement category in rounds 11 and 12 from the Critical Assessment of protein Structure Prediction (CASP) experiment. The dataset consists of a wide range of GDTTS values from 0.3 to 0.9 (Fig 2A). The sampled ΔGDTTS, that expresses the relative change in model quality relative to the starting model, ranges from −0.3 to 0.12 (Fig 2B), details for each trajectory are shown in S2 Table. The class assignment of each snapshot, shown in Fig 2C, into one of the three different states *I*, *N* and *D*, have a relative distribution of 8.2, 14.5 and 77.3 percent, respectively. An analysis of the trajectory data as a Markov chain model shows the transition probabilities between the different states (Fig 2D). These show that increased and decreased conformational states are more stable with a probability of 0.806 and 0.947 to remain in the same state, compared to no-change state with 0.626. This is also expressed by the observation that these states sample with higher frequency longer continuous segments, see Fig 2E (improved state) and Fig 2F (decreased state), as compared to the no-change state (Fig 2G).

The aim of the classifier is to learn a temporal model in order to identify improved conformational states (*I*) with high precision, and decreased conformational states (*D*) with high recall. Thus, during training we are trying to minimize the number of false positive predictions for the improved state and the number of false negative predictions for the decreased state class. This results in a model with higher confidence that the predicted *I* state is correct and that a large number of *D* states could be identified. Additionally, the metric F1 is computed that is the harmonic mean of precision and recall. We compare this to all three base-line machine learning models. A detailed definition of the used performance metrics is given in the methods section.

Comparison of model performance to other classifiers

The bar-plots seen in the panels of Fig 3A–3C quantify the classification performance for all three classes *I*, *N* and *D* of the temporal RNN model and compare it to KNN, RF and LR. Most notably is the prediction performance of the RNN for the improved state. The RNN is able to identify improvements in folds with a markedly better precision than classical machine learning models. To be precise, the mean cross-validation precision for RNN, KNN, RF and LR have values of 0.415, 0.121, 0.139 and 0.000, respectively. For recall on the improved class the values are 0.037, 0.065, 0.001 and 0.000 for RNN, KNN, RF and LR, respectively. Values for the decreased class performance are similar for all models (Fig 3C). For this class the models RNN, KNN, RF and LR produce a mean cross-validation precision of 0.790, 0.798, 0.778 and 0.777, respectively. The recall is 0.960, 0.888, 0.985 and 0.987 for RNN, KNN and RF and LR, respectively.

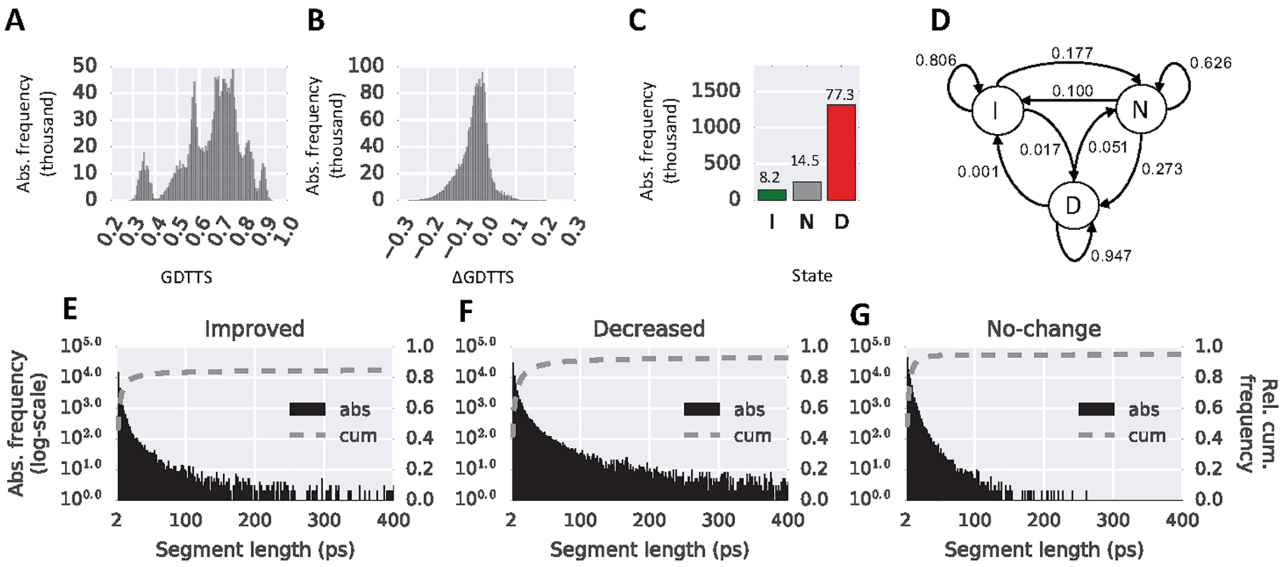

Fig 2. Trajectory dataset. (A) Histogram of GDTTS in the trajectory data. (B) Histogram of ΔGDTTS in the trajectory data. A positive value indicates an improvement and a negative value a decrease in model quality. (C) Absolute frequency of the three different states improved, *I*, no-change, *N*, and decreased, *D*. (D) Markov chain model of the three states improved, *I*, no-change, *N*, and decreased, *D*, visualised as circles and their directed transition probabilities shown as labelled arrows. (E)-(G) Histograms of the continues segmentation length shown as absolute frequencies and cumulative frequencies for the three states (E) improved, (F) decreased and (G) no-change.

https://doi.org/10.1371/journal.pone.0202652.g002

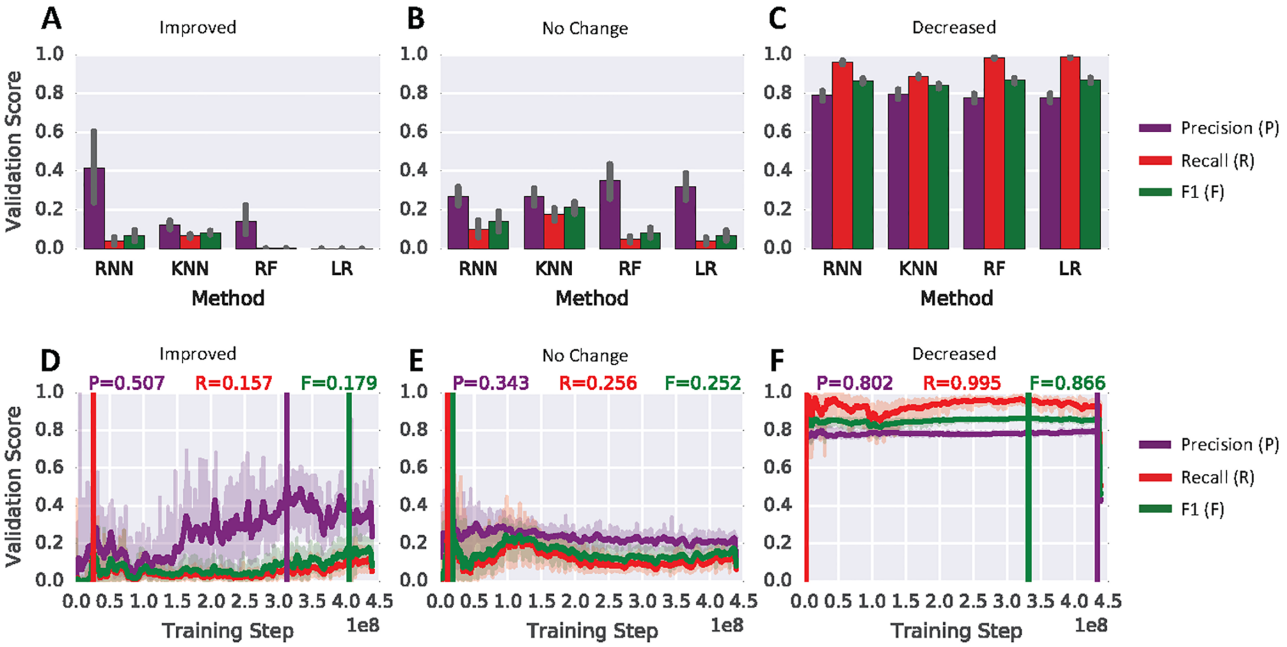

Fig 3. RNN performance comparison. (A)–(C) Comparison of validation-set performance for the 3 different states improved, no-change and decreased. Shown are the recurrent neural network (RNN) against k-nearest neighbours (KNN), random forest (RF) and logistic regression (LR) on the full cross-validation sets. The shown performance is the mean value of all seven validation sets. The error bars indicate the standard deviation computed from all 7 folds. (D)-(E) Classification precision, recall and F1 as a function of training steps for the three different states. Shown is the training progress for CV fold 4. The vertical lines indicate the best performance for each metric based on a moving average with a window-size of 30.

https://doi.org/10.1371/journal.pone.0202652.g003

Table 1. Classification confusion for improved state predictions. The table compares the results for predicted improved states to their actual ground truth state assignment for RNN, KNN, RF and LR.

		Predicted State (Improved)			
		RNN	KNN	RF	LR
Actual	I	1636	1556	34	0
	N	314	2386	25	0
	D	1653	6690	39	0
	Total	3603	10632	98	0

https://doi.org/10.1371/journal.pone.0202652.t001

The shown confusion in classification for improved state predictions in Table 1 shows the miss-assignment of predicted classes versus the actual class for all 4 tested models for validation-fold 4 of the cross validation (CV). For example, the RNN model predicted the correct true positive assignment for improved snapshots 1636 times and assigned the label improved incorrectly 314 times to no-change snapshots and 1653 times to decreased snapshots. Compared to the three other models KNN, RF and LR this represents a notably better performance at identifying improved snapshots. For KNN a similar number, i.e. 1556, of true positive improved snapshot assignments compared to RNN could be achieved. However, this comes with a large number of false positive assignments where the KNN incorrectly assigns the improved label to 2386 no-change snapshots and 6690 decreased snapshots. The RF model is hardly predicting the improved class. This model generates 34 true positive assignments for the improved class. The number of false positive predictions, where the actual class is different from the predicted, is 25 and 39 for no-change and decreased, respectively. The LR model is not able to predict improved snapshots at all, i.e. the number of assignments is zero.

Fig 3D–3F shows the validation score for precision, recall and F1 as functions of training steps for the three classes (*I*, *N* and *D*). The improved class shown in Fig 3D indicates that several million training steps are necessary to reach the best running average precision of 0.507. The best precision and recall for classes no-change (Fig 3E) and decreased (Fig 3F) are reached early on during training. Furthermore, for these two classes the validation score stays stable during the 300 epoch training process.

In order to assess whether the distance metric features (RMSD_SM and GDTTS_SM) are essential for classification performance of our RNN model, a second version was trained without these features. The largest effect on prediction performance is observed for the no-change state (see S3 Fig). The mean cross-validation performance for precision decreased from 0.269 to 0.227, for recall from 0.101 to 0.045 and for F1 from 0.140 to 0.069. However, only the change for F1 produced a significant result with a p-value < 0.05. The prediction performance for the two other classes, improved and decreased state, remained largely unchanged with no significant change in mean validation score for all three metrics (see S3 Fig).

Analysis of prediction performance

Fig 4A shows DeepTrajectory's true positive (TP), false negative (FN) and false positive (FP) predictions as cumulative distributions of ΔGDTTS for the improved state class. The results show a marked increase of FP predictions as ΔGDTTS tends towards the lower bound of the increased state definition (ΔGDTTS = 0.01). TP predictions have the highest increase in density for ΔGDTTS in the range from 0.01 to 0.05. For FN predictions of the improved state, the curve is less steep in the range 0.025 to 0.1 compared to TP, indicating fewer FN assignments. The no-change state cumulative distribution of FP predictions shows a rapid increase in the ΔGDTTS range from -0.05 to -0.01, with only a shallow increase for 0.01 to 0.05 (S4A Fig). The cumulative distribution of FN and TP follow the same trend for no-change state

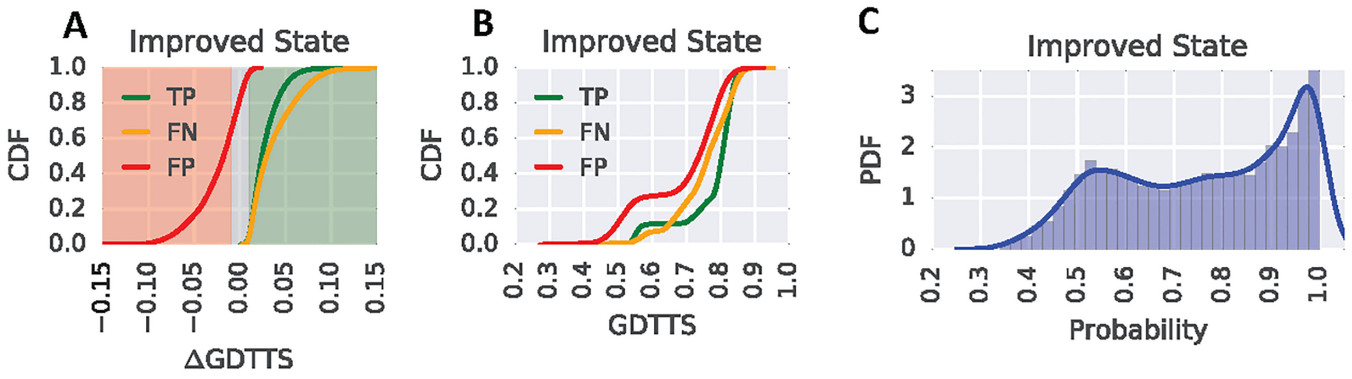

Fig 4. Performance analysis of DeepTrajectory. (A) Cumulative distribution of true positive (TP), false negative (FN) and false positive (FP) predictions for the improved state as a function of ΔGDTTS. The background colours red, gray and green indicate the ΔGDTTS regions for the improved, no-change and decreased states, respectively. (B) Cumulative distribution of true positive (TP), false negative (FN) and false positive (FP) predictions for the improved state as a function of GDTTS. (C) Shows the distribution of assigned probabilities for improved state predictions.

https://doi.org/10.1371/journal.pone.0202652.g004

predictions (S4A Fig). For decreased state predictions most FP are distributed in the range from ΔGDTTS -0.01 to 0.01. The cumulative distribution of FN stays below the distribution of TP for the ΔGDTTS range -0.15 to -0.01 (S5A Fig).

The cumulative density of TP, FN and FP as a function of absolute GDTTS is visualized in Fig 4B for improved state predictions. A more rapid increase in FP predictions for GDTTS in the range 0.45 to 0.7 as compared to TP and FN is observed. This indicates that correct predictions of lower starting model qualities are less likely. However, TP predictions markedly increase for GDTTS in the range 0.7 to 0.85, showing the successful prediction of the descent to the native state of the latter part of the folding funnel. The cumulative TP distribution of the no-change state shows a similar behavior, where only higher quality models with GDTTS of 0.5 or more produce an increase in TP (S4B Fig), whereas the decreased state distribution of TP, FN and FP shows a linear growth for all three curves (S5B Fig).

In order to obtain an understanding of the confidence of DeepTrajectory in assigning the three different states, the predicted probabilities are considered as a variable and the probability density function of these probabilities were constructed from the whole data-set. Our model produces a wide range of assigned probabilities for predicted classes improved and no-change state (Fig 4C and S4C Fig) where the majority of probabilities are uniformly distributed from values of 0.5 to 0.9, with a slight increase in density in the range from 0.9 to 1.0. The probability density function for predicted decreased states is markedly different, most of the RNN's computed probabilities are observed in the range from 0.9 to 1.0, whereas the range from 0.4 to 0.9 has a low density (S5C Fig).

Temporal and deep representation is important for precision and recall

We analyzed how important the temporal aspect of our RNN for classification success is. An RNN with sequence length of 1 (Fig 5A) and 5 (Fig 5B) experience training instability and a drastic drop in precision and recall for the improved state class compared to a model of length 30 (Fig 5C) and 50 (Fig 5D). To further test the benefit of temporal pattern-learning on classification performance an additional experiment was performed that randomized the original snapshot order of each trajectory in the training set. The results suggest instability in the training process for the improved state class, as seen in S6A Fig. During training, the validation-set precision fluctuates strongly and recall and F1 drop further when compared to the learning curves on the original trajectory data (Fig 3D). The learning performance for the two other

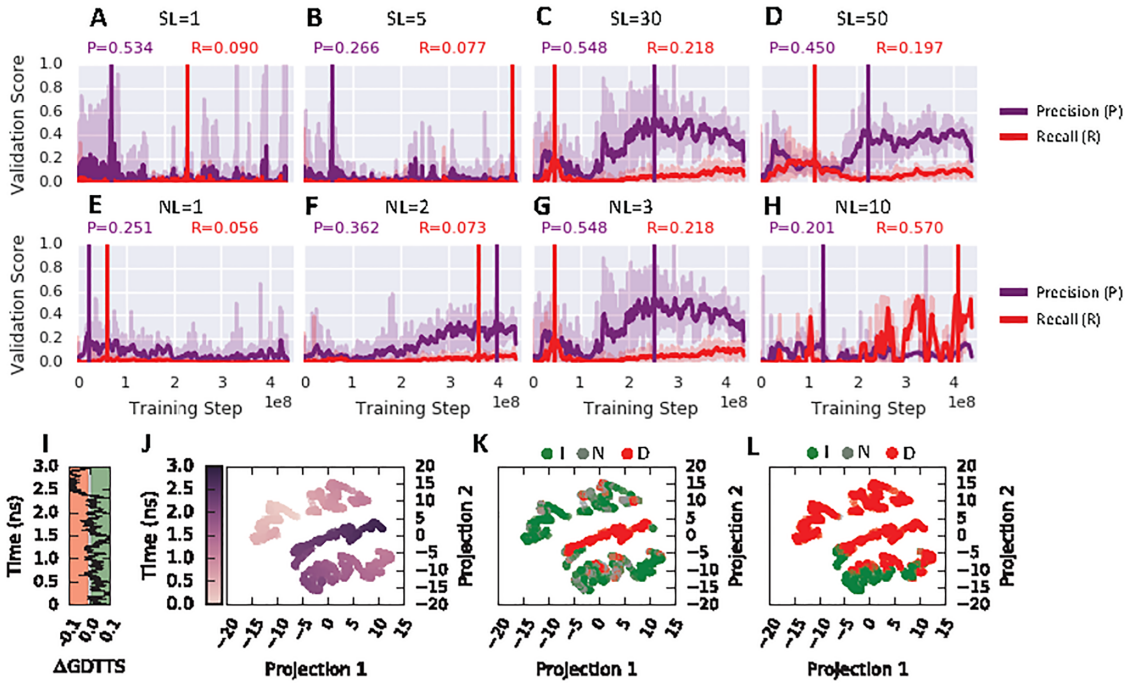

Fig 5. (A)–(D) Effect of sequence length (SL) 1, 5, 30 and 50 on precision and recall for the improved state class as a function of training steps tested on validation fold 4. (E)–(H) Effect of number of hidden layers (NL) with values 1, 2, 3 and 10 on precision and recall for the improved state class as a function of training steps tested on the 4th validation fold. The vertical lines indicate the best performance based on a moving average with a window-size of 30. (I) Trajectory trace of target TR821, the ΔGDTTS is shown as a function of its 3 ns simulation time. Sampling of the green colored regions indicate an improved state, grey regions a no-change state and red a decreased state. Visualizations of the last hidden layer where the data-points are colored by (J) time, (K) ground truth and (L) predicted labels. The 1024-dimensional last hidden layer was projected to 2-dimensions with t-SNE.

https://doi.org/10.1371/journal.pone.0202652.g005

classes, no-change and decreased state, remained unaffected by trajectory randomization (S6B and S6C Fig). Furthermore, the depth of the RNN is an important aspect too. An RNN with 1 (Fig 5E) or 2 (Fig 5F) layers results in a drop in precision and recall compared to 3 layers (Fig 5G). Furthermore, a drop is observed when 10 layers (Fig 5H) are used within the tested training time of 300 epochs.

In order to better understand how the temporal aspect of our trajectory data is learned by the RNN, we analysed the output of the last hidden layer. We examined the internal feature representation learned by the RNN using t-distributed Stochastic Neighbour Embedding (t-SNE) [29]. As an example, a trajectory of one protein system is visualised. In Fig 5I, the ΔGDTTS change as a function of time for a 3 ns simulation is shown. The learned representation by the RNN of this trajectory is shown in Fig 5J–5L, where each point represents a projection from 1024 to 2 dimensions from the output of the last hidden layer. The projection in Fig 5J is coloured by simulation time. We can see that points close in time also cluster together close in the projected space. The projections in Fig 5K and 5L are coloured by the ground truth label and the predicted label, respectively. The projections of the predicted improved class shows pattern of aggregation into the same region.

Discussion

The results show that the proposed RNN model with GRU cells is able to outperform classical machine learning methods such as RF, LR and KNN, which are representative of state-of-the-

art classical machine learning algorithms and have been successfully applied to other bioinformatic domains [30–32]. In particular, the model presented here achieves a mean precision of 0.415 on the validation set of the CV compared to 0.121, 0.140 and 0.000 for KNN, RF and LR, respectively. This suggests that learned temporal dependencies of the used energy terms and distance metrics as input features are important to identify sections of fold improvements in the trajectory. This claim is further supported by inspection of the transition probabilities between *I*, *N* and *D* in Fig 2D that are not random. The transition probabilities of staying in their respective states from time-step t to t+1 are 0.806, 0.626 and 0.947 for *I*, *N*, *D* respectively, and indicates that sequential state awareness is important for this particular classification problem.

A main issue for successfully training machine-learning models, for real-world applications, is class imbalance of the available training data. This is a problem associated with our MD trajectory data, as indeed with other applications in bioinformatics such as ranking of 3D models of protein-protein docking solutions [30], protein function prediction [33] or prediction of protein-protein interaction networks [34]. The problem of class imbalance is commonly addressed by under or oversampling of the training data where data-points from the over represented class are removed, or more data is collected to obtain more examples for the under-represented class, respectively. However, for the case of time-series data under-sampling cannot easily be performed if the temporal patterns are to be conserved. Likewise, oversampling, i.e. generating more MD trajectories, would not solve the problem. The underlying process of conformation sampling with MD would still result in the same imbalanced distribution of improved, no-change and decreased states. RF classifiers are especially affected by class imbalance [35] as our results confirm, which produces a low numbers of TP predictions compared to KNN that are more robust against class imbalance [25]. Another alternative to cope with class imbalance in machine-learning is cost-sensitive learning. In case of our DeepTrajectory model we introduced a weighted cross-entropy function in order to increase the classification performance for the increased state class. For the base-line models, RF and LR, class weights could be applied that weight classes to be inversely proportional to their training set frequency with a potential benefit on prediction performance.

A major application of DeepTrajectory would be in on-line classification of MD refinement simulations. Here, the model would be applied in parallel to a MD simulation to classify each snapshot of the trajectory. This would allow to probe whether the sampling has yielded enough improved conformational states and can be stopped, or vice versa, if no snapshots are classified as improved during the sampling run the simulation could be continued or restarted with different conditions; however, more work is needed to develop this concept into an easy to use methodology for structure refinement. Moreover, the presented methodology could be extended to complete folding simulations by guiding the sampling process from a partially folded state to a near native state [36]. For example, this could be used in combination with molecular dynamics and Markov-state models in order to classify micro-states [37, 38]. This would allow for selecting the correct micro-states that point to a descent of the folding funnel. Finally, similar to the RNN applied to trajectories of protein monomers, the proposed model could be trained on the trajectories of protein-protein assemblies [39, 40]. Input features for such a method would need to focus on molecular descriptors that specifically describe protein-protein interactions [41, 42].

We believe DeepTrajectory makes an important contribution to the field of protein structure refinement, with applications in rational drug design [43] and systems biology [44]. Our work has shown that a temporal model based on a RNN can predict, with high precision, the state changes to improved protein conformations. This opens the door for more application driven research exploiting temporal learning and deep learning as means to improve accuracy of protein structure prediction.

Materials and methods

Data and source code availability

The source code of the model as well as an example of how to train it is available from https://github.com/OneAngstrom/DeepTrajectory. The data used in this work is available for download from https://zenodo.org/record/1183354. This data contains the PDB files of the raw trajectories, starting models and reference PDB structures; and a comma separated file that contains the pre-computed trajectory features and labels.

The model

The DeepTrajectory model is described in Fig 1. The model is implemented and tested on the TensorFlow Python library in version 1.0.0 [45]. Essentially, the model learns via a supervised learning-task to assign the class y from a set of given input features, x, for each time-point, $[\tau]_i$, of a trajectory v. Here, the three possible classes are improved, no-change and decreased. The ground truth assignment, denoted as y', is then formalized such that

$$y' = \begin{cases} \boldsymbol{i} & \text{if } \Delta\,\text{GDTTS} \geq 0.01 \\ \boldsymbol{n} & \text{if } \Delta\,\text{GDTTS} < 0.01 \text{ and } \Delta\,\text{GDTTS} \geq -0.01 \\ \boldsymbol{d} & \text{if } \Delta\,\text{GDTTS} < -0.01, \end{cases} \tag{1}$$

where $\boldsymbol{i}$, $\boldsymbol{n}$ and $\boldsymbol{d}$ represent one-hot encoded probability vectors [1, 0, 0], [0, 1, 0] and [0, 0, 1], respectively. The variable ΔGDTTS is the difference between GDTTS from the starting model at time-point zero and the GDTTS from the snapshot at time-point t as computed to the reference crystal structure. A negative ΔGDTTS value reflects a decrease in model quality and a positive ΔGDTTS value an increase in model quality.

The model is based on an RNN with GRU that adaptively learns long and short term dependencies of inputs to assign the class y [20]. The layout of the RNN is illustrated in S1A Fig, where starting from the input sequence $x_0, \ldots, x_t$ the predictions $y_0, \ldots, y_t$ are produced via stacking hidden layers of GRU cells and by a layer with a softmax activation function Ω that normalizes the output h_t^l (at time t of the last layer l) to a probability vector y such that

$$[y_t]_j = \Omega_j(h_t^l) = \frac{\exp(W_j^l h_t^l)}{\sum_{j'}^{C} \exp(W_{j'}^l h_t^l)}, \tag{2}$$

for all $j = 1, \ldots, C$ classes, where W_j^l are the rows of the weight matrix of the last layer. The activation and its output, h_t^l in layer l at time t, of a GRU cell is computed as

$$h_t^l = z \odot h_{t-1}^l + (1 - z_t) \odot \tilde{h}_t^l. \tag{3}$$

This represents a linear interpolation of the activation at time-point $t - 1$ denoted as h_{t-1}^l and its candidate activation $\tilde{h}_t^l$. The update gate, z, controls how much the cell updates its state, such that

$$z = \sigma(W_z^l h_t^{l-1} + U_z^l h_{t-1}^l), \tag{4}$$

where the activation function σ is sigmoid. The candidate state $\tilde{h}_t^l$ is computed such that

$$\tilde{h}_t^l = \phi(W^l h_t^{l-1} + U^l(r \odot h_{t-1}^l)). \tag{5}$$

Here, ϕ, r and $\odot$ denote a hyperbolic tangent activation function, a reset gate and an element wise multiplication, respectively. The reset gate, r, is computed with the same formulation as z but different weight matrices, i.e.

$$r = \sigma(W_r^l h_t^{l-1} + U_r^l h_{t-1}^l). \tag{6}$$

An illustration of these equations is shown in S1B Fig.

During training the weight matrices W^l, U^l, W_r^l, U_r^l, W_z^l and U_z^l are learned for each layer l. The weight matrices are shared through time t. The loss function L

$$L_{y'}(y) = -\sum_j \omega_j (y_j' \log(y_j)), \tag{7}$$

is minimized during training and represents the weighted cross entropy. The vector $\boldsymbol{\omega}$ encodes the weights for classes $j = 1, \ldots, C$. The objective of this classifier is to achieve a high precision for the improved class (i.e. reducing the false positive rate) and a high recall for the decreased class (i.e. reducing false negative rate). This is achieved by setting $\boldsymbol{\omega} = [0.05, 1, 10]$.

The RNN model is trained with the Adam optimizer [46] on input features x from the set of trajectories v selected for training. In order to achieve one sequential input for the training, all n trajectories are concatenated to size $\tau \times n$. The training is performed on k mini-batches b of the data where in each training iteration the batch b_k continuous without overlap to the previous batch iteration till the epoch is finished. This process is visualized in S1C and S1D Fig.

Data set

The trajectory data originates from our own laboratory's refinement method in CASP11 and CASP12 for which the reference crystal structure is available in the PDB. These targets are listed in S1 Table. The detailed description of the sampling process is described in section "Sampling procedure". In total, the trajectory data consists of 3419 ns cumulated simulation time and 1, 709, 704 snapshots with $\Delta t = 2$ ps from 30 CASP11 and 12 CASP12 targets for which crystal structures were available. A detailed overview of the generated trajectories for each target and their snapshot composition is provided in S2 Table.

In total 19 features were used. 17 of these features originate from ten different potential energy functions and two features are the distance metrics GDTTS and RMSD that measure for each snapshots the difference to the starting model. All molecular descriptors are normalized per target to zero mean and unit standard deviation. The complete list of features is given in Table 2.

Cross-validation

The CV set is made up of 7 folds, where for each fold the training set consists of trajectories of 36 targets and the validation set for 6 targets. The assignment of a proteins trajectories to a fold is random. However, the relative distribution of classes of snapshots between training and validation set is enforced to be similar with a maximum difference of 6 percent as shown in Table 3 columns I, N and D. A detailed overview of each targets fold assignment is given in S2 Table.

Model hyper-parameter

The RNN model was trained for every fold of the CV for 300 epochs with the following hyper-parameter: sequence length = 30, batch-size = 50, internal size = 1024, number of layers = 3, learning rate = 0.0001 and dropout with p-keep = 0.9.

Table 2. RNN features. Table lists the feature name, the descriptor that produced the feature and the reference for the descriptor.

Feature	Descriptor	Reference
N_DDFIRESUM	DFIRE	[47]
N_DDFIRETERM1	DFIRE	[47]
N_DDFIRETERM2	DFIRE	[47]
N_DDFIRETERM3	DFIRE	[47]
N_DDFIRETERM4	DFIRE	[47]
N_DOPE	DOPE	[48]
N_DOPE_HR	DOPE	[48]
N_RMSD_SM	RMSD to starting model	NA
N_GDTTS_SM	GDTTS to staring model	NA
N_DOOP	DOOP	[49]
N_CALRW	calRW	[50]
N_CALRWP	calRWplus	[50]
N_GOAP	GOAP	[51]
N_GOAPAG	GOAP	[51]
N_BOND	Modeller	[52]
N_ANGLE	Modeller	[52]
N_DIHEDRAL	Modeller	[52]
N_IMPROPER	Modeller	[52]
N_MOLPDF	Mol. PDF	[52]

https://doi.org/10.1371/journal.pone.0202652.t002

Baseline models

The python package scikit-learn [53] in version 0.18.1 was used to perform the training and testing of the baseline models. The same features, class-labels and CV folds as shown in Table 3 were used.

Random forest. The training of the random forest classifier [23] uses 500 trees where samples are bootstrapped and the gini impurity criterion is used to judge the quality of the split when building the trees. No restriction for the maximum depth of the tree is imposed, however, for each internal node in a tree the minimum sample size must be greater than 30. The number of features for each tree is $\sqrt{n}$ where n = 19, i.e. the total number of features.

K nearest neighbour. For the K nearest neighbour classifier [24] the number of neighbours and the leaf-size was set to 5 and 30, respectively. A uniform distribution where all points are weighted equally in each neighbourhood was chosen. The algorithm to search for the nearest neighbours was set to 'auto' where the best algorithm from ball-tree [54], kd-tree [55] and a brute force approach was selected for fitting the model with the Minkowski distance metric with $p = 2$, which is equivalent to the Euclidean distance.

Logistic regression. Fitting of the logistic regression model [25] is performed with L2 regularization with a strength of 1.0 and with a tolerance of $1e - 4$ as the stopping criteria. In order to make the multi-class predictions with LR, the training task is translated into a binary classification problem where for each label a fit of the LR is performed.

Classifier performance metrics

In order to quantify the performance of the RNN and to compare it to other classifiers the metrics recall, precision and F1 (i.e. harmonic mean between precision and recall) were computed. For these three metrics the performance from all three classes are reported. These metrics are

Table 3. Cross validation summary. Summary of each fold of the 7-fold CV of the trajectory data. Shown are the number of targets (# TR), number of trajectories (# Trj), number of snapshots (# Snap.), percentage of snapshots with improved quality (I (%)), percentage of snapshots with no change in quality (N (%)), percentage of snapshots with decreased quality (D (%)).

Training Set						
Fold	**# TR**	**# Trj.**	**# Snap.**	**I (%)**	**N (%)**	**D (%)**
0	36	780	1,463,580	8.38	15.00	76.62
1	36	772	1,451,572	8.39	14.07	77.54
2	36	772	1,451,572	8.54	14.00	77.46
3	36	772	1,451,572	7.47	14.38	78.14
4	36	780	1,462,780	8.09	14.46	77.45
5	36	782	1,499,782	8.22	14.90	76.88
6	36	766	1,477,366	8.35	14.87	76.78
Validation Set						
Fold	**# TR**	**# Trj.**	**# Snap.**	**I (%)**	**N (%)**	**D (%)**
0	6	124	246,124	7.16	11.74	81.10
1	6	132	258,132	7.17	17.14	75.70
2	6	132	258,132	6.34	17.49	76.17
3	6	132	258,132	12.34	15.35	72.31
4	6	124	246,924	8.90	14.93	76.17
5	6	122	209,922	8.12	11.87	80.01
6	6	138	232,338	7.31	12.36	80.33

https://doi.org/10.1371/journal.pone.0202652.t003

defined as follows:

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{8}$$

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{9}$$

$$\text{F1} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \tag{10}$$

Where TP is the number of true positives, FN the number of false negatives and FP the number of false positives.

Structural model assessment metrics

The model quality of the snapshots is quantified by the two metrics GDTTS and Cα-RMSD.

RMSD. The root mean square deviation (RMSD) quantifies the disagreement of the predicted model to the reference structure. A lower value indicates a better fit to the reference structure. The definition is such that

$$\text{RMSD}(\boldsymbol{v}, \boldsymbol{w}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \|v_i - w_i\|^2}, \tag{11}$$

where $\boldsymbol{v}$, $\boldsymbol{w}$ are the set of atom coordinates for the model and reference structure, respectively. An optimal superimposition of $\boldsymbol{v}$ to $\boldsymbol{w}$ is performed prior to RMSD calculation.

GDTTS. The global distance test total score (GDTTS) is a model quality metric that evaluates quality based on the percentage of residues under different distance cutoffs given by

$$\mathrm{GDTTS} = \frac{\mathrm{GDT}_{P_1} + \mathrm{GDT}_{P_2} + \mathrm{GDT}_{P_4} + \mathrm{GDT}_{P_8}}{4}, \quad (12)$$

where P_n is the percentage of residues below distance cutoff n in Å with respect to a reference structure. A higher value indicates a better fit to the reference structure.

Sampling procedure

The targets and starting models for the refinement simulations are given by the CASP committee and originate from rounds 11 and 12. These structures represent a diverse set of proteins with different folds and initial model quality ranging from 30 to 90 GDTTS.

The trajectories used for training and testing the RNN originate from 5 different MD-based sampling procedures. These are known as (i) Unrestrained sampling (no_rst), no position or distance restraints to the residues of the starting are applied; (ii) position restraint sampling (point_rst), position restraints to the Cα-atom of structurally conserved residues are applied; (iii) distance restraint sampling (dist_rst), residue-residue distance restraints are applied to residues that are structurally conserved; (iv) metadynamic sampling with exclusive residue-residue contacts (cm_excl), sampling with position restraints and in contact map space (CMS) with metadynamics of unique residue-residue contacts; (v) metadynamic sampling with minimum distance residue-residue contact (cm_min), sampling with position restraints and in CMS of residue contact pairs of the minimum initial distance.

Position and distance restraint generation and sampling. The generation of restraints is outlined in S2A Fig. For every CASP refinement target models from all participating predictors were downloaded. The number of models varies from target to target, but on average 180 submissions were available. Often, a substantial part of these submissions were physically implausible, i.e. they contained long extended stretches. In order to avoid including these into the analysis a 10 Å Cα-RMSD cutoff to the provided starting model was applied.

From this filtered set, position and distance restraints for structurally conserved regions were generated and applied to Cα-atoms. Position restraints were applied if the per-residue Cα-RMSF calculated from the filtered set is < 3 Å (see S2C Fig for an example). In order to determine conserved residue-residue distances all possible combinations of Cα-Cα pairs were measured and distance restraints were applied if all of the following criteria are true: a) the Cα-Cα pairs are at least 5 residues apart; b) the Cα-Cα distance is below 9 Å; c) the standard deviation of the distance is below 1 Å (see S2D Fig for an example).

For each target three different simulation setups are executed: a) 3 ns long MD run without restraints, replicated 8 times; b) 3 ns long MD run with position restraints, replicated 8 times; c) 3 ns long MD run with distance restraints, replicated 8 times (see S2B Fig). All MD simulations were computed with GROMACS, using version 4.6 [56], and the G54a7 force field [57]. For all initial target structures hydrogen atoms were added and the systems were neutralized with Na^+ and Cl^- counter ions. A cubic simulation system with a 12 Å buffer between the edge of the box and the protein was solvated with TIP3P water molecules [58]. All targets were then subject to an energy minimisation using the steepest decent algorithm with a maximum of 50000 steps. This was followed by an equilibrium phase to relax the structure and its solvent. MD simulations (a) and (b) were subject to a 2 step equilibrium protocol where all heavy atoms were position restrained by a force of 1000 kJ $mol^{-1}nm^{-1}$ throughout the equilibration. In the first phase an NVT equilibration of the system was performed to increase the temperature from 0 K to 300 K in 100 ps using V-rescale [59] for temperature coupling. The second

phase consisted of a 300 ps long NPT equilibration of the system's pressure to 1 bar using Parrinello Rahman pressure coupling [60]. For MD simulation (c) a second NPT equilibration was applied, where the first step consisted of a 200 ps long equilibration with full heavy atoms position restraints and distance restraints, and the second step of a 200 ps equilibration with distance restraints only.

For all simulation setups a leap-frog integrator with a Δt of 2 fs was used and coordinates, velocities, energies, and forces were saved every 2 ps. Long range electrostatic interactions were treated with the Particle Mesh Ewald method [61] with a cutoff of 10 Å. Temperature and pressure coupling were controlled by the V-rescale and the Parrinello-Rahman method and were set to 300 K and 1 bar, respectively.

Contact map generation and sampling. All available models from participating predictors of a target are downloaded from the prediction center server. Each model is compared to the starting model and the Cα-RMSD is calculated, models with a Cα-RMSD > 10 Å are removed from the set. This was done in order to remove outliers from the set.

The filtered set is used to determine structurally conserved residues. These are identified by computing the per residue root mean square fluctuation (RMSF) of Cα atoms. Residues with a RMSF < 3 Å are considered conserved and movements are restraint during the sampling process.

From the structures in the filtered set of CASP predictions, residue-residue contacts are identified with a Cα or Cβ distance below 8 Å (S2E Fig), with the exception of direct neighbours, which are removed from the list. From these contacts two contact maps (CM) are generated, namely CM_{exl} and CM_{min} (S2F Fig). CM_{exl} contains contacts that are exclusive to one model from the filtered set, whereas the map CM_{min} contains contacts with the lowest Cα/Cβ distance. From these CMs we can define two CVs describing the CMS:

$$CV1(R) = 1/N \sum_{\gamma \in CM_{exl}} (D_\gamma(R) - D_\gamma(R_{ref}))^2 \tag{13}$$

$$CV2(R) = 1/N \sum_{\gamma \in CM_{min}} (D_\gamma(R) - D_\gamma(R_{ref}))^2 \tag{14}$$

$$D_\gamma(R) = \frac{1 - (r_\gamma / r_\gamma^0)^n}{1 - (r_\gamma / r_\gamma^0)^m} \tag{15}$$

The sigmoid distance function $D_\gamma(R)$ is used to quantify the formation of a contact γ in structure R, where r_γ is the contact distance in structure R and r_γ^0 is the contact distance in reference structure R_{ref} which denotes to one of the models from the filtered set of CASP12 models where the contact was observed. Variables n and m are constant and set to $n = 6$ and $m = 10$.

The preparation of the starting model prior to the sampling process follows a GROMACS standard procedure where the system is solvated, energy minimized and equilibrated for 300 ps. The sampling with metadynamics in CMS is performed at 300 K for 10 ns with 5 replicas for each CM definition, resulting in 100 ns sampling data for each target. The sampling of the CMS was performed with the GROMACS plug-in PLUMED2 [62] where a Gaussian addition is deposited every 2 ps with $\sigma = 0.5$, a bias factor of 10 and an initial height of 5 kJ/mol.

Supporting information

S1 Fig. Extended DeepTrajectory method figure. (A) Schematic overview of the RNN with GRU cells. (B) GRU cell, (C) & (D) Visualisation of the trajectory data v and the process of

mini-batch creation and propagation to the RNN.
(PDF)

S2 Fig. Sampling protocol. (A) Flowchart outlining the generation of the restraints and contact maps. (B) Flowchart outlining the different MD sampling procedures (C) Example of point restraints applied to a protein. (D) Example of residue-residue distance restraints applied to a protein. (E) Definition of residue-residue contact (F) Contact map definition for CM_{exl} and CM_{min}.
(PDF)

S3 Fig. Comparison of DeepTrajectory classification performance with and without distance metric features. Classification performance is shown for all three classes improved, no-change and decreased. Models with and without features RMSD_SM and GDTTS_SM are denoted as w/ dist. and w/o dist in the legend, respectively. The colored bars in purple (precision), red (recall) and green (F1) show the mean validation performance and the error bars the standard deviation of the 7-fold cross-validation. The black horizontal lines show the significance with p-value < 0.05 (*) and non-significance with p-value ≥ 0.05 (ns) between two groups as computed by the Wilcoxon rank-sum test.
(PDF)

S4 Fig. Prediction performance analysis for the no-change state predictions. (A) Cumulative distribution of true positive (TP), false negative (FN) and false positive (FP) predictions for the no-change state as a function of ΔGDTTS. The background colours red, gray and green indicate the ΔGDTTS regions for the improved, no-change and decreased states, respectively. (B) Cumulative distribution of true positive (TP), false negative (FN) and false positive (FP) predictions for the no-change state as a function of GDTTS. (C) Show the distribution of assigned probabilities for no-change state predictions.
(PDF)

S5 Fig. Prediction performance analysis for the decreased state predictions. (A) Cumulative distribution of true positive (TP), false negative (FN) and false positive (FP) predictions for the decreased state as a function of ΔGDTTS. The background colours red, gray and green indicate the ΔGDTTS regions for the improved, no-change and decreased states, respectively. (B) Cumulative distribution of true positive (TP), false negative (FN) and false positive (FP) predictions for the decreased state as a function of GDTTS. (C) Show the distribution of assigned probabilities for decreased state predictions.
(PDF)

S6 Fig. Learning curves for randomized trajectory snapshots. The trajectory snapshots of the training-set have been randomly shuffled and the prediction performance on validation trajectories is shown. A notable effect on performance can be observed for improved state class, showing instable learning behavior. Classification precision (purple), recall (red) and F1 (green) as a function of training steps for the three different states (A) improved, (B) no-change and (C) decreased are visualized for CV fold 4. The vertical lines indicate the best performance for each metric based on a moving average with a window-size of 30.
(PDF)

S1 Table. Protein target overview. Table contains the CASP identifier (starting with T0), the CASP refinement identifier (starting with TR), the PDB id of the reference crystal structure and a short description of the protein.
(PDF)

S2 Table. Trajectory and cross validation overview for all protein targets. Each row shows the trajectory name, the cross-validation fold assignment, the number of snapshots per trajectory, the total number of sampling runs and the absolute number of sampled improved, no-change and decreased states.
(PDF)

Acknowledgments

We would like to thank Robert Jenkins, Esther Wershof, David Jones and Cen Wan for the useful comments and discussions. This work was supported by the Francis Crick Institute which receives its core funding from Cancer Research UK (FC001003), the UK Medical Research Council (FC001003), and the Wellcome Trust (FC001003).

Author Contributions

Conceptualization: Erik Pfeiffenberger, Paul A. Bates.

Data curation: Erik Pfeiffenberger.

Formal analysis: Erik Pfeiffenberger.

Funding acquisition: Paul A. Bates.

Investigation: Erik Pfeiffenberger.

Methodology: Erik Pfeiffenberger.

Project administration: Paul A. Bates.

Resources: Erik Pfeiffenberger.

Software: Erik Pfeiffenberger.

Supervision: Paul A. Bates.

Validation: Erik Pfeiffenberger.

Visualization: Erik Pfeiffenberger.

Writing – original draft: Erik Pfeiffenberger.

Writing – review & editing: Erik Pfeiffenberger, Paul A. Bates.

References

1. Baker D, Sali A. Protein structure prediction and structural genomics. Science. 2001; 294(5540):93–96. https://doi.org/10.1126/science.1065659 PMID: 11588250
2. Moult J, Fidelis K, Kryshtafovych A, Schwede T, Tramontano A. Critical assessment of methods of protein structure prediction: Progress and new directions in round XI. Proteins: Structure, Function and Bioinformatics. 2016; 84(S1):4–14. https://doi.org/10.1002/prot.25064
3. Huang YJ, Mao B, Aramini JM, Montelione GT. Assessment of template-based protein structure predictions in CASP10. Proteins: Structure, Function and Bioinformatics. 2014; 82(SUPPL.2):43–56. https://doi.org/10.1002/prot.24488
4. Kelly LA, Mezulis S, Yates C, Wass M, Sternberg M. The Phyre2 web portal for protein modelling, prediction, and analysis. Nature Protocols. 2015; 10(6):845–858. https://doi.org/10.1038/nprot.2015.053
5. Moult J, Fidelis K, Kryshtafovych A, Schwede T, Tramontano A. Critical assessment of methods of protein structure prediction (CASP)—round X. Proteins: Structure, Function and Bioinformatics. 2014; 82 (SUPPL.2):1–6. https://doi.org/10.1002/prot.24452
6. Mariani V, Kiefer F, Schmidt T, Haas J, Schwede T. Assessment of template based protein structure predictions in CASP9. Proteins: Structure, Function and Bioinformatics. 2011; 79(SUPPL. 10):37–58. https://doi.org/10.1002/prot.23177

7. Kryshtafovych A, Venclovas Č, Fidelis K, Moult J. Progress over the first decade of CASP experiments. Proteins: Structure, Function and Genetics. 2005; 61(SUPPL. 7):225–236. https://doi.org/10.1002/prot.20740
8. Moult J. A decade of CASP: Progress, bottlenecks and prognosis in protein structure prediction. Current Opinion in Structural Biology. 2005; 15(3 SPEC. ISS.):285–289. https://doi.org/10.1016/j.sbi.2005.05.011 PMID: 15939584
9. Mirjalili V, Noyes K, Feig M. Physics-based protein structure refinement through multiple molecular dynamics trajectories and structure averaging. Proteins: Structure, Function and Bioinformatics. 2014; 82(SUPPL.2):196–207. https://doi.org/10.1002/prot.24336
10. Feig M. Local Protein Structure Refinement via Molecular Dynamics Simulations with locPREFMD. Journal of Chemical Information and Modeling. 2016; 56(7):1304–1312. https://doi.org/10.1021/acs.jcim.6b00222 PMID: 27380201
11. Hovan L, Oleinikovas V, Yalinca H, Kryshtafovych A, Saladino G, Gervasio FL. Assessment of the model refinement category in CASP12. Proteins: Structure, Function, and Bioinformatics. 2017; 86 (S1):152–167.
12. Park H, Ovchinnikov S, Kim DE, DiMaio F, Baker D. Protein homology model refinement by large-scale energy optimization. Proceedings of the National Academy of Sciences. 2018; p. 201719115.
13. Feig M. Computational protein structure refinement: Almost there, yet still so far to go. Wiley Interdisciplinary Reviews: Computational Molecular Science. 2017; 7(3):e1307.
14. Modi V, Dunbrack RL. Assessment of refinement of template-based models in CASP11. Proteins: Structure, Function, and Bioinformatics. 2016; 84(S1):260–281. https://doi.org/10.1002/prot.25048
15. Nugent T, Cozzetto D, Jones DT. Evaluation of predictions in the CASP10 model refinement category. Proteins: Structure, Function and Bioinformatics. 2014; 82(SUPPL.2):98–111. https://doi.org/10.1002/prot.24377
16. Tran NH, Zhang X, Xin L, Shan B, Li M. De novo peptide sequencing by deep learning. Proceedings of the National Academy of Sciences. 2017; p. 201705691.
17. Wu Y, Schuster M, Chen Z, Le QV, Norouzi M, Macherey W, et al. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. ArXiv e-prints. 2016; p. 1–23.
18. Lee B, Baek J, Yoon S. deepTarget: End-to-end Learning Framework for microRNA Target Prediction using Deep Recurrent Neural Networks. ArXiv e-prints. 2016
19. Schmidhuber J. Deep Learning in Neural Networks: An Overview. Neural Networks. 2015; 61:85–117. https://doi.org/10.1016/j.neunet.2014.09.003 PMID: 25462637
20. Cho K, van Merrienboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, et al. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. CoRR. 2014;abs/1406.1.
21. Cozzetto D, Kryshtafovych A, Fidelis K, Moult J, Rost B, Tramontano A. Evaluation of template-based models in CASP8 with standard measures. Proteins: Structure, Function and Bioinformatics. 2009; 77 (SUPPL. 9):18–28. https://doi.org/10.1002/prot.22561
22. Zemla A, Venclovas Č, Moult J, Fidelis K. Processing and evaluation of predictions in CASP4. Proteins: Structure, Function and Genetics. 2001; 45(SUPPL. 5):13–21. https://doi.org/10.1002/prot.10052
23. Breiman L. Random forests. Machine learning. 2001; 45(1):5–32. https://doi.org/10.1023/A:1010933404324
24. Cunningham P, Delany SJ. K -Nearest Neighbour Classifiers. Multiple Classifier Systems. 2007; p. 1–17.
25. Bishop C. Pattern recognition and machine learning. vol. 4. Springer; 2006.
26. Manavalan B, Lee J. SVMQA: support–vector-machine-based protein single-model quality assessment. Bioinformatics. 2017; 33(16):2496–2503. https://doi.org/10.1093/bioinformatics/btx222 PMID: 28419290
27. Jones DT, Singh T, Kosciolek T, Tetchner S. MetaPSICOV: Combining coevolution methods for accurate prediction of contacts and long range hydrogen bonding in proteins. Bioinformatics. 2015; 31 (7):999–1006. https://doi.org/10.1093/bioinformatics/btu791 PMID: 25431331
28. Agius R, Torchala M, Moal IH, Fernández-Recio J, Bates PA. Characterizing changes in the rate of protein-protein dissociation upon interface mutation using hotspot energy and organization. PLoS Comput Biol. 2013; 9(9):e1003216. https://doi.org/10.1371/journal.pcbi.1003216 PMID: 24039569
29. Maaten LVD, Hinton G. Visualizing Data using t-SNE. Journal of Machine Learning Research 1. 2008; 620(1):267–84.

30. Pfeiffenberger E, Chaleil RAG, Moal IH, Bates PA. A machine learning approach for ranking clusters of docked protein-protein complexes by pairwise cluster comparison. Proteins: Structure, Function and Bioinformatics. 2017; 85(3):528–543. https://doi.org/10.1002/prot.25218

31. Liao J, Chin K. Logistic regression for disease classification using microarray data: model selection in a large p and small n case. Bioinformatics. 2007; 23(15):1945–1951. https://doi.org/10.1093/bioinformatics/btm287 PMID: 17540680

32. Li L, Umbach DM, Terry P, Taylor JA. Application of the GA/KNN method to SELDI proteomics data. Bioinformatics. 2004; 20(10):1638–1640. https://doi.org/10.1093/bioinformatics/bth098 PMID: 14962943

33. Fa R, Cozzetto D, Wan C, Jones DT. Predicting human protein function with multi-task deep neural networks. PLOS ONE. 2018; 13(6):e0198216. https://doi.org/10.1371/journal.pone.0198216 PMID: 29889900

34. Hamp T, Rost B. More challenges for machine-learning protein interactions. Bioinformatics. 2015; 31 (10):1521–1525. https://doi.org/10.1093/bioinformatics/btu857 PMID: 25586513

35. He H, Garcia EA. Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering. 2009; 21(9):1263–1284. https://doi.org/10.1109/TKDE.2008.239

36. Lindorff-Larsen K, Piana S, Dror RO, Shaw DE. How fast-folding proteins fold. Science. 2011; 334 (6055):517–520. https://doi.org/10.1126/science.1208351 PMID: 22034434

37. Chodera JD, Noé F. Markov state models of biomolecular conformational dynamics. Current Opinion in Structural Biology. 2014; 25:135–144. https://doi.org/10.1016/j.sbi.2014.04.002 PMID: 24836551

38. Weber JK, Pande VS. Characterization and rapid sampling of protein folding Markov state model topologies. Journal of Chemical Theory and Computation. 2011; 7(10):3405–3411. https://doi.org/10.1021/ct2004484 PMID: 22140370

39. Zacharias M. Accounting for conformational changes during protein-protein docking. Current Opinion in Structural Biology. 2010; 20(2):180–186. https://doi.org/10.1016/j.sbi.2010.02.001 PMID: 20194014

40. Król M, Tournier AL, Bates PA. Flexible relaxation of rigid-body docking solutions. Proteins: Structure, Function and Genetics. 2007; 68(1):159–169. https://doi.org/10.1002/prot.21391

41. Moal IH, Jiménez-García B, Fernández-Recio J. CCharPPI web server: computational characterization of protein–protein interactions from structure. Bioinformatics. 2015; 31(1):123–125. https://doi.org/10.1093/bioinformatics/btu594 PMID: 25183488

42. Vangone A, Bonvin AMJJ. Contacts-based prediction of binding affinity in protein–protein complexes. eLife. 2015; 4(JULY2015). https://doi.org/10.7554/eLife.07454 PMID: 26193119

43. Schmidt T, Bergner A, Schwede T. Modelling three-dimensional protein structures for applications in drug design. Drug Discovery Today. 2014; 19(7):890–897. https://doi.org/10.1016/j.drudis.2013.10.027 PMID: 24216321

44. Brunk E, Mih N, Monk J, Zhang Z, O'Brien EJ, Bliven SE, et al. Systems biology of the structural proteome. BMC Systems Biology. 2016; 10(1). https://doi.org/10.1186/s12918-016-0271-6 PMID: 26969117

45. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. CoRR. 2016;abs/1603.0.

46. Kingma DP, Ba J. Adam: A Method for Stochastic Optimization. CoRR. 2014;abs/1412.6.

47. Liu S, Zhang C, Zhou H, Zhou Y. A physical reference state unifies the structure-derived potential of mean force for protein folding and binding. Proteins: Structure, Function, and Bioinformatics. 2004; 56 (1):93–101. https://doi.org/10.1002/prot.20019

48. Shen MY, Sali A. Statistical potential for assessment and prediction of protein structures. Protein Science. 2006; 15(11):2507–2524. https://doi.org/10.1110/ps.062416606 PMID: 17075131

49. Chae MH, Krull F, Knapp EW. Optimized distance-dependent atom-pair-based potential DOOP for protein structure prediction. Proteins: Structure, Function and Bioinformatics. 2015; 83(5):881–890. https://doi.org/10.1002/prot.24782

50. Zhang J, Zhang Y. A novel side-chain orientation dependent potential derived from random-walk reference state for protein fold selection and structure prediction. PLoS ONE. 2010; 5(10):e15386. https://doi.org/10.1371/journal.pone.0015386 PMID: 21060880

51. Zhou H, Skolnick J. GOAP: a generalized orientation-dependent, all-atom statistical potential for protein structure prediction. Biophysical journal. 2011; 101(8):2043–52. https://doi.org/10.1016/j.bpj.2011.09.012 PMID: 22004759

52. Eswar N, Eramian D, Webb B, Shen MY, Sali A. Protein Structure Modeling with MODELLER. In: Protein Structure Prediction; 2008. p. 145–159.

53. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research. 2011; 12(Oct):2825–2830.

54. Liu T, Moore AW, Gray A. New Algorithms for Efficient High-Dimensional Nonparametric Classification. Journal of Machine Learning Research. 2006; 7:1135–1158.

55. Bentley JL. Multidimensional binary search trees used for associative searching. Communications of the ACM. 1975; 18(9):509–517. https://doi.org/10.1145/361002.361007

56. Hess B, Kutzner C, Van Der Spoel D, Lindahl E. GRGMACS 4: Algorithms for highly efficient, load-balanced, and scalable molecular simulation. Journal of Chemical Theory and Computation. 2008; 4 (3):435–447. https://doi.org/10.1021/ct700301q PMID: 26620784

57. Schmid N, Eichenberger AP, Choutko A, Riniker S, Winger M, Mark AE, et al. Definition and testing of the GROMOS force-field versions 54A7 and 54B7. European Biophysics Journal. 2011; 40(7):843–856. https://doi.org/10.1007/s00249-011-0700-9 PMID: 21533652

58. Jorgensen WL, Chandrasekhar J, Madura JD, Impey RW, Klein ML. Comparison of simple potential functions for simulating liquid water. The Journal of Chemical Physics. 1983; 79(2):926–935. https://doi.org/10.1063/1.445869

59. Bussi G, Donadio D, Parrinello M. Canonical sampling through velocity rescaling. Journal of Chemical Physics. 2007; 126(1). https://doi.org/10.1063/1.2408420 PMID: 17212484

60. Berendsen HJC, Postma JPM, van Gunsteren WF, DiNola A, Haak JR. Molecular dynamics with coupling to an external bath. The Journal of Chemical Physics. 1984; 81(8):3684–3690. https://doi.org/10.1063/1.448118

61. Darden T, York D, Pedersen L. Particle mesh Ewald: An N log(N) method for Ewald sums in large systems. The Journal of Chemical Physics. 1993; 98(12):10089–10092. https://doi.org/10.1063/1.464397

62. Tribello GA, Bonomi M, Branduardi D, Camilloni C, Bussi G. PLUMED 2: New feathers for an old bird. Computer Physics Communications. 2014; 185(2):604–613. https://doi.org/10.1016/j.cpc.2013.09.018