Article

Highly accurate protein structure prediction with AlphaFold

https://doi.org/10.1038/s41586-021-03819-2

Received: 11 May 2021

Accepted: 12 July 2021

Published online: 15 July 2021

Open access

Check for updates

John Jumper[1,4]✉, Richard Evans[1,4], Alexander Pritzel[1,4], Tim Green[1,4], Michael Figurnov[1,4], Olaf Ronneberger[1,4], Kathryn Tunyasuvunakool[1,4], Russ Bates[1,4], Augustin Žídek[1,4], Anna Potapenko[1,4], Alex Bridgland[1,4], Clemens Meyer[1,4], Simon A. A. Kohl[1,4], Andrew J. Ballard[1,4], Andrew Cowie[1,4], Bernardino Romera-Paredes[1,4], Stanislav Nikolov[1,4], Rishub Jain[1,4], Jonas Adler[1], Trevor Back[1], Stig Petersen[1], David Reiman[1], Ellen Clancy[1], Michal Zielinski[1], Martin Steinegger[2,3], Michalina Pacholska[1], Tamas Berghammer[1], Sebastian Bodenstein[1], David Silver[1], Oriol Vinyals[1], Andrew W. Senior[1], Koray Kavukcuoglu[1], Pushmeet Kohli[1] & Demis Hassabis[1,4]✉

Proteins are essential to life, and understanding their structure can facilitate a mechanistic understanding of their function. Through an enormous experimental effort[1–4], the structures of around 100,000 unique proteins have been determined[5], but this represents a small fraction of the billions of known protein sequences[6,7]. Structural coverage is bottlenecked by the months to years of painstaking effort required to determine a single protein structure. Accurate computational approaches are needed to address this gap and to enable large-scale structural bioinformatics. Predicting the three-dimensional structure that a protein will adopt based solely on its amino acid sequence—the structure prediction component of the 'protein folding problem'[8]—has been an important open research problem for more than 50 years[9]. Despite recent progress[10–14], existing methods fall far short of atomic accuracy, especially when no homologous structure is available. Here we provide the first computational method that can regularly predict protein structures with atomic accuracy even in cases in which no similar structure is known. We validated an entirely redesigned version of our neural network-based model, AlphaFold, in the challenging 14th Critical Assessment of protein Structure Prediction (CASP14)[15], demonstrating accuracy competitive with experimental structures in a majority of cases and greatly outperforming other methods. Underpinning the latest version of AlphaFold is a novel machine learning approach that incorporates physical and biological knowledge about protein structure, leveraging multi-sequence alignments, into the design of the deep learning algorithm.

The development of computational methods to predict three-dimensional (3D) protein structures from the protein sequence has proceeded along two complementary paths that focus on either the physical interactions or the evolutionary history. The physical interaction programme heavily integrates our understanding of molecular driving forces into either thermodynamic or kinetic simulation of protein physics[16] or statistical approximations thereof[17]. Although theoretically very appealing, this approach has proved highly challenging for even moderate-sized proteins due to the computational intractability of molecular simulation, the context dependence of protein stability and the difficulty of producing sufficiently accurate models of protein physics. The evolutionary programme has provided an alternative in recent years, in which the constraints on protein structure are derived from bioinformatics analysis of the evolutionary history of proteins, homology to solved structures[18,19] and pairwise evolutionary correlations[20–24]. This bioinformatics approach has benefited greatly from the steady growth of experimental protein structures deposited in the Protein Data Bank (PDB)[5], the explosion of genomic sequencing and the rapid development of deep learning techniques to interpret these correlations. Despite these advances, contemporary physical and evolutionary-history-based approaches produce predictions that are far short of experimental accuracy in the majority of cases in which a close homologue has not been solved experimentally and this has limited their utility for many biological applications.

In this study, we develop the first, to our knowledge, computational approach capable of predicting protein structures to near experimental accuracy in a majority of cases. The neural network AlphaFold that we developed was entered into the CASP14 assessment (May–July 2020; entered under the team name 'AlphaFold2' and a completely different model from our CASP13 AlphaFold system[10]). The CASP assessment is carried out biennially using recently solved structures that have not been deposited in the PDB or publicly disclosed so that it is a blind test

[1]DeepMind, London, UK. [2]School of Biological Sciences, Seoul National University, Seoul, South Korea. [3]Artificial Intelligence Institute, Seoul National University, Seoul, South Korea. [4]These authors contributed equally: John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Demis Hassabis.
✉e-mail: jumper@deepmind.com; dhcontact@deepmind.com

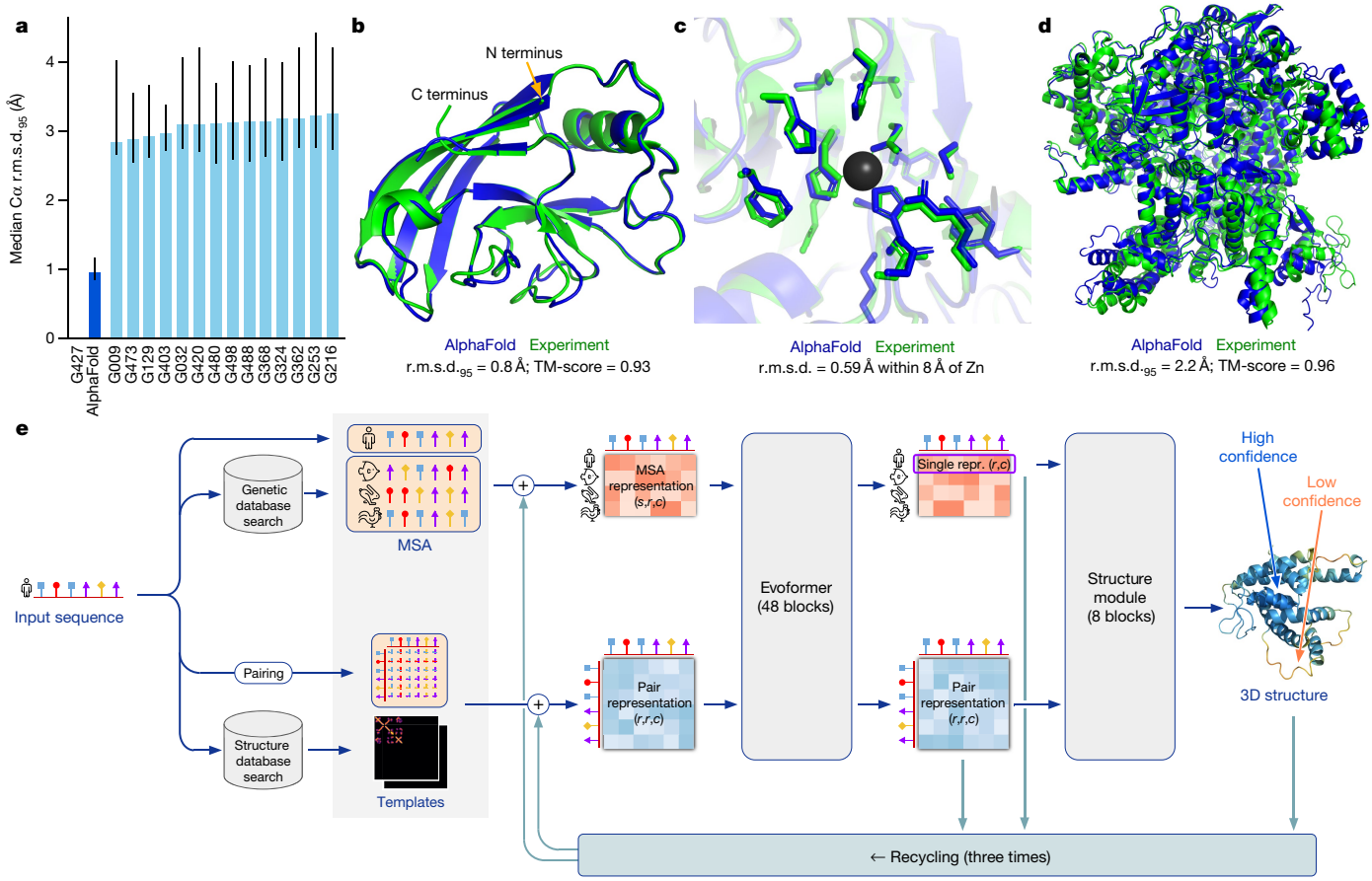


Fig. 1 | AlphaFold produces highly accurate structures. a, The performance of AlphaFold on the CASP14 dataset ($n = 87$ protein domains) relative to the top-15 entries (out of 146 entries), group numbers correspond to the numbers assigned to entrants by CASP. Data are median and the 95% confidence interval of the median, estimated from 10,000 bootstrap samples. **b**, Our prediction of CASP14 target T1049 (PDB 6Y4F, blue) compared with the true (experimental) structure (green). Four residues in the C terminus of the crystal structure are *B*-factor outliers and are not depicted. **c**, CASP14 target T1056 (PDB 6YJ1). An example of a well-predicted zinc-binding site (AlphaFold has accurate side chains even though it does not explicitly predict the zinc ion). **d**, CASP target T1044 (PDB 6VR4)—a 2,180-residue single chain—was predicted with correct domain packing (the prediction was made after CASP using AlphaFold without intervention). **e**, Model architecture. Arrows show the information flow among the various components described in this paper. Array shapes are shown in parentheses with *s*, number of sequences (N_{seq} in the main text); *r*, number of residues (N_{res} in the main text); *c*, number of channels.

for the participating methods, and has long served as the gold-standard assessment for the accuracy of structure prediction[25,26].

In CASP14, AlphaFold structures were vastly more accurate than competing methods. AlphaFold structures had a median backbone accuracy of 0.96 Å r.m.s.d.$_{95}$ (Cα root-mean-square deviation at 95% residue coverage) (95% confidence interval = 0.85–1.16 Å) whereas the next best performing method had a median backbone accuracy of 2.8 Å r.m.s.d.$_{95}$ (95% confidence interval = 2.7–4.0 Å) (measured on CASP domains; see Fig. 1a for backbone accuracy and Supplementary Fig. 14 for all-atom accuracy). As a comparison point for this accuracy, the width of a carbon atom is approximately 1.4 Å. In addition to very accurate domain structures (Fig. 1b), AlphaFold is able to produce highly accurate side chains (Fig. 1c) when the backbone is highly accurate and considerably improves over template-based methods even when strong templates are available. The all-atom accuracy of AlphaFold was 1.5 Å r.m.s.d.$_{95}$ (95% confidence interval = 1.2–1.6 Å) compared with the 3.5 Å r.m.s.d.$_{95}$ (95% confidence interval = 3.1–4.2 Å) of the best alternative method. Our methods are scalable to very long proteins with accurate domains and domain-packing (see Fig. 1d for the prediction of a 2,180-residue protein with no structural homologues). Finally, the model is able to provide precise, per-residue estimates of its reliability that should enable the confident use of these predictions.

We demonstrate in Fig. 2a that the high accuracy that AlphaFold demonstrated in CASP14 extends to a large sample of recently released PDB structures; in this dataset, all structures were deposited in the PDB after our training data cut-off and are analysed as full chains (see Methods, Supplementary Fig. 15 and Supplementary Table 6 for more details). Furthermore, we observe high side-chain accuracy when the backbone prediction is accurate (Fig. 2b) and we show that our confidence measure, the predicted local-distance difference test (pLDDT), reliably predicts the Cα local-distance difference test (lDDT-Cα) accuracy of the corresponding prediction (Fig. 2c). We also find that the global superposition metric template modelling score (TM-score)[27] can be accurately estimated (Fig. 2d). Overall, these analyses validate that the high accuracy and reliability of AlphaFold on CASP14 proteins also transfers to an uncurated collection of recent PDB submissions, as would be expected (see Supplementary Methods 1.15 and Supplementary Fig. 11 for confirmation that this high accuracy extends to new folds).

The AlphaFold network

AlphaFold greatly improves the accuracy of structure prediction by incorporating novel neural network architectures and training procedures based on the evolutionary, physical and geometric constraints of protein structures. In particular, we demonstrate a new architecture to jointly embed multiple sequence alignments (MSAs) and pairwise features, a new output representation and associated loss that enable accurate end-to-end structure prediction, a new equivariant attention

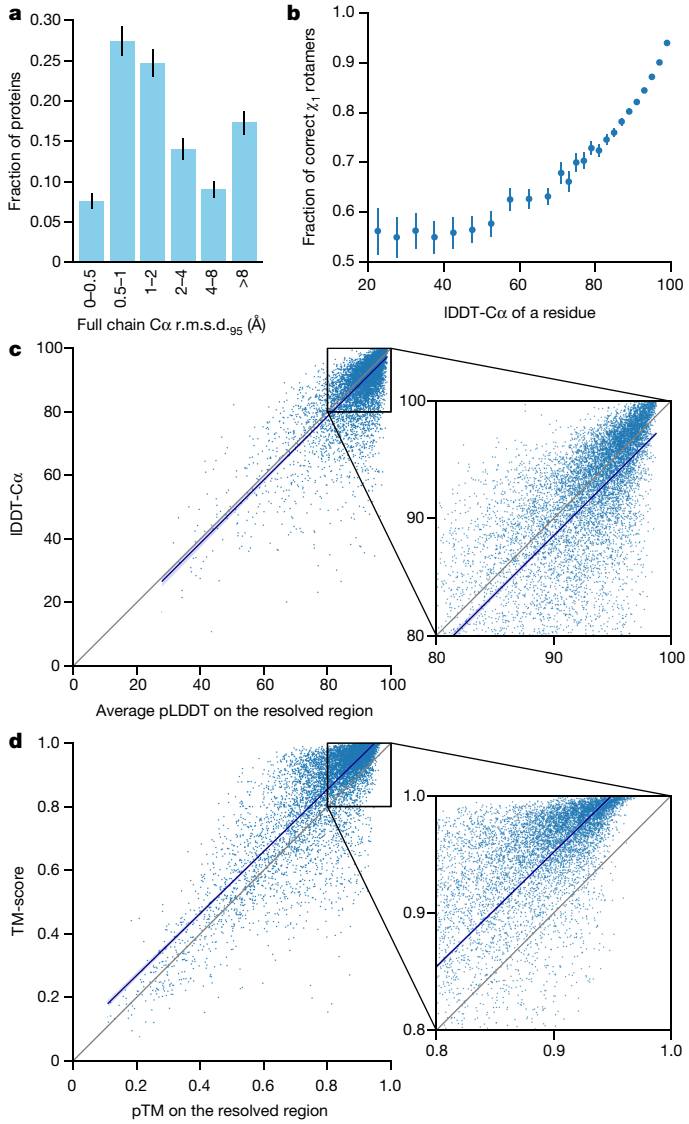


Fig. 2 | Accuracy of AlphaFold on recent PDB structures. The analysed structures are newer than any structure in the training set. Further filtering is applied to reduce redundancy (see Methods). **a**, Histogram of backbone r.m.s.d. for full chains (Cα r.m.s.d. at 95% coverage). Error bars are 95% confidence intervals (Poisson). This dataset excludes proteins with a template (identified by hmmsearch) from the training set with more than 40% sequence identity covering more than 1% of the chain (n = 3,144 protein chains). The overall median is 1.46 Å (95% confidence interval = 1.40–1.56 Å). Note that this measure will be highly sensitive to domain packing and domain accuracy; a high r.m.s.d. is expected for some chains with uncertain packing or packing errors. **b**, Correlation between backbone accuracy and side-chain accuracy. Filtered to structures with any observed side chains and resolution better than 2.5 Å (n = 5,317 protein chains); side chains were further filtered to B-factor <30 Å^2. A rotamer is classified as correct if the predicted torsion angle is within 40°. Each point aggregates a range of lDDT-Cα, with a bin size of 2 units above 70 lDDT-Cα and 5 units otherwise. Points correspond to the mean accuracy; error bars are 95% confidence intervals (Student t-test) of the mean on a per-residue basis. **c**, Confidence score compared to the true accuracy on chains. Least-squares linear fit lDDT-Cα = 0.997 × pLDDT − 1.17 (Pearson's r = 0.76). n = 10,795 protein chains. The shaded region of the linear fit represents a 95% confidence interval estimated from 10,000 bootstrap samples. In the companion paper[39], additional quantification of the reliability of pLDDT as a confidence measure is provided. **d**, Correlation between pTM and full chain TM-score. Least-squares linear fit TM-score = 0.98 × pTM + 0.07 (Pearson's r = 0.85). n = 10,795 protein chains. The shaded region of the linear fit represents a 95% confidence interval estimated from 10,000 bootstrap samples.

architecture, use of intermediate losses to achieve iterative refinement of predictions, masked MSA loss to jointly train with the structure, learning from unlabelled protein sequences using self-distillation and self-estimates of accuracy.

The AlphaFold network directly predicts the 3D coordinates of all heavy atoms for a given protein using the primary amino acid sequence and aligned sequences of homologues as inputs (Fig. 1e; see Methods for details of inputs including databases, MSA construction and use of templates). A description of the most important ideas and components is provided below. The full network architecture and training procedure are provided in the Supplementary Methods.

The network comprises two main stages. First, the trunk of the network processes the inputs through repeated layers of a novel neural network block that we term Evoformer to produce an $N_{seq} \times N_{res}$ array (N_{seq}, number of sequences; N_{res}, number of residues) that represents a processed MSA and an $N_{res} \times N_{res}$ array that represents residue pairs. The MSA representation is initialized with the raw MSA (although see Supplementary Methods 1.2.7 for details of handling very deep MSAs). The Evoformer blocks contain a number of attention-based and non-attention-based components. We show evidence in 'Interpreting the neural network' that a concrete structural hypothesis arises early within the Evoformer blocks and is continuously refined. The key innovations in the Evoformer block are new mechanisms to exchange information within the MSA and pair representations that enable direct reasoning about the spatial and evolutionary relationships.

The trunk of the network is followed by the structure module that introduces an explicit 3D structure in the form of a rotation and translation for each residue of the protein (global rigid body frames). These representations are initialized in a trivial state with all rotations set to the identity and all positions set to the origin, but rapidly develop and refine a highly accurate protein structure with precise atomic details. Key innovations in this section of the network include breaking the chain structure to allow simultaneous local refinement of all parts of the structure, a novel equivariant transformer to allow the network to implicitly reason about the unrepresented side-chain atoms and a loss term that places substantial weight on the orientational correctness of the residues. Both within the structure module and throughout the whole network, we reinforce the notion of iterative refinement by repeatedly applying the final loss to outputs and then feeding the outputs recursively into the same modules. The iterative refinement using the whole network (which we term 'recycling' and is related to approaches in computer vision[28,29]) contributes markedly to accuracy with minor extra training time (see Supplementary Methods 1.8 for details).

Evoformer

The key principle of the building block of the network—named Evoformer (Figs. 1e, 3a)—is to view the prediction of protein structures as a graph inference problem in 3D space in which the edges of the graph are defined by residues in proximity. The elements of the pair representation encode information about the relation between the residues (Fig. 3b). The columns of the MSA representation encode the individual residues of the input sequence while the rows represent the sequences in which those residues appear. Within this framework, we define a number of update operations that are applied in each block in which the different update operations are applied in series.

The MSA representation updates the pair representation through an element-wise outer product that is summed over the MSA sequence dimension. In contrast to previous work[30], this operation is applied within every block rather than once in the network, which enables the continuous communication from the evolving MSA representation to the pair representation.

Within the pair representation, there are two different update patterns. Both are inspired by the necessity of consistency of the pair

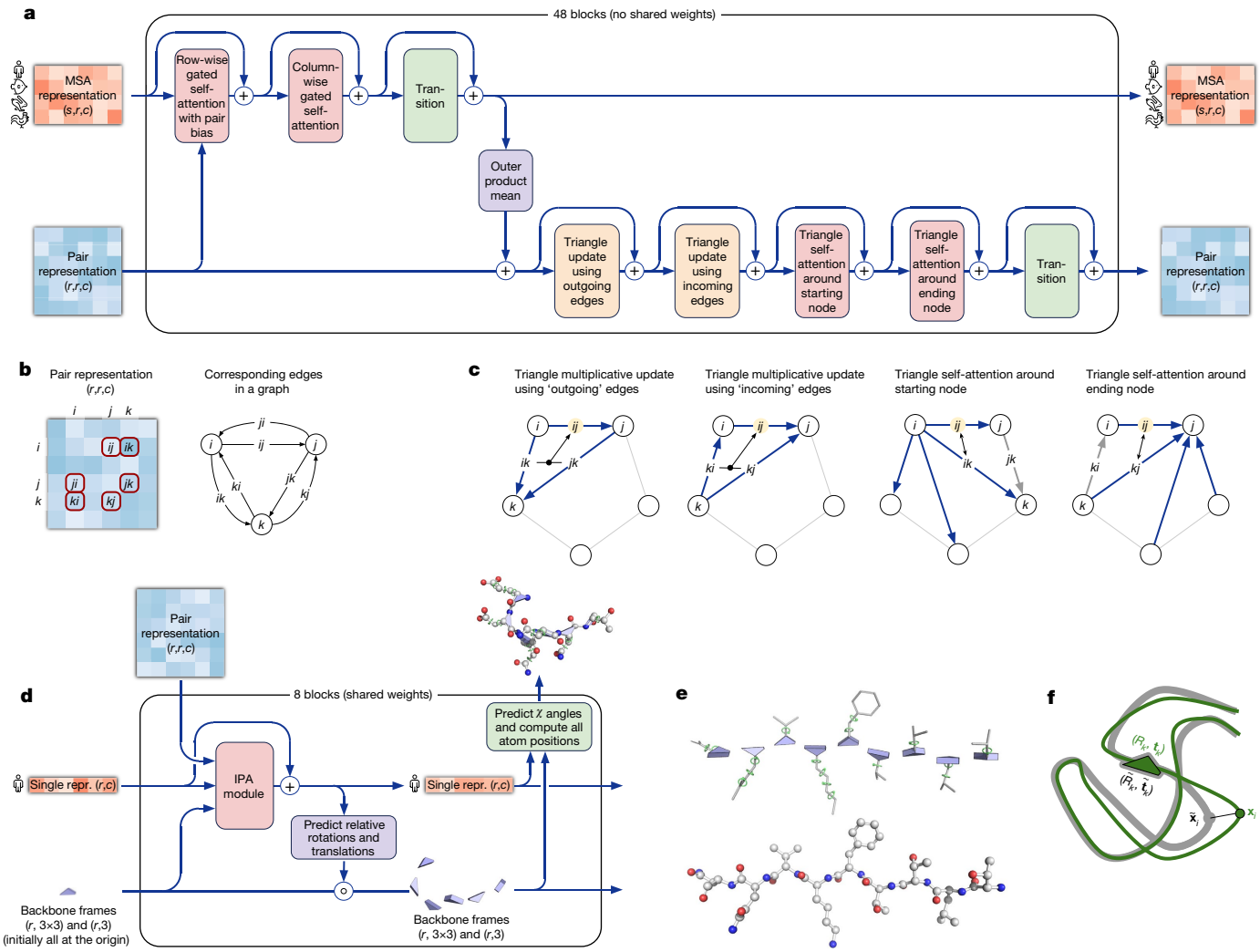


Fig. 3 | Architectural details. a, Evoformer block. Arrows show the information flow. The shape of the arrays is shown in parentheses. **b**, The pair representation interpreted as directed edges in a graph. **c**, Triangle multiplicative update and triangle self-attention. The circles represent residues. Entries in the pair representation are illustrated as directed edges and in each diagram, the edge being updated is *ij*. **d**, Structure module including Invariant point attention (IPA) module. The single representation is a copy of the first row of the MSA representation. **e**, Residue gas: a representation of each residue as one free-floating rigid body for the backbone (blue triangles) and χ angles for the side chains (green circles). The corresponding atomic structure is shown below. **f**, Frame aligned point error (FAPE). Green, predicted structure; grey, true structure; $(R_k, \mathbf{t}_k)$, frames; $\mathbf{x}_i$, atom positions.

representation—for a pairwise description of amino acids to be representable as a single 3D structure, many constraints must be satisfied including the triangle inequality on distances. On the basis of this intuition, we arrange the update operations on the pair representation in terms of triangles of edges involving three different nodes (Fig. 3c). In particular, we add an extra logit bias to axial attention[31] to include the 'missing edge' of the triangle and we define a non-attention update operation 'triangle multiplicative update' that uses two edges to update the missing third edge (see Supplementary Methods 1.6.5 for details). The triangle multiplicative update was developed originally as a more symmetric and cheaper replacement for the attention, and networks that use only the attention or multiplicative update are both able to produce high-accuracy structures. However, the combination of the two updates is more accurate.

We also use a variant of axial attention within the MSA representation. During the per-sequence attention in the MSA, we project additional logits from the pair stack to bias the MSA attention. This closes the loop by providing information flow from the pair representation back into the MSA representation, ensuring that the overall Evoformer block is able to fully mix information between the pair and MSA representations and prepare for structure generation within the structure module.

End-to-end structure prediction

The structure module (Fig. 3d) operates on a concrete 3D backbone structure using the pair representation and the original sequence row (single representation) of the MSA representation from the trunk. The 3D backbone structure is represented as N_{res} independent rotations and translations, each with respect to the global frame (residue gas) (Fig. 3e). These rotations and translations—representing the geometry of the N-Cα-C atoms—prioritize the orientation of the protein backbone so that the location of the side chain of each residue is highly constrained within that frame. Conversely, the peptide bond geometry is completely unconstrained and the network is observed to frequently violate the chain constraint during the application of the structure module as breaking this constraint enables the local refinement of all parts of the chain without solving complex loop closure problems. Satisfaction of the peptide bond geometry is encouraged during fine-tuning by a violation loss term. Exact enforcement of peptide bond geometry is only achieved in the post-prediction relaxation of the structure by gradient descent in the Amber[32] force field. Empirically, this final relaxation does not improve the accuracy of the model as measured by the

global distance test (GDT)[33] or lDDT-Cα[34] but does remove distracting stereochemical violations without the loss of accuracy.

The residue gas representation is updated iteratively in two stages (Fig. 3d). First, a geometry-aware attention operation that we term 'invariant point attention' (IPA) is used to update an N_{res} set of neural activations (single representation) without changing the 3D positions, then an equivariant update operation is performed on the residue gas using the updated activations. The IPA augments each of the usual attention queries, keys and values with 3D points that are produced in the local frame of each residue such that the final value is invariant to global rotations and translations (see Methods 'IPA' for details). The 3D queries and keys also impose a strong spatial/locality bias on the attention, which is well-suited to the iterative refinement of the protein structure. After each attention operation and element-wise transition block, the module computes an update to the rotation and translation of each backbone frame. The application of these updates within the local frame of each residue makes the overall attention and update block an equivariant operation on the residue gas.

Predictions of side-chain χ angles as well as the final, per-residue accuracy of the structure (pLDDT) are computed with small per-residue networks on the final activations at the end of the network. The estimate of the TM-score (pTM) is obtained from a pairwise error prediction that is computed as a linear projection from the final pair representation. The final loss (which we term the frame-aligned point error (FAPE) (Fig. 3f)) compares the predicted atom positions to the true positions under many different alignments. For each alignment, defined by aligning the predicted frame $(R_k, \mathbf{t}_k)$ to the corresponding true frame, we compute the distance of all predicted atom positions $\mathbf{x}_i$ from the true atom positions. The resulting $N_{\text{frames}} \times N_{\text{atoms}}$ distances are penalized with a clamped L^1 loss. This creates a strong bias for atoms to be correct relative to the local frame of each residue and hence correct with respect to its side-chain interactions, as well as providing the main source of chirality for AlphaFold (Supplementary Methods 1.9.3 and Supplementary Fig. 9).

Training with labelled and unlabelled data

The AlphaFold architecture is able to train to high accuracy using only supervised learning on PDB data, but we are able to enhance accuracy (Fig. 4a) using an approach similar to noisy student self-distillation[35]. In this procedure, we use a trained network to predict the structure of around 350,000 diverse sequences from Uniclust30[36] and make a new dataset of predicted structures filtered to a high-confidence subset. We then train the same architecture again from scratch using a mixture of PDB data and this new dataset of predicted structures as the training data, in which the various training data augmentations such as cropping and MSA subsampling make it challenging for the network to recapitulate the previously predicted structures. This self-distillation procedure makes effective use of the unlabelled sequence data and considerably improves the accuracy of the resulting network.

Additionally, we randomly mask out or mutate individual residues within the MSA and have a Bidirectional Encoder Representations from Transformers (BERT)-style[37] objective to predict the masked elements of the MSA sequences. This objective encourages the network to learn to interpret phylogenetic and covariation relationships without hardcoding a particular correlation statistic into the features. The BERT objective is trained jointly with the normal PDB structure loss on the same training examples and is not pre-trained, in contrast to recent independent work[38].

Interpreting the neural network

To understand how AlphaFold predicts protein structure, we trained a separate structure module for each of the 48 Evoformer blocks in the network while keeping all parameters of the main network frozen (Supplementary Methods 1.14). Including our recycling stages, this provides a trajectory of 192 intermediate structures—one per full

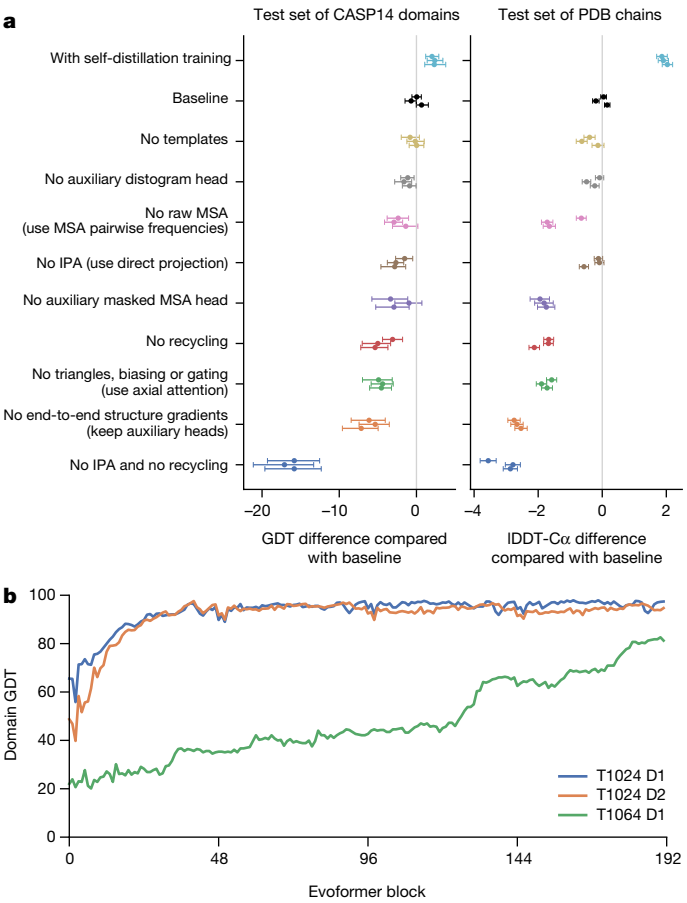


Fig. 4 | Interpreting the neural network. a, Ablation results on two target sets: the CASP14 set of domains ($n = 87$ protein domains) and the PDB test set of chains with template coverage of ≤30% at 30% identity ($n = 2,261$ protein chains). Domains are scored with GDT and chains are scored with lDDT-Cα. The ablations are reported as a difference compared with the average of the three baseline seeds. Means (points) and 95% bootstrap percentile intervals (error bars) are computed using bootstrap estimates of 10,000 samples. **b**, Domain GDT trajectory over 4 recycling iterations and 48 Evoformer blocks on CASP14 targets LmrP (T1024) and Orf8 (T1064) where D1 and D2 refer to the individual domains as defined by the CASP assessment. Both T1024 domains obtain the correct structure early in the network, whereas the structure of T1064 changes multiple times and requires nearly the full depth of the network to reach the final structure. Note, 48 Evoformer blocks comprise one recycling iteration.

Evoformer block—in which each intermediate represents the belief of the network of the most likely structure at that block. The resulting trajectories are surprisingly smooth after the first few blocks, showing that AlphaFold makes constant incremental improvements to the structure until it can no longer improve (see Fig. 4b for a trajectory of accuracy). These trajectories also illustrate the role of network depth. For very challenging proteins such as ORF8 of SARS-CoV-2 (T1064), the network searches and rearranges secondary structure elements for many layers before settling on a good structure. For other proteins such as LmrP (T1024), the network finds the final structure within the first few layers. Structure trajectories of CASP14 targets T1024, T1044, T1064 and T1091 that demonstrate a clear iterative building process for a range of protein sizes and difficulties are shown in Supplementary Videos 1–4. In Supplementary Methods 1.16 and Supplementary Figs. 12, 13, we interpret the attention maps produced by AlphaFold layers.

Figure 4a contains detailed ablations of the components of AlphaFold that demonstrate that a variety of different mechanisms contribute to AlphaFold accuracy. Detailed descriptions of each ablation model, their training details, extended discussion of ablation results and the

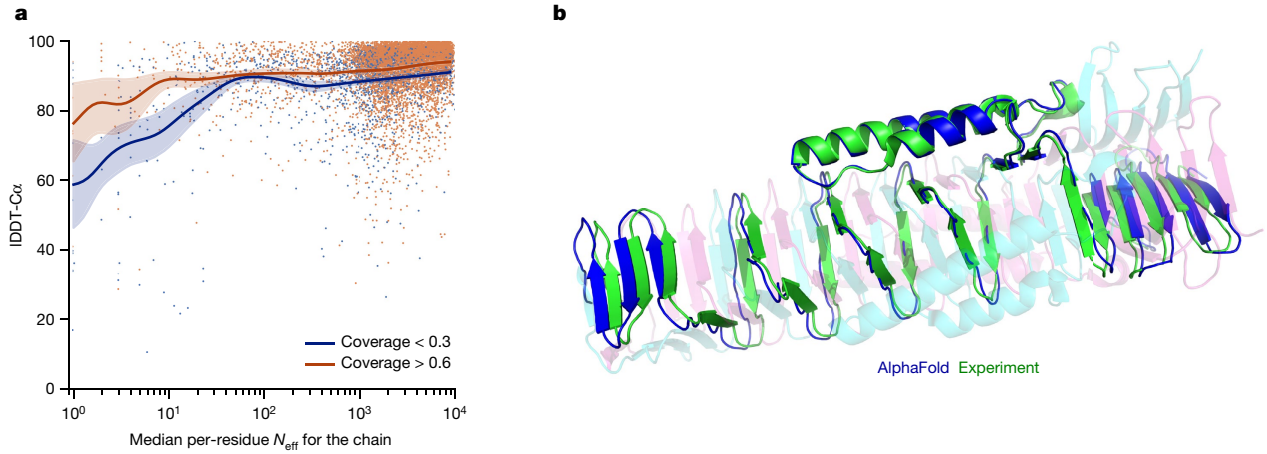


Fig. 5 | Effect of MSA depth and cross-chain contacts. a, Backbone accuracy (lDDT-Cα) for the redundancy-reduced set of the PDB after our training data cut-off, restricting to proteins in which at most 25% of the long-range contacts are between different heteromer chains. We further consider two groups of proteins based on template coverage at 30% sequence identity: covering more than 60% of the chain (n = 6,743 protein chains) and covering less than 30% of the chain (n = 1,596 protein chains). MSA depth is computed by counting the number of non-gap residues for each position in the MSA (using the N_{eff} weighting scheme; see Methods for details) and taking the median across residues. The curves are obtained through Gaussian kernel average smoothing (window size is 0.2 units in $\log_{10}(N_{eff})$); the shaded area is the 95% confidence interval estimated using bootstrap of 10,000 samples. **b**, An intertwined homotrimer (PDB 6SK0) is correctly predicted without input stoichiometry and only a weak template (blue is predicted and green is experimental).

effect of MSA depth on each ablation are provided in Supplementary Methods 1.13 and Supplementary Fig. 10.

MSA depth and cross-chain contacts

Although AlphaFold has a high accuracy across the vast majority of deposited PDB structures, we note that there are still factors that affect accuracy or limit the applicability of the model. The model uses MSAs and the accuracy decreases substantially when the median alignment depth is less than around 30 sequences (see Fig. 5a for details). We observe a threshold effect where improvements in MSA depth over around 100 sequences lead to small gains. We hypothesize that the MSA information is needed to coarsely find the correct structure within the early stages of the network, but refinement of that prediction into a high-accuracy model does not depend crucially on the MSA information. The other substantial limitation that we have observed is that AlphaFold is much weaker for proteins that have few intra-chain or homotypic contacts compared to the number of heterotypic contacts (further details are provided in a companion paper[39]). This typically occurs for bridging domains within larger complexes in which the shape of the protein is created almost entirely by interactions with other chains in the complex. Conversely, AlphaFold is often able to give high-accuracy predictions for homomers, even when the chains are substantially intertwined (Fig. 5b). We expect that the ideas of AlphaFold are readily applicable to predicting full hetero-complexes in a future system and that this will remove the difficulty with protein chains that have a large number of hetero-contacts.

Related work

The prediction of protein structures has had a long and varied development, which is extensively covered in a number of reviews[14,40–43]. Despite the long history of applying neural networks to structure prediction[14,42,43], they have only recently come to improve structure prediction[10,11,44,45]. These approaches effectively leverage the rapid improvement in computer vision systems[46] by treating the problem of protein structure prediction as converting an 'image' of evolutionary couplings[22–24] to an 'image' of the protein distance matrix and then integrating the distance predictions into a heuristic system that produces the final 3D coordinate prediction. A few recent studies have been developed to predict the 3D coordinates directly[47–50], but the accuracy of these approaches does not match traditional, hand-crafted structure prediction pipelines[51]. In parallel, the success of attention-based networks for language processing[52] and, more recently, computer vision[31,53] has inspired the exploration of attention-based methods for interpreting protein sequences[54–56].

Discussion

The methodology that we have taken in designing AlphaFold is a combination of the bioinformatics and physical approaches: we use a physical and geometric inductive bias to build components that learn from PDB data with minimal imposition of handcrafted features (for example, AlphaFold builds hydrogen bonds effectively without a hydrogen bond score function). This results in a network that learns far more efficiently from the limited data in the PDB but is able to cope with the complexity and variety of structural data.

In particular, AlphaFold is able to handle missing the physical context and produce accurate models in challenging cases such as intertwined homomers or proteins that only fold in the presence of an unknown haem group. The ability to handle underspecified structural conditions is essential to learning from PDB structures as the PDB represents the full range of conditions in which structures have been solved. In general, AlphaFold is trained to produce the protein structure most likely to appear as part of a PDB structure. For example, in cases in which a particular stochiometry, ligand or ion is predictable from the sequence alone, AlphaFold is likely to produce a structure that respects those constraints implicitly.

AlphaFold has already demonstrated its utility to the experimental community, both for molecular replacement[57] and for interpreting cryogenic electron microscopy maps[58]. Moreover, because AlphaFold outputs protein coordinates directly, AlphaFold produces predictions in graphics processing unit (GPU) minutes to GPU hours depending on the length of the protein sequence (for example, around one GPU minute per model for 384 residues; see Methods for details). This opens up the exciting possibility of predicting structures at the proteome-scale and beyond—in a companion paper[39], we demonstrate the application of AlphaFold to the entire human proteome[39].

The explosion in available genomic sequencing techniques and data has revolutionized bioinformatics but the intrinsic challenge of experimental structure determination has prevented a similar expansion in our structural knowledge. By developing an accurate protein structure

prediction algorithm, coupled with existing large and well-curated structure and sequence databases assembled by the experimental community, we hope to accelerate the advancement of structural bioinformatics that can keep pace with the genomics revolution. We hope that AlphaFold—and computational approaches that apply its techniques for other biophysical problems—will become essential tools of modern biology.

Online content

Any methods, additional references, Nature Research reporting summaries, source data, extended data, supplementary information, acknowledgements, peer review information; details of author contributions and competing interests; and statements of data and code availability are available at https://doi.org/10.1038/s41586-021-03819-2.

1. Thompson, M. C., Yeates, T. O. & Rodriguez, J. A. Advances in methods for atomic resolution macromolecular structure determination. *F1000Res.* **9**, 667 (2020).
2. Bai, X.-C., McMullan, G. & Scheres, S. H. W. How cryo-EM is revolutionizing structural biology. *Trends Biochem. Sci.* **40**, 49–57 (2015).
3. Jaskolski, M., Dauter, Z. & Wlodawer, A. A brief history of macromolecular crystallography, illustrated by a family tree and its Nobel fruits. *FEBS J.* **281**, 3985–4009 (2014).
4. Wüthrich, K. The way to NMR structures of proteins. *Nat. Struct. Biol.* **8**, 923–925 (2001).
5. wwPDB Consortium. Protein Data Bank: the single global archive for 3D macromolecular structure data. *Nucleic Acids Res.* **47**, D520–D528 (2018).
6. Mitchell, A. L. et al. MGnify: the microbiome analysis resource in 2020. *Nucleic Acids Res.* **48**, D570–D578 (2020).
7. Steinegger, M., Mirdita, M. & Söding, J. Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold. *Nat. Methods* **16**, 603–606 (2019).
8. Dill, K. A., Ozkan, S. B., Shell, M. S. & Weikl, T. R. The protein folding problem. *Annu. Rev. Biophys.* **37**, 289–316 (2008).
9. Anfinsen, C. B. Principles that govern the folding of protein chains. *Science* **181**, 223–230 (1973).
10. Senior, A. W. et al. Improved protein structure prediction using potentials from deep learning. *Nature* **577**, 706–710 (2020).
11. Wang, S., Sun, S., Li, Z., Zhang, R. & Xu, J. Accurate de novo prediction of protein contact map by ultra-deep learning model. *PLOS Comput. Biol.* **13**, e1005324 (2017).
12. Zheng, W. et al. Deep-learning contact-map guided protein structure prediction in CASP13. *Proteins* **87**, 1149–1164 (2019).
13. Abriata, L. A., Tamò, G. E. & Dal Peraro, M. A further leap of improvement in tertiary structure prediction in CASP13 prompts new routes for future assessments. *Proteins* **87**, 1100–1112 (2019).
14. Pearce, R. & Zhang, Y. Deep learning techniques have significantly impacted protein structure prediction and protein design. *Curr. Opin. Struct. Biol.* **68**, 194–207 (2021).
15. Moult, J., Fidelis, K., Kryshtafovych, A., Schwede, T. & Topf, M. Critical assessment of techniques for protein structure prediction, fourteenth round. *CASP 14 Abstract Book* https://www.predictioncenter.org/casp14/doc/CASP14_Abstracts.pdf (2020).
16. Brini, E., Simmerling, C. & Dill, K. Protein storytelling through physics. *Science* **370**, eaaz3041 (2020).
17. Sippl, M. J. Calculation of conformational ensembles from potentials of mean force. An approach to the knowledge-based prediction of local structures in globular proteins. *J. Mol. Biol.* **213**, 859–883 (1990).
18. Šali, A. & Blundell, T. L. Comparative protein modelling by satisfaction of spatial restraints. *J. Mol. Biol.* **234**, 779–815 (1993).
19. Roy, A., Kucukural, A. & Zhang, Y. I-TASSER: a unified platform for automated protein structure and function prediction. *Nat. Protocols* **5**, 725–738 (2010).
20. Altschuh, D., Lesk, A. M., Bloomer, A. C. & Klug, A. Correlation of co-ordinated amino acid substitutions with function in viruses related to tobacco mosaic virus. *J. Mol. Biol.* **193**, 693–707 (1987).
21. Shindyalov, I. N., Kolchanov, N. A. & Sander, C. Can three-dimensional contacts in protein structures be predicted by analysis of correlated mutations? *Protein Eng.* **7**, 349–358 (1994).
22. Weigt, M., White, R. A., Szurmant, H., Hoch, J. A. & Hwa, T. Identification of direct residue contacts in protein–protein interaction by message passing. *Proc. Natl Acad. Sci. USA* **106**, 67–72 (2009).
23. Marks, D. S. et al. Protein 3D structure computed from evolutionary sequence variation. *PLoS ONE* **6**, e28766 (2011).
24. Jones, D. T., Buchan, D. W. A., Cozzetto, D. & Pontil, M. PSICOV: precise structural contact prediction using sparse inverse covariance estimation on large multiple sequence alignments. *Bioinformatics* **28**, 184–190 (2012).
25. Moult, J., Pedersen, J. T., Judson, R. & Fidelis, K. A large-scale experiment to assess protein structure prediction methods. *Proteins* **23**, ii–iv (1995).
26. Kryshtafovych, A., Schwede, T., Topf, M., Fidelis, K. & Moult, J. Critical assessment of methods of protein structure prediction (CASP)-round XIII. *Proteins* **87**, 1011–1020 (2019).
27. Zhang, Y. & Skolnick, J. Scoring function for automated assessment of protein structure template quality. *Proteins* **57**, 702–710 (2004).
28. Tu, Z. & Bai, X. Auto-context and its application to high-level vision tasks and 3D brain image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **32**, 1744–1757 (2010).
29. Carreira, J., Agrawal, P., Fragkiadaki, K. & Malik, J. Human pose estimation with iterative error feedback. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition* 4733–4742 (2016).
30. Mirabello, C. & Wallner, B. rawMSA: end-to-end deep learning using raw multiple sequence alignments. *PLoS ONE* **14**, e0220182 (2019).
31. Huang, Z. et al. CCNet: criss-cross attention for semantic segmentation. In *Proc. IEEE/CVF International Conference on Computer Vision* 603–612 (2019).
32. Hornak, V. et al. Comparison of multiple Amber force fields and development of improved protein backbone parameters. *Proteins* **65**, 712–725 (2006).
33. Zemla, A. LGA: a method for finding 3D similarities in protein structures. *Nucleic Acids Res.* **31**, 3370–3374 (2003).
34. Mariani, V., Biasini, M., Barbato, A. & Schwede, T. lDDT: a local superposition-free score for comparing protein structures and models using distance difference tests. *Bioinformatics* **29**, 2722–2728 (2013).
35. Xie, Q., Luong, M.-T., Hovy, E. & Le, Q. V. Self-training with noisy student improves imagenet classification. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition* 10687–10698 (2020).
36. Mirdita, M. et al. Uniclust databases of clustered and deeply annotated protein sequences and alignments. *Nucleic Acids Res.* **45**, D170–D176 (2017).
37. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proc. 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* 1, 4171–4186 (2019).
38. Rao, R. et al. MSA transformer. In *Proc. 38th International Conference on Machine Learning* PMLR 139, 8844–8856 (2021).
39. Tunyasuvunakool, K. et al. Highly accurate protein structure prediction for the human proteome. *Nature* https://doi.org/10.1038/s41586-021-03828-1 (2021).
40. Kuhlman, B. & Bradley, P. Advances in protein structure prediction and design. *Nat. Rev. Mol. Cell Biol.* **20**, 681–697 (2019).
41. Marks, D. S., Hopf, T. A. & Sander, C. Protein structure prediction from sequence variation. *Nat. Biotechnol.* **30**, 1072–1080 (2012).
42. Qian, N. & Sejnowski, T. J. Predicting the secondary structure of globular proteins using neural network models. *J. Mol. Biol.* **202**, 865–884 (1988).
43. Fariselli, P., Olmea, O., Valencia, A. & Casadio, R. Prediction of contact maps with neural networks and correlated mutations. *Protein Eng.* **14**, 835–843 (2001).
44. Yang, J. et al. Improved protein structure prediction using predicted interresidue orientations. *Proc. Natl Acad. Sci. USA* **117**, 1496–1503 (2020).
45. Li, Y. et al. Deducing high-accuracy protein contact-maps from a triplet of coevolutionary matrices through deep residual convolutional networks. *PLOS Comput. Biol.* **17**, e1008865 (2021).
46. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition* 770–778 (2016).
47. AlQuraishi, M. End-to-end differentiable learning of protein structure. *Cell Syst.* **8**, 292–301 (2019).
48. Senior, A. W. et al. Protein structure prediction using multiple deep neural networks in the 13th Critical Assessment of Protein Structure Prediction (CASP13). *Proteins* **87**, 1141–1148 (2019).
49. Ingraham, J., Riesselman, A. J., Sander, C. & Marks, D. S. Learning protein structure with a differentiable simulator. in *Proc. International Conference on Learning Representations* (2019).
50. Li, J. Universal transforming geometric network. Preprint at https://arxiv.org/abs/1908.00723 (2019).
51. Xu, J., McPartlon, M. & Li, J. Improved protein structure prediction by deep learning irrespective of co-evolution information. *Nat. Mach. Intell.* **3**, 601–609 (2021).
52. Vaswani, A. et al. Attention is all you need. In *Advances in Neural Information Processing Systems* 5998–6008 (2017).
53. Wang, H. et al. Axial-deeplab: stand-alone axial-attention for panoptic segmentation. in *European Conference on Computer Vision* 108–126 (Springer, 2020).
54. Alley, E. C., Khimulya, G., Biswas, S., AlQuraishi, M. & Church, G. M. Unified rational protein engineering with sequence-based deep representation learning. *Nat. Methods* **16**, 1315–1322 (2019).
55. Heinzinger, M. et al. Modeling aspects of the language of life through transfer-learning protein sequences. *BMC Bioinformatics* **20**, 723 (2019).
56. Rives, A. et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proc. Natl Acad. Sci. USA* **118**, e2016239118 (2021).
57. Pereira, J. et al. High-accuracy protein structure prediction in CASP14. *Proteins* https://doi.org/10.1002/prot.26171 (2021).
58. Gupta, M. et al. CryoEM and AI reveal a structure of SARS-CoV-2 Nsp2, a multifunctional protein involved in key host processes. Preprint at https://doi.org/10.1101/2021.05.10.443524 (2021).

Methods

Full algorithm details

Extensive explanations of the components and their motivations are available in Supplementary Methods 1.1–1.10, in addition, pseudocode is available in Supplementary Information Algorithms 1–32, network diagrams in Supplementary Figs. 1–8, input features in Supplementary Table 1 and additional details are provided in Supplementary Tables 2, 3. Training and inference details are provided in Supplementary Methods 1.11–1.12 and Supplementary Tables 4, 5.

IPA

The IPA module combines the pair representation, the single representation and the geometric representation to update the single representation (Supplementary Fig. 8). Each of these representations contributes affinities to the shared attention weights and then uses these weights to map its values to the output. The IPA operates in 3D space. Each residue produces query points, key points and value points in its local frame. These points are projected into the global frame using the backbone frame of the residue in which they interact with each other. The resulting points are then projected back into the local frame. The affinity computation in the 3D space uses squared distances and the coordinate transformations ensure the invariance of this module with respect to the global frame (see Supplementary Methods 1.8.2 'Invariant point attention (IPA)' for the algorithm, proof of invariance and a description of the full multi-head version). A related construction that uses classic geometric invariants to construct pairwise features in place of the learned 3D points has been applied to protein design[59].

In addition to the IPA, standard dot product attention is computed on the abstract single representation and a special attention on the pair representation. The pair representation augments both the logits and the values of the attention process, which is the primary way in which the pair representation controls the structure generation.

Inputs and data sources

Inputs to the network are the primary sequence, sequences from evolutionarily related proteins in the form of a MSA created by standard tools including jackhmmer[60] and HHBlits[61], and 3D atom coordinates of a small number of homologous structures (templates) where available. For both the MSA and templates, the search processes are tuned for high recall; spurious matches will probably appear in the raw MSA but this matches the training condition of the network.

One of the sequence databases used, Big Fantastic Database (BFD), was custom-made and released publicly (see 'Data availability') and was used by several CASP teams. BFD is one of the largest publicly available collections of protein families. It consists of 65,983,866 families represented as MSAs and hidden Markov models (HMMs) covering 2,204,359,010 protein sequences from reference databases, metagenomes and metatranscriptomes.

BFD was built in three steps. First, 2,423,213,294 protein sequences were collected from UniProt (Swiss-Prot&TrEMBL, 2017-11)[62], a soil reference protein catalogue and the marine eukaryotic reference catalogue[7], and clustered to 30% sequence identity, while enforcing a 90% alignment coverage of the shorter sequences using MMseqs2/Linclust[63]. This resulted in 345,159,030 clusters. For computational efficiency, we removed all clusters with less than three members, resulting in 61,083,719 clusters. Second, we added 166,510,624 representative protein sequences from Metaclust NR (2017-05; discarding all sequences shorter than 150 residues)[63] by aligning them against the cluster representatives using MMseqs2[64]. Sequences that fulfilled the sequence identity and coverage criteria were assigned to the best scoring cluster. The remaining 25,347,429 sequences that could not be assigned were clustered separately and added as new clusters, resulting in the final clustering. Third, for each of the clusters, we computed an MSA using FAMSA[65] and computed the HMMs following the Uniclust HH-suite database protocol[36].

The following versions of public datasets were used in this study. Our models were trained on a copy of the PDB[5] downloaded on 28 August 2019. For finding template structures at prediction time, we used a copy of the PDB downloaded on 14 May 2020, and the PDB70[66] clustering database downloaded on 13 May 2020. For MSA lookup at both training and prediction time, we used Uniref90[67] v.2020_01, BFD, Uniclust30[36] v.2018_08 and MGnify[6] v.2018_12. For sequence distillation, we used Uniclust30[36] v.2018_08 to construct a distillation structure dataset. Full details are provided in Supplementary Methods 1.2.

For MSA search on BFD + Uniclust30, and template search against PDB70, we used HHBlits[61] and HHSearch[66] from hh-suite v.3.0-beta.3 (version 14/07/2017). For MSA search on Uniref90 and clustered MGnify, we used jackhmmer from HMMER3[68]. For constrained relaxation of structures, we used OpenMM v.7.3.1[69] with the Amber99sb force field[32]. For neural network construction, running and other analyses, we used TensorFlow[70], Sonnet[71], NumPy[72], Python[73] and Colab[74].

To quantify the effect of the different sequence data sources, we re-ran the CASP14 proteins using the same models but varying how the MSA was constructed. Removing BFD reduced the mean accuracy by 0.4 GDT, removing Mgnify reduced the mean accuracy by 0.7 GDT, and removing both reduced the mean accuracy by 6.1 GDT. In each case, we found that most targets had very small changes in accuracy but a few outliers had very large (20+ GDT) differences. This is consistent with the results in Fig. 5a in which the depth of the MSA is relatively unimportant until it approaches a threshold value of around 30 sequences when the MSA size effects become quite large. We observe mostly overlapping effects between inclusion of BFD and Mgnify, but having at least one of these metagenomics databases is very important for target classes that are poorly represented in UniRef, and having both was necessary to achieve full CASP accuracy.

Training regimen

To train, we use structures from the PDB with a maximum release date of 30 April 2018. Chains are sampled in inverse proportion to cluster size of a 40% sequence identity clustering. We then randomly crop them to 256 residues and assemble into batches of size 128. We train the model on Tensor Processing Unit (TPU) v3 with a batch size of 1 per TPU core, hence the model uses 128 TPU v3 cores. The model is trained until convergence (around 10 million samples) and further fine-tuned using longer crops of 384 residues, larger MSA stack and reduced learning rate (see Supplementary Methods 1.11 for the exact configuration). The initial training stage takes approximately 1 week, and the fine-tuning stage takes approximately 4 additional days.

The network is supervised by the FAPE loss and a number of auxiliary losses. First, the final pair representation is linearly projected to a binned distance distribution (distogram) prediction, scored with a cross-entropy loss. Second, we use random masking on the input MSAs and require the network to reconstruct the masked regions from the output MSA representation using a BERT-like loss[37]. Third, the output single representations of the structure module are used to predict binned per-residue lDDT-Cα values. Finally, we use an auxiliary side-chain loss during training, and an auxiliary structure violation loss during fine-tuning. Detailed descriptions and weighting are provided in the Supplementary Information.

An initial model trained with the above objectives was used to make structure predictions for a Uniclust dataset of 355,993 sequences with the full MSAs. These predictions were then used to train a final model with identical hyperparameters, except for sampling examples 75% of the time from the Uniclust prediction set, with sub-sampled MSAs, and 25% of the time from the clustered PDB set.

We train five different models using different random seeds, some with templates and some without, to encourage diversity in the predictions (see Supplementary Table 5 and Supplementary Methods 1.12.1

for details). We also fine-tuned these models after CASP14 to add a pTM prediction objective (Supplementary Methods 1.9.7) and use the obtained models for Fig. 2d.

Inference regimen

We inference the five trained models and use the predicted confidence score to select the best model per target.

Using our CASP14 configuration for AlphaFold, the trunk of the network is run multiple times with different random choices for the MSA cluster centres (see Supplementary Methods 1.11.2 for details of the ensembling procedure). The full time to make a structure prediction varies considerably depending on the length of the protein. Representative timings for the neural network using a single model on V100 GPU are 4.8 min with 256 residues, 9.2 min with 384 residues and 18 h at 2,500 residues. These timings are measured using our open-source code, and the open-source code is notably faster than the version we ran in CASP14 as we now use the XLA compiler[75].

Since CASP14, we have found that the accuracy of the network without ensembling is very close or equal to the accuracy with ensembling and we turn off ensembling for most inference. Without ensembling, the network is 8× faster and the representative timings for a single model are 0.6 min with 256 residues, 1.1 min with 384 residues and 2.1 h with 2,500 residues.

Inferencing large proteins can easily exceed the memory of a single GPU. For a V100 with 16 GB of memory, we can predict the structure of proteins up to around 1,300 residues without ensembling and the 256- and 384-residue inference times are using the memory of a single GPU. The memory usage is approximately quadratic in the number of residues, so a 2,500-residue protein involves using unified memory so that we can greatly exceed the memory of a single V100. In our cloud setup, a single V100 is used for computation on a 2,500-residue protein but we requested four GPUs to have sufficient memory.

Searching genetic sequence databases to prepare inputs and final relaxation of the structures take additional central processing unit (CPU) time but do not require a GPU or TPU.

Metrics

The predicted structure is compared to the true structure from the PDB in terms of lDDT metric[34], as this metric reports the domain accuracy without requiring a domain segmentation of chain structures. The distances are either computed between all heavy atoms (lDDT) or only the Cα atoms to measure the backbone accuracy (lDDT-Cα). As lDDT-Cα only focuses on the Cα atoms, it does not include the penalty for structural violations and clashes. Domain accuracies in CASP are reported as GDT[33] and the TM-score[27] is used as a full chain global superposition metric.

We also report accuracies using the r.m.s.d._{95} (Cα r.m.s.d._{95} at 95% coverage). We perform five iterations of (1) a least-squares alignment of the predicted structure and the PDB structure on the currently chosen Cα atoms (using all Cα atoms in the first iteration); (2) selecting the 95% of Cα atoms with the lowest alignment error. The r.m.s.d. of the atoms chosen for the final iterations is the r.m.s.d._{95}. This metric is more robust to apparent errors that can originate from crystal structure artefacts, although in some cases the removed 5% of residues will contain genuine modelling errors.

Test set of recent PDB sequences

For evaluation on recent PDB sequences (Figs. 2a–d, 4a, 5a), we used a copy of the PDB downloaded 15 February 2021. Structures were filtered to those with a release date after 30 April 2018 (the date limit for inclusion in the training set for AlphaFold). Chains were further filtered to remove sequences that consisted of a single amino acid as well as sequences with an ambiguous chemical component at any residue position. Exact duplicates were removed, with the chain with the most resolved Cα atoms used as the representative sequence. Subsequently, structures with less than 16 resolved residues, with unknown residues or solved by NMR methods were removed. As the PDB contains many near-duplicate sequences, the chain with the highest resolution was selected from each cluster in the PDB 40% sequence clustering of the data. Furthermore, we removed all sequences for which fewer than 80 amino acids had the alpha carbon resolved and removed chains with more than 1,400 residues. The final dataset contained 10,795 protein sequences.

The procedure for filtering the recent PDB dataset based on prior template identity was as follows. Hmmsearch was run with default parameters against a copy of the PDB SEQRES fasta downloaded 15 February 2021. Template hits were accepted if the associated structure had a release date earlier than 30 April 2018. Each residue position in a query sequence was assigned the maximum identity of any template hit covering that position. Filtering then proceeded as described in the individual figure legends, based on a combination of maximum identity and sequence coverage.

The MSA depth analysis was based on computing the normalized number of effective sequences (N_{eff}) for each position of a query sequence. Per-residue N_{eff} values were obtained by counting the number of non-gap residues in the MSA for this position and weighting the sequences using the N_{eff} scheme[76] with a threshold of 80% sequence identity measured on the region that is non-gap in either sequence.

Reporting summary

Further information on research design is available in the Nature Research Reporting Summary linked to this paper.

Data availability

All input data are freely available from public sources.

Structures from the PDB were used for training and as templates (https://www.wwpdb.org/ftp/pdb-ftp-sites; for the associated sequence data and 40% sequence clustering see also https://ftp.wwpdb.org/pub/pdb/derived_data/ and https://cdn.rcsb.org/resources/sequence/clusters/bc-40.out). Training used a version of the PDB downloaded 28 August 2019, while the CASP14 template search used a version downloaded 14 May 2020. The template search also used the PDB70 database, downloaded 13 May 2020 (https://wwwuser.gwdg.de/~compbiol/data/hhsuite/databases/hhsuite_dbs/).

We show experimental structures from the PDB with accession numbers 6Y4F[77], 6YJ1[78], 6VR4[79], 6SK0[80], 6FES[81], 6W6W[82], 6T1Z[83] and 7JTL[84].

For MSA lookup at both the training and prediction time, we used UniRef90 v.2020_01 (https://ftp.ebi.ac.uk/pub/databases/uniprot/previous_releases/release-2020_01/uniref/), BFD (https://bfd.mmseqs.com), Uniclust30 v.2018_08 (https://wwwuser.gwdg.de/~compbiol/uniclust/2018_08/) and MGnify clusters v.2018_12 (https://ftp.ebi.ac.uk/pub/databases/metagenomics/peptide_database/2018_12/). Uniclust30 v.2018_08 was also used as input for constructing a distillation structure dataset.

Code availability

Source code for the AlphaFold model, trained weights and inference script are available under an open-source license at https://github.com/deepmind/alphafold.

Neural networks were developed with TensorFlow v.1 (https://github.com/tensorflow/tensorflow), Sonnet v.1 (https://github.com/deepmind/sonnet), JAX v.0.1.69 (https://github.com/google/jax/) and Haiku v.0.0.4 (https://github.com/deepmind/dm-haiku). The XLA compiler is bundled with JAX and does not have a separate version number.

For MSA search on BFD+Uniclust30, and for template search against PDB70, we used HHBlits and HHSearch from hh-suite v.3.0-beta.3 release 14/07/2017 (https://github.com/soedinglab/hh-suite). For MSA search on UniRef90 and clustered MGnify, we used jackhmmer from

HMMER v.3.3 (http://eddylab.org/software/hmmer/). For constrained relaxation of structures, we used OpenMM v.7.3.1 (https://github.com/openmm/openmm) with the Amber99sb force field.

Construction of BFD used MMseqs2 v.925AF (https://github.com/soedinglab/MMseqs2) and FAMSA v.1.2.5 (https://github.com/refresh-bio/FAMSA).

Data analysis used Python v.3.6 (https://www.python.org/), NumPy v.1.16.4 (https://github.com/numpy/numpy), SciPy v.1.2.1 (https://www.scipy.org/), seaborn v.0.11.1 (https://github.com/mwaskom/seaborn), Matplotlib v.3.3.4 (https://github.com/matplotlib/matplotlib), bokeh v.1.4.0 (https://github.com/bokeh/bokeh), pandas v.1.1.5 (https://github.com/pandas-dev/pandas), plotnine v.0.8.0 (https://github.com/has2k1/plotnine), statsmodels v.0.12.2 (https://github.com/statsmodels/statsmodels) and Colab (https://research.google.com/colaboratory). TM-align v.20190822 (https://zhanglab.dcmb.med.umich.edu/TM-align/) was used for computing TM-scores. Structure visualizations were created in Pymol v.2.3.0 (https://github.com/schrodinger/pymol-open-source).

59. Ingraham, J., Garg, V. K., Barzilay, R. & Jaakkola, T. Generative models for graph-based protein design. in *Proc. 33rd Conference on Neural Information Processing Systems* (2019).
60. Johnson, L. S., Eddy, S. R. & Portugaly, E. Hidden Markov model speed heuristic and iterative HMM search procedure. *BMC Bioinformatics* **11**, 431 (2010).
61. Remmert, M., Biegert, A., Hauser, A. & Söding, J. HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment. *Nat. Methods* **9**, 173–175 (2012).
62. The UniProt Consortium. UniProt: the universal protein knowledgebase in 2021. *Nucleic Acids Res.* **49**, D480–D489 (2020).
63. Steinegger, M. & Söding, J. Clustering huge protein sequence sets in linear time. *Nat. Commun.* **9**, 2542 (2018).
64. Steinegger, M. & Söding, J. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nat. Biotechnol.* **35**, 1026–1028 (2017).
65. Deorowicz, S., Debudaj-Grabysz, A. & Gudyś, A. FAMSA: fast and accurate multiple sequence alignment of huge protein families. *Sci. Rep.* **6**, 33964 (2016).
66. Steinegger, M. et al. HH-suite3 for fast remote homology detection and deep protein annotation. *BMC Bioinformatics* **20**, 473 (2019).
67. Suzek, B. E., Wang, Y., Huang, H., McGarvey, P. B. & Wu, C. H. UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics* **31**, 926–932 (2015).
68. Eddy, S. R. Accelerated profile HMM searches. *PLOS Comput. Biol.* **7**, e1002195 (2011).
69. Eastman, P. et al. OpenMM 7: rapid development of high performance algorithms for molecular dynamics. *PLOS Comput. Biol.* **13**, e1005659 (2017).
70. Ashish, A. M. A. et al. TensorFlow: large-scale machine learning on heterogeneous systems. Preprint at https://arxiv.org/abs/1603.04467 (2015).
71. Reynolds, M. et al. Open sourcing Sonnet – a new library for constructing neural networks. *DeepMind* https://deepmind.com/blog/open-sourcing-sonnet/ (7 April 2017).
72. Harris, C. R. et al. Array programming with NumPy. *Nature* **585**, 357–362 (2020).
73. Van Rossum, G. & Drake, F. L. *Python 3 Reference Manual* (CreateSpace, 2009).
74. Bisong, E. in *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners* 59–64 (Apress, 2019).
75. TensorFlow. XLA: Optimizing Compiler for TensorFlow. https://www.tensorflow.org/xla (2018).
76. Wu, T., Hou, J., Adhikari, B. & Cheng, J. Analysis of several key factors influencing deep learning-based inter-residue contact prediction. *Bioinformatics* **36**, 1091–1098 (2020).
77. Jiang, W. et al. MrpH, a new class of metal-binding adhesin, requires zinc to mediate biofilm formation. *PLoS Pathog.* **16**, e1008707 (2020).
78. Dunne, M., Ernst, P., Sobieraj, A., Pluckthun, A. & Loessner, M. J. The M23 peptidase domain of the Staphylococcal phage 2638A endolysin. *PDB* https://doi.org/10.2210/pdb6YJ1/pdb (2020).
79. Drobysheva, A. V. et al. Structure and function of virion RNA polymerase of a crAss-like phage. *Nature* **589**, 306–309 (2021).
80. Flaugnatti, N. et al. Structural basis for loading and inhibition of a bacterial T6SS phospholipase effector by the VgrG spike. *EMBO J.* **39**, e104129 (2020).
81. ElGamacy, M. et al. An interface-driven design strategy yields a novel, corrugated protein architecture. *ACS Synth. Biol.* **7**, 2226–2235 (2018).
82. Lim, C. J. et al. The structure of human CST reveals a decameric assembly bound to telomeric DNA. *Science* **368**, 1081–1085 (2020).
83. Debruycker, V. et al. An embedded lipid in the multidrug transporter LmrP suggests a mechanism for polyspecificity. *Nat. Struct. Mol. Biol.* **27**, 829–835 (2020).
84. Flower, T. G. et al. Structure of SARS-CoV-2 ORF8, a rapidly evolving immune evasion protein. *Proc. Natl Acad. Sci. USA* **118**, e2021785118 (2021).

Acknowledgements We thank A. Rrustemi, A. Gu, A. Guseynov, B. Hechtman, C. Beattie, C. Jones, C. Donner, E. Parisotto, E. Elsen, F. Popovici, G. Necula, H. Maclean, J. Menick, J. Kirkpatrick, J. Molloy, J. Yim, J. Stanway, K. Simonyan, L. Sifre, L. Martens, M. Johnson, M. O'Neill, N. Antropova, R. Hadsell, S. Blackwell, S. Das, S. Hou, S. Gouws, S. Wheelwright, T. Hennigan, T. Ward, Z. Wu, Ž. Avsec and the Research Platform Team for their contributions; M. Mirdita for his help with the datasets; M. Piovesan-Forster, A. Nelson and R. Kemp for their help managing the project; the JAX, TensorFlow and XLA teams for detailed support and enabling machine learning models of the complexity of AlphaFold; our colleagues at DeepMind, Google and Alphabet for their encouragement and support; and J. Moult and the CASP14 organizers, and the experimentalists whose structures enabled the assessment. M.S. acknowledges support from the National Research Foundation of Korea grant (2019R1A6A1A10073437, 2020M3A9G7103933) and the Creative-Pioneering Researchers Program through Seoul National University.

Author contributions J.J. and D.H. led the research. J.J., R.E., A. Pritzel, M.F., O.R., R.B., A. Potapenko, S.A.A.K., B.R.-P., J.A., M.P., T. Berghammer and O.V. developed the neural network architecture and training. T.G., A.Ž., K.T., R.B., A.B., R.E., A.J.B., A.C., S.N., R.J., D.R., M.Z. and S.B. developed the data, analytics and inference systems. D.H., K.K., P.K., C.M. and E.C. managed the research. T.G. led the technical platform. P.K., A.W.S., K.K., O.V., D.S., S.P. and T. Back contributed technical advice and ideas. M.S. created the BFD genomics database and provided technical assistance on HHBlits. D.H., R.E., A.W.S. and K.K. conceived the AlphaFold project. J.J., R.E. and A.W.S. conceived the end-to-end approach. J.J., A. Pritzel, O.R., A. Potapenko, R.E., M.F., T.G., K.T., C.M. and D.H. wrote the paper.

Competing interests J.J., R.E., A. Pritzel, T.G., M.F., O.R., R.B., A.B., S.A.A.K., D.R. and A.W.S. have filed non-provisional patent applications 16/701,070 and PCT/EP2020/084238, and provisional patent applications 63/107,362, 63/118,917, 63/118,918, 63/118,921 and 63/118,919, each in the name of DeepMind Technologies Limited, each pending, relating to machine learning for predicting protein structures. The other authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41586-021-03819-2.

Correspondence and requests for materials should be addressed to J.J. or D.H.

Peer review information *Nature* thanks Mohammed AlQuraishi, Charlotte Deane and Yang Zhang for their contribution to the peer review of this work.

Supplementary Information for: Highly accurate protein structure prediction with AlphaFold

John Jumper[1*+], Richard Evans[1*], Alexander Pritzel[1*], Tim Green[1*], Michael Figurnov[1*], Olaf Ronneberger[1*], Kathryn Tunyasuvunakool[1*], Russ Bates[1*], Augustin Žídek[1*], Anna Potapenko[1*], Alex Bridgland[1*], Clemens Meyer[1*], Simon A A Kohl[1*], Andrew J Ballard[1*], Andrew Cowie[1*], Bernardino Romera-Paredes[1*], Stanislav Nikolov[1*], Rishub Jain[1*], Jonas Adler[1], Trevor Back[1], Stig Petersen[1], David Reiman[1], Ellen Clancy[1], Michal Zielinski[1], Martin Steinegger[2], Michalina Pacholska[1], Tamas Berghammer[1], Sebastian Bodenstein[1], David Silver[1], Oriol Vinyals[1], Andrew W Senior[1], Koray Kavukcuoglu[1], Pushmeet Kohli[1], and Demis Hassabis[1*+]

[1]DeepMind, London, UK
[2]School of Biological Sciences and Artificial Intelligence Institute, Seoul National University, Seoul, South Korea
[*]These authors contributed equally
[+]Corresponding authors: John Jumper, Demis Hassabis

Contents

List of Supplementary Figures

List of Supplementary Tables

List of Algorithms

1 Supplementary Methods

1.1 Notation

We denote the number of residues in the input primary sequence by N_{res} (cropped during training), the number of templates used in the model by N_{templ}, the number of all available MSA sequences by $N_{\text{all\_seq}}$, the number of clusters after MSA clustering by N_{clust}, the number of sequences processed in the MSA stack by N_{seq} (where $N_{\text{seq}} = N_{\text{clust}} + N_{\text{templ}}$), and the number of unclustered MSA sequences by $N_{\text{extra\_seq}}$ (after sub-sampling, see subsubsection 1.2.7 for details). Concrete values for these parameters are given in the training details (subsection 1.11). On the model side, we also denote the number of blocks in Evoformer-like stacks by N_{block} (subsection 1.6), the number of ensembling iterations by N_{ensemble} (subsubsection 1.11.2), and the number of recycling iterations by N_{cycle} (subsection 1.10).

We present architectural details in Algorithms, where we use the following conventions. We use capitalized operator names when they encapsulate learnt parameters, e.g. we use Linear for a linear transformation with a weights matrix W and a bias vector $\mathbf{b}$, and LinearNoBias for the linear transformation without the bias vector. Note that when we have multiple outputs from the Linear operator at the same line of an algorithm, we imply different trainable weights for each output. We use LayerNorm for the layer normalization [85] operating on the channel dimensions with learnable per-channel gains and biases. We also use capitalized names for random operators, such as those related to dropout. For functions without parameters we use lower case operator names, e.g. sigmoid, softmax, stopgrad. We use $\odot$ for the element-wise multiplication, $\otimes$ for the outer product, $\oplus$ for the outer sum, and $\mathbf{a}^\top \mathbf{b}$ for the dot product of two vectors. Indices i, j, k always operate on the residue dimension, indices s, t on the sequence dimension, and index h on the attention heads dimension. The channel dimension is implicit and we type the channel-wise vectors in bold, e.g. $\mathbf{z}_{ij}$. Algorithms operate on sets of such vectors, e.g. we use $\{\mathbf{z}_{ij}\}$ to denote all pair representations.

In the structure module, we denote Euclidean transformations corresponding to frames by $T = (R, \vec{\mathbf{t}})$, with $R \in \mathbb{R}^{3\times3}$ for the rotation and $\vec{\mathbf{t}} \in \mathbb{R}^3$ for the translation components. We use the $\circ$ operator to denote application of a transformation to an atomic position $\vec{\mathbf{x}} \in \mathbb{R}^3$:

$$\begin{aligned}\vec{\mathbf{x}}_{\text{result}} &= T \circ \vec{\mathbf{x}} \\ &= (R, \vec{\mathbf{t}}) \circ \vec{\mathbf{x}} \\ &= R\vec{\mathbf{x}} + \vec{\mathbf{t}}\,.\end{aligned}$$

The $\circ$ operator also denotes composition of Euclidean transformations:

$$\begin{aligned}T_{\text{result}} &= T_1 \circ T_2 \\ (R_{\text{result}}, \vec{\mathbf{t}}_{\text{result}}) &= (R_1, \vec{\mathbf{t}}_1) \circ (R_2, \vec{\mathbf{t}}_2) \\ &= (R_1 R_2, \quad R_1 \vec{\mathbf{t}}_2 + \vec{\mathbf{t}}_1)\end{aligned}$$

We use T^{-1} to denote the group inverse of the transform T:

$$\begin{aligned} T^{-1} &= (R, \vec{\mathbf{t}})^{-1} \\ &= (R^{-1},\ -R^{-1}\vec{\mathbf{t}}) \end{aligned}$$

1.2 Data pipeline

The data pipeline is the first step when running AlphaFold. It takes an mmCIF file (in the training mode) or a FASTA file (in the inference mode) and produces input features for the model. In the training mode, a single mmCIF file can produce multiple separate training examples, one for each chain in the file. The data pipeline includes the following steps.

1.2.1 Parsing

The input file is parsed and basic metadata is extracted from it. For FASTA, this is only the sequence and name; for mmCIF this is the sequence, atom coordinates, release date, name, and resolution. We also resolve alternative locations for atoms/residues, taking the one with the largest occupancy, change MSE residues into MET residues, and fix arginine naming ambiguities (making sure that NH1 is always closer to CD than NH2).

1.2.2 Genetic search

Multiple genetic databases are searched using JackHMMER v3.3 [86] and HHBlits v3.0-beta.3 [87]. We used JackHMMER with MGnify [88], JackHMMER with UniRef90 [89], HHBlits with Uniclust30 [90] + BFD (see Input and Data Sources in the main text methods for details). The output multiple sequence alignments (MSAs) are de-duplicated and stacked.

The MSA depth was limited to 5,000 sequences for JackHMMER with MGnify, 10,000 sequences for JackHMMER with UniRef90 and unlimited for HHBlits. The following flags were set to a non-default value for each of the tools:

JackHMMER: -N 1 -E 0.0001 --incE 0.0001 --F1 0.0005 --F2 0.00005 --F3 0.0000005.

HHBlits: -n 3 -e 0.001 -realign_max 100000 -maxfilt 100000 -min_prefilter_hits 1000 -maxseq 1000000.

1.2.3 Template search

The structural templates fed to the model are retrieved with the following steps:

1. The UniRef90 MSA obtained in the previous step is used to search PDB70 using HHSearch [91]. The only flag set to a non-default value for HHSearch runs was -maxseq 1000000. Structural data for each hit is obtained from the corresponding mmCIF file in the PDB database. If the sequence from PDB70 does not exactly match the sequence in the mmCIF file then the two are aligned using Kalign[92].

2. During training we exclude all templates that were released after the query training structure. We also filter out templates that are identical to (or a subset of) the input primary sequence, or that are too small (less than 10 residues or of length less than 10% of the primary sequence).

3. At inference time we provide the model the top 4 templates, sorted by the expected number of correctly aligned residues (the "sum_probs" feature output by HHSearch). At training time we first restrict the available templates to up to 20 with the highest "sum_probs". Then we choose random k templates out of this restricted set of n templates, where $k = \min(\text{Uniform}[0, n], 4)$. This has the effect of showing the network potentially bad templates, or no templates at all, during training so the network cannot rely on just copying the template.

1.2.4 Training data

With 75% probability a training example comes from the self-distillation set (see subsection 1.3) and with 25% probability the training example is a known structure from the Protein Data Bank. We loop over this hybrid set multiple times during training and we apply a number of stochastic filters (subsubsection 1.2.5), MSA pre-processing steps (subsubsection 1.2.6 and subsubsection 1.2.7), and residue cropping (subsubsection 1.2.8) every time when a protein is encountered. This means, we may observe different targets in training epochs, with different samples of the MSA data, and also cropped to different regions.

1.2.5 Filtering

The following filters are applied to the training data:

- Input mmCIFs are restricted to have resolution less than 9 Å. This is not a very restrictive filter and only removes around 0.2% of structures.
- Protein chains are accepted with probability $\frac{1}{512}\max(\min(N_{\text{res}}, 512), 256)$, where N_{res} is the length of the chain. This re-balances the length distribution and makes the network train on crops from the longer chains more often.
- Sequences are filtered out when any single amino acid accounts for more than 80% of the input primary sequence. This filter removes about 0.8% of sequences.
- Protein chains are accepted with the probability inverse to the size of the cluster that this chain falls into. We use 40% sequence identity clusters of the Protein Data Bank clustered with MMSeqs2 [93].

1.2.6 MSA block deletion

During training contiguous blocks of sequences are deleted from the MSA (see Algorithm 1). The MSA is grouped by tool and ordered by the normal output of each tool, typically e-value. This means that similar sequences are more likely to be adjacent in the MSA and block deletions are more likely to generate diversity that removes whole branches of the phylogeny.

Algorithm 1 MSA block deletion

def MSABlockDeletion($\mathbf{msa}$) :

block_size $= \lfloor 0.3 \cdot N_{\text{all\_seq}} \rfloor$
to_delete $= \{\}$
for all $j \in [1, \ldots, 5]$ **do**
  block_start $\leftarrow$ uniform$(1, N_{\text{all_seq}})$
  to_delete $\leftarrow$ to_delete $\cup$ [block_start, ..., block_start + block_size -1]
end for
keep $\leftarrow [1, \ldots, N_{\text{all\_seq}}] \setminus$ to_delete
$\mathbf{msa} \leftarrow \mathbf{msa}_{\text{keep}}$
return $\mathbf{msa}$

1.2.7 MSA clustering

The computational and peak memory cost of the main Evoformer module scales as $N_{\text{seq}}^2 \times N_{\text{res}}$, so it is highly desirable to reduce the number of sequences used in the main Evoformer module for purely computational reasons. Our first version of this procedure randomly chose a fixed-size subset of sequences without replacement when the MSA was too large (originally 128 sequences). This procedure has the clear downside that

sequences not included in the random subset have no influence on the prediction. The presented version is a modification of this procedure where we still choose a random subset of sequences without replacement to be representatives, but for each sequence in the full MSA we associate that sequence with the nearest sequence in the set of representatives (we call this "clustering" with random cluster centres though no attempt is made to ensure the cluster centres are well-distributed). To maintain the fixed-size and bounded cost properties, so we use only the amino acid and deletion frequencies of all sequences associated with each representative sequence and provide these mini-profiles as extra features in addition to the representative sequence (we roughly double the number of input features of each representative sequence without increasing the number of representative sequences). This allows all sequences to have some influence on the final prediction, which seems desirable.

Since this procedure achieves the goal of bounding computational cost, we have not experimented much with alternatives to the procedure. We have attempted a couple of methods to encourage diversity among the sampled sequences at inference time (e.g. a bias to pick representatives far from each other), but gains were very small to none so we did not pursue them further.

In detail the MSA is clustered (grouped) using the following procedure:

1. N_{clust} sequences are selected randomly as MSA cluster centres, the first cluster centre is always set to the query amino acid sequence.

2. A mask is generated such that each position in a MSA cluster centre has a 15% probability of being included in the mask. Each element in the MSA that is included in the mask is replaced in the following way:
   - With 10% probability amino acids are replaced with a uniformly sampled random amino acid.
   - With 10% probability amino acids are replaced with an amino acid sampled from the MSA profile for a given position.
   - With 10% probability amino acids are not replaced.
   - With 70% probability amino acids are replaced with a special token (masked_msa_token).

   These masked positions are the prediction targets used in subsubsection 1.9.9. Note that this masking is used both at training time, and at inference time.

3. The remaining sequences are assigned to their closest cluster by Hamming distance (ignoring masked out residues and gaps). For each sequence cluster, several statistics are computed, such as the per residue amino acid distribution). See "cluster" features in Table 1 for a full description.

4. The MSA sequences that have not been selected as cluster centres at step 1 are used to randomly sample $N_{\text{extra\_seq}}$ sequences without replacement. If there are less than $N_{\text{extra\_seq}}$ remaining sequences available, all of them are used. These sequences form "extra" MSA features in Table 1.

Please note that throughout the rest of the manuscript we use the term "MSA clusters" to refer to the clusters produced at step 3 above.

1.2.8 Residue cropping

During training the residue dimension in all data is cropped in the following way.

1. As described in subsubsection 1.11.5 mini-batches are processed in two modes. In the unclamped loss mode the crop start position is sampled from Uniform$[1, n - x + 1]$ where n is seq_length minus crop_size and x is sampled from Uniform$[0, n]$. In clamped loss mode the crop position is sampled from Uniform$[1, n + 1]$.

2. The residues dimension is cropped to a single contiguous region, with the crop start position defined above. The final crop size is denoted by N_{res} and its concrete value is provided in subsection 1.11.

1.2.9 Featurization and model inputs

Features in Table 1 are computed and aggregated into the following main inputs to the model:

- **target_feat** This is a feature of size $[N_{\text{res}}, 21]$ consisting of the "aatype" feature.
- **residue_index** This is a feature of size $[N_{\text{res}}]$ consisting of the "residue_index" feature.
- **msa_feat** This is a feature of size $[N_{\text{clust}}, N_{\text{res}}, 49]$ constructed by concatenating "cluster_msa", "cluster_has_deletion", "cluster_deletion_value", "cluster_deletion_mean", "cluster_profile". We draw $N_{\text{cycle}} \times N_{\text{ensemble}}$ random samples from this feature to provide each recycling/ensembling iteration of the network with a different sample (see subsubsection 1.11.2).
- **extra_msa_feat** This is a feature of size $[N_{\text{extra\_seq}}, N_{\text{res}}, 25]$ constructed by concatenating "extra_msa", "extra_msa_has_deletion", "extra_msa_deletion_value". Together with "msa_feat' above we also draw $N_{\text{cycle}} \times N_{\text{ensemble}}$ random samples from this feature (see subsubsection 1.11.2).
- **template_pair_feat** This is a feature of size $[N_{\text{templ}}, N_{\text{res}}, N_{\text{res}}, 88]$ and consists of concatenation of the pair residue features "template_distogram", "template_unit_vector", and also several residue features, which are transformed into pair features. The "template_aatype" feature is included via tiling and stacking (this is done twice, in both residue directions). Also the mask features "template_pseudo_beta_mask" and "template_backbone_frame_mask" are included, where the feature $f_{ij} = \text{mask}_i \cdot \text{mask}_j$.
- **template_angle_feat** This is a feature of size $[N_{\text{templ}}, N_{\text{res}}, 51]$ constructed by concatenating the following features: "template_aatype", "template_torsion_angles", "template_alt_torsion_angles", and "template_torsion_angles_mask".

Table 1 | Input features to the model. Feature dimensions: N_{res} is the number of residues, N_{clust} is the number of MSA clusters, $N_{\text{extra\_seq}}$ is the number of additional unclustered MSA sequences, and N_{templ} is the number of templates.

Feature & Shape	Description
aatype $[N_{\text{res}}, 21]$	One-hot representation of the input amino acid sequence (20 amino acids + unknown).
cluster_msa $[N_{\text{clust}}, N_{\text{res}}, 23]$	One-hot representation of the msa cluster centre sequences (20 amino acids + unknown + gap + masked_msa_token).
cluster_has_deletion $[N_{\text{clust}}, N_{\text{res}}, 1]$	A binary feature indicating if there is a deletion to the left of the residue in the MSA cluster centres.
cluster_deletion_value $[N_{\text{clust}}, N_{\text{res}}, 1]$	The raw deletion counts (the number of deletions to the left of every position in the MSA cluster centres) are transformed to the range $[0, 1]$ using $\frac{2}{\pi} \arctan \frac{d}{3}$ where d are the raw counts.
cluster_deletion_mean $[N_{\text{clust}}, N_{\text{res}}, 1]$	The mean deletions for every residue in every cluster are computed as $\frac{1}{n} \sum_{i=1}^{n} d_{ij}$ where n is the number of sequences in the cluster and d_{ij} is the number of deletions to the left of the ith sequence and jth residue. These are then transformed into the range $[0, 1]$ in the same way as for the cluster_deletion_value feature above.
cluster_profile $[N_{\text{clust}}, N_{\text{res}}, 23]$	The distribution across amino acid types for each residue in each MSA cluster (20 amino acids + unknown + gap + masked_msa_token).

Feature & Shape	**Description**
extra_msa $[N_{\text{extra\_seq}}, N_{\text{res}}, 23]$	One-hot representation of all MSA sequences not selected as cluster centres (20 amino acids + unknown + gap + masked_msa_token).
extra_msa_has_deletion $[N_{\text{extra\_seq}}, N_{\text{res}}, 1]$	A binary feature indicating if there is a deletion to the left of the residue in the extra MSA sequences.
extra_msa_deletion_value $[N_{\text{extra\_seq}}, N_{\text{res}}, 1]$	The raw deletion counts to the left of every residue in the extra_msa, converted to the range $[0, 1]$ using the same formula as for cluster_deletion_value.
template_aatype $[N_{\text{templ}}, N_{\text{res}}, 22]$	One-hot representation of the amino acid sequence (20 amino acids + unknown and gap).
template_mask $[N_{\text{templ}}, N_{\text{res}}]$	Mask indicating if a template residue exists and has coordinates.
template_pseudo_beta_mask $[N_{\text{templ}}, N_{\text{res}}]$	Mask indicating if the beta carbon (alpha carbon for glycine) atom has coordinates for the template at this residue.
template_backbone_frame_mask $[N_{\text{templ}}, N_{\text{res}}]$	A mask indicating if the coordinates of all the required atoms to compute the backbone frame (used in the template_unit_vector feature) exist.
template_distogram $[N_{\text{templ}}, N_{\text{res}}, N_{\text{res}}, 39]$	A one-hot pairwise feature indicating the distance between beta carbons (alpha carbon used for glycine) atoms. The pairwise distances are discretized into 38 bins of equal width between 3.25 Å and 50.75 Å; and one more bin contains any larger distances.
template_unit_vector $[N_{\text{templ}}, N_{\text{res}}, N_{\text{res}}, 3]$	The unit vector of the displacement of the alpha carbon atom of all residues within the local frame of each residue. These local frames are computed in the same way as for the target structure, see subsubsection 1.8.1. (Current models were trained with this feature set to zero.)
template_torsion_angles $[N_{\text{templ}}, N_{\text{res}}, 14]$	The 3 backbone torsion angles and up to 4 side-chain torsion angles for each residue represented as sine and cosine encoding.
template_alt_torsion_angles $[N_{\text{templ}}, N_{\text{res}}, 14]$	Alternative torsion angles for side chain parts with 180^o-rotation symmetry.
template_torsion_angles_mask $[N_{\text{templ}}, N_{\text{res}}, 14]$	A mask indicating if the torsion angle is present in the template structure.
residue_index $[N_{\text{res}}]$	The index into the original amino acid sequence.

1.3 Self-distillation dataset

We perform a self-distillation procedure similar to [94] on unlabelled protein sequences. To create the sequence dataset we compute multiple sequence alignments of every cluster in Uniclust30 (version 2018-08) against the same database. We then greedily remove sequences which appear in another sequence's MSA to give 6,318,986 sequences in total. We further filter this by removing sequences with more than 1,024 or fewer

than 200 amino acids, or whose MSA contain fewer than 200 alignments. This gives a final dataset of 355,993 sequences.

For building the distillation set predicted structures, we use a single (no re-ranking) "undistilled" model, i.e. trained on just the PDB dataset. Using this undistilled model, we predict the structure of every sequence in the set constructed above to create a dataset of structures to use at training time. For every pair of predicted residues we calculate a confidence metric by computing the Kullback–Leibler divergence between the pairwise distance distribution and a reference distribution for that residue separation:

$$c_{ij} = D_{KL}\left(p^{\text{ref}}_{|i-j|}(r) \middle|\middle| p_{i,j}(r)\right). \tag{1}$$

The reference distribution is computed by taking distance distribution predictions for 1,000 randomly sampled sequences in Uniclust30 and computing the mean distribution for a given sequence separation.

This pairwise metric c_{ij} is then averaged over j corresponding to a maximum sequence separation of ± 128 and excluding $i = j$ to give a per-residue confidence metric c_i. Similar to pLDDT, we find that higher values of this confidence correlate with higher prediction accuracy. At training time we mask out residues with $c_i < 0.5$. This KL-based metric was introduced into the model before we developed pLDDT, and we expect that filtering the distillation set based on pLDDT would work equally well or better.

At training time, extra augmentation is added to distillation dataset examples by uniformly sampling the MSA to 1000 sequences without replacement (this is on top of any sampling that happens in the data pipeline, see subsection 1.2).

1.4 AlphaFold Inference

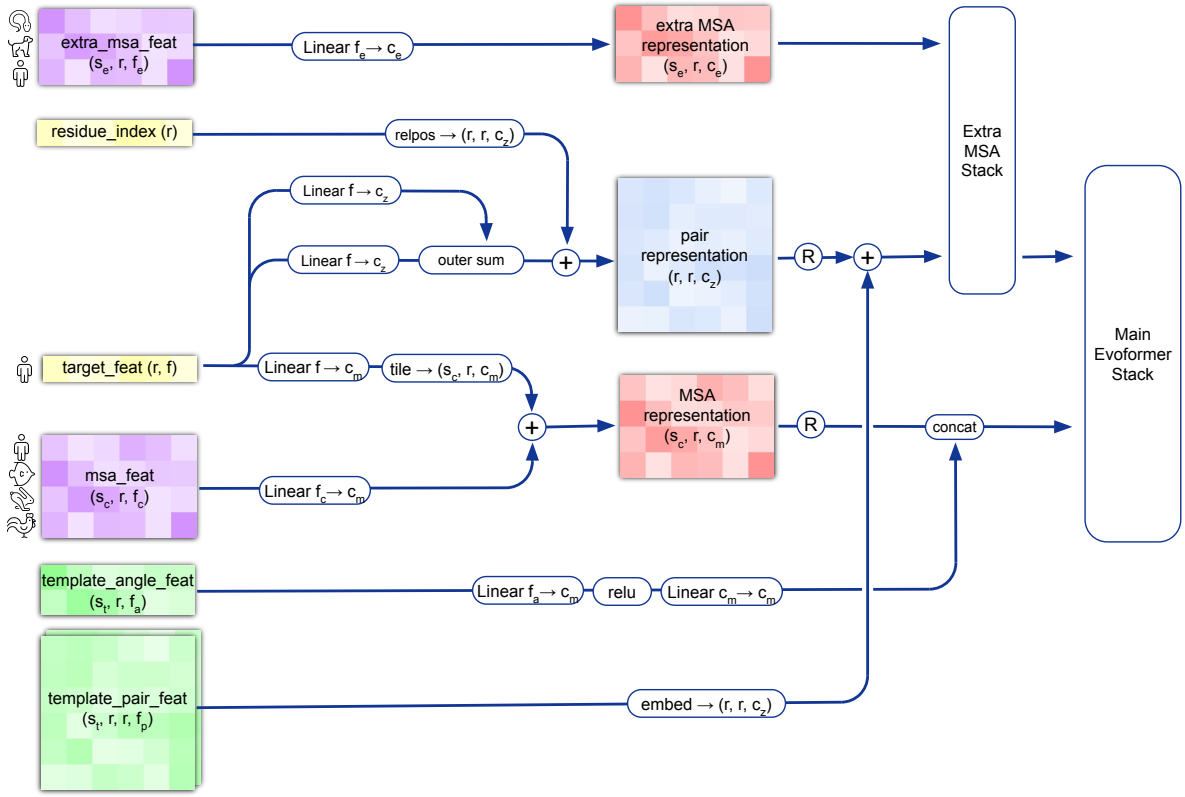
For inference, the AlphaFold receives input features derived from the amino-acid sequence, MSA, and templates (see subsubsection 1.2.9) and outputs features including atom coordinates, the distogram, and per-residue confidence scores. Algorithm 2 outlines the main steps (see also Fig 1e and the corresponding description in the main article).

The whole network is executed sequentially N_{cycle} times, where the outputs of the former execution are recycled as inputs for the next execution (see subsection 1.10 for details). The first part of the network (feature embedding and Evoformer, see Suppl. Fig. 1) is executed independently N_{ensemble} times with differently sampled inputs (see subsubsection 1.11.2). Their outputs are averaged to create the inputs for the Structure module and for recycling. These averaged Evoformer embeddings are denoted with an extra hat, e.g. $\hat{z}_{ij}$ for pair representations and $\hat{s}_i$ for single representations. The $N_{\text{cycle}} \times N_{\text{ensemble}}$ random samples of "msa_feat" and "extra_msa_feat" are denoted as a set of sets $\left\{\{\mathbf{f}^{\text{msa\_feat}}_{s_c i}\}_{c,n}\right\}$, and $\left\{\{\mathbf{f}^{\text{extra\_msa\_feat}}_{s_e i}\}_{c,n}\right\}$. Please note that ensembling is used only during inference and it has a very minor impact on the accuracy in the final system (subsubsection 1.12.2), thus can be considered as an optional technique.

The first part of the network (Algorithm 2 line 5) starts with embedding a new sample from the MSA (see subsection 1.5 for details) to create the initial version of the MSA representation $\{\mathbf{m}_{si}\}$, and the pair representation $\{\mathbf{z}_{ij}\}$. The first row of the MSA representation and the full pair representation are updated (Algorithm 2 line 6) by the recycled outputs from the previous iteration (see subsection 1.10 for details) – for the first iteration the recycled outputs are initialized to zero.

The next steps integrate the information from the templates (Algorithm 2 line 7). The "template_angle_feat" (derived from amino acid types and the torsion angles) are embedded by a shallow MLP and concatenated to the MSA representation. The "template_pair_feat" (derived from residue pairs, see subsubsection 1.2.9) are embedded by a shallow attention network (see subsubsection 1.7.1 for details) and added to the pair representation.

The final step of the embedding procedure (Algorithm 2 line 14) processes the extra MSA features (derived from individual unclustered MSA sequences) by shallow Evoformer-like network that is optimized for the large number of sequences (see subsubsection 1.7.2) to update the pair representation.



Supplementary Figure 1 | Input feature embeddings. Dimension names: r: residues, f: features, c: channels, s_c: clustered MSA sequences, s_e: extra MSA sequences, s_t: template sequences. Operator relpos is defined in Algorithm 4. Template embedding and extra MSA stack are explained in subsection 1.7. Recycling injection is denoted with R, see subsection 1.10 for details.

The main trunk of the network is the Evoformer stack (subsection 1.6) that produces the final representations (Algorithm 2 line 17). The pair representation, the first row of the MSA representation and a single representation derived from that row are kept for recycling and the structure module.

The second part of the network, the Structure Module (Algorithm 2 line 21), takes the final representations as input and predicts the atom coordinates $\{\vec{\mathbf{x}}_i^a\}$ and the per-residue confidence $\{r_i^{\mathrm{pLDDT}}\}$ (see subsection 1.8).

Algorithm 2 AlphaFold Model Inference

def Inference $\Big(\{\mathbf{f}_i^{\text{target\_feat}}\}, \{f_i^{\text{residue\_index}}\}, \Big\{\{\mathbf{f}_{s_c i}^{\text{msa\_feat}}\}_{c,n}\Big\}, \Big\{\{\mathbf{f}_{s_e i}^{\text{extra\_msa\_feat}}\}_{c,n}\Big\},$
$\{\mathbf{f}_{s_t i}^{\text{template\_angle\_feat}}\}, \{\mathbf{f}_{s_t ij}^{\text{template\_pair\_feat}}\}, N_{\text{cycle}} = 4, N_{\text{ensemble}} = 3\Big)$:

 # Recycling iterations:

$\hat{\mathbf{m}}_{1i}^{\text{prev}}, \hat{\mathbf{z}}_{ij}^{\text{prev}}, \vec{\mathbf{x}}_i^{\text{prev,C}^\beta} = \mathbf{0}, \mathbf{0}, \vec{\mathbf{0}}$

for all $c \in [1, \ldots, N_{\text{cycle}}]$ **do** *# shared weights*

 # Average representations in ensemble:

$\{\hat{\mathbf{m}}_{1i}\}, \{\hat{\mathbf{z}}_{ij}\}, \{\hat{\mathbf{s}}_i\} = \mathbf{0}, \mathbf{0}, \mathbf{0}$

for all $n \in [1, \ldots, N_{\text{ensemble}}]$ **do** *# shared weights*

 # Embed clustered MSA (use different MSA samples in each iteration):

$\{\mathbf{m}_{s_c i}\}, \{\mathbf{z}_{ij}\} = \text{InputEmbedder}(\{\mathbf{f}_i^{\text{target\_feat}}\}, \{f_i^{\text{residue\_index}}\}, \{\mathbf{f}_{s_c i}^{\text{msa\_feat}}\}_{c,n})$

 # Inject previous outputs for recycling:

$\{\mathbf{m}_{1i}\}, \{\mathbf{z}_{ij}\}$ += $\text{RecyclingEmbedder}(\{\hat{\mathbf{m}}_{1i}^{\text{prev}}\}, \{\hat{\mathbf{z}}_{ij}^{\text{prev}}\}, \{\vec{\mathbf{x}}_i^{\text{prev,C}^\beta}\})$

 # Embed templates:

$\mathbf{a}_{s_t i} \leftarrow \text{Linear}(\text{relu}(\text{Linear}(\mathbf{f}_{s_t i}^{\text{template\_angle\_feat}})))$ $\quad \mathbf{a}_{s_t i} \in \mathbb{R}^{c_m}, c_m = 256$

$\mathbf{m}_{si} = \text{concat}_s(\mathbf{m}_{s_c i}, \mathbf{a}_{s_t i})$

$\mathbf{t}_{s_t ij} = \text{Linear}(\mathbf{f}_{s_t ij}^{\text{template\_pair\_feat}})$ $\quad \mathbf{t}_{s_t ij} \in \mathbb{R}^{c_t}, c_t = 64$

for all $s_t \in [1, \ldots, N_{\text{templ}}]$ **do** *# shared weights*

$\{\mathbf{t}_{s_t ij}\} \leftarrow \text{TemplatePairStack}(\{\mathbf{t}_{s_t ij}\})$

end for

$\{\mathbf{z}_{ij}\}$ += $\text{TemplatePointwiseAttention}(\{\mathbf{t}_{s_t ij}\}, \{\mathbf{z}_{ij}\})$

 # Embed extra MSA features (use different samples in each iteration):

$\{\mathbf{a}_{s_e i}\} \leftarrow \{\mathbf{f}_{s_e i}^{\text{extra\_msa\_feat}}\}_{c,n}$

$\mathbf{e}_{s_e i} = \text{Linear}(\mathbf{a}_{s_e i})$ $\quad \mathbf{e}_{s_e i} \in \mathbb{R}^{c_e}, c_e = 64$

$\{\mathbf{z}_{ij}\} \leftarrow \text{ExtraMsaStack}(\{\mathbf{e}_{s_e i}\}, \{\mathbf{z}_{ij}\})$

 # Main trunk of the network:

$\{\mathbf{m}_{si}\}, \{\mathbf{z}_{ij}\}, \{\mathbf{s}_i\} \leftarrow \text{EvoformerStack}(\{\mathbf{m}_{si}\}, \{\mathbf{z}_{ij}\})$

$\{\hat{\mathbf{m}}_{1i}\}, \{\hat{\mathbf{z}}_{ij}\}, \{\hat{\mathbf{s}}_i\}$ += $\{\mathbf{m}_{1i}\}, \{\mathbf{z}_{ij}\}, \{\mathbf{s}_i\}$

end for

$\{\hat{\mathbf{m}}_{1i}\}, \{\hat{\mathbf{z}}_{ij}\}, \{\hat{\mathbf{s}}_i\}$ /= N_{ensemble}

 # Structure and confidence prediction:

$\{\vec{\mathbf{x}}_i^a\}, \{r_i^{\text{pLDDT}}\} = \text{StructureModule}(\{\hat{\mathbf{s}}_i\}, \{\hat{\mathbf{z}}_{ij}\})$

$\{\hat{\mathbf{m}}_{1i}^{\text{prev}}\}, \{\hat{\mathbf{z}}_{ij}^{\text{prev}}\}, \{\vec{\mathbf{x}}_i^{\text{prev,C}^\beta}\} \leftarrow \{\hat{\mathbf{m}}_{1i}\}, \{\hat{\mathbf{z}}_{ij}\}, \{\vec{\mathbf{x}}_i^{\text{C}^\beta}\}$

end for

return $\{\vec{\mathbf{x}}_i^a\}, \{r_i^{\text{pLDDT}}\}$

1.5 Input embeddings

Input features listed in subsubsection 1.2.9 are embedded into the network according to the Suppl. Fig. 1. Initial embedding details are also presented in Algorithm 3. Input primary sequence and MSA features are transformed to form MSA representation $\{\mathbf{m}_{si}\}$ with $\mathbf{m}_{si} \in \mathbb{R}^{c_m}$, $c_m = 256$, and $s \in \{1 \dots N_{\text{seq}}\}$, $i \in \{1 \dots N_{\text{res}}\}$. Primary sequence features are transformed to form pair representation $\{\mathbf{z}_{ij}\}$ with $\mathbf{z}_{ij} \in \mathbb{R}^{c_z}$, $c_z = 128$, and $i, j \in \{1 \dots N_{\text{res}}\}$. These two representations are also informed by the templates and unclustered MSA features. The details of their corresponding embedding modules are provided in subsection 1.7.

Algorithm 3 Embeddings for initial representations

def $\text{InputEmbedder}(\{\mathbf{f}_i^{\text{target\_feat}}\}, \{f_i^{\text{residue\_index}}\}, \{\mathbf{f}_{s_c i}^{\text{msa\_feat}}\})$:

1: $\mathbf{a}_i, \mathbf{b}_i = \text{Linear}(\mathbf{f}_i^{\text{target\_feat}})$ — $\mathbf{a}_i, \mathbf{b}_i \in \mathbb{R}^{c_z}, c_z = 128$

2: $\mathbf{z}_{ij} = \mathbf{a}_i + \mathbf{b}_j$ — $\mathbf{z}_{ij} \in \mathbb{R}^{c_z}, c_z = 128$

3: $\{\mathbf{z}_{ij}\}$ += $\text{relpos}(\{f_i^{\text{residue\_index}}\})$

4: $\mathbf{m}_{s_c i} = \text{Linear}(\mathbf{f}_{s_c i}^{\text{msa\_feat}}) + \text{Linear}(\mathbf{f}_i^{\text{target\_feat}})$ — $\mathbf{m}_{s_c i} \in \mathbb{R}^{c_m}, c_m = 256$

5: **return** $\{\mathbf{m}_{s_c i}\}, \{\mathbf{z}_{ij}\}$

To provide the network with information about the positions of residues in the chain, we also encode relative positional features (Algorithm 4) into the initial pair representations. Specifically, for a residue pair $i, j \in \{1 \dots N_{\text{res}}\}$ we compute the clipped relative distance within a chain, encode it as a one-hot vector, and add this vector's linear projection to $\mathbf{z}_{ij}$. Since we are clipping by the maximum value 32, any larger distances within the residue chain will not be distinguished by this feature. This inductive bias de-emphasizes primary sequence distances. Compared to the more traditional approach of encoding positions in the frequency space [95], this relative encoding scheme empirically allows the network to be evaluated without quality degradation on much longer sequences than it was trained on. A related construction was used in [96].

Algorithm 4 Relative position encoding

def $\text{relpos}(\{f_i^{\text{residue\_index}}\}, \mathbf{v}_{\text{bins}} = [-32, -31, \dots, 32])$:

1: $d_{ij} = f_i^{\text{residue\_index}} - f_j^{\text{residue\_index}}$ — $d_{ij} \in \mathbb{Z}$

2: $\mathbf{p}_{ij} = \text{Linear}(\text{one\_hot}(d_{ij}, \mathbf{v}_{\text{bins}}))$ — $\mathbf{p}_{ij} \in \mathbb{R}^{c_z}$

3: **return** $\{\mathbf{p}_{ij}\}$

Algorithm 5 One-hot encoding with nearest bin

def $\text{one\_hot}(x, \mathbf{v}_{\text{bins}})$: — $x \in \mathbb{R}$, $\mathbf{v}_{\text{bins}} \in \mathbb{R}^{N_{\text{bins}}}$

1: $\mathbf{p} = \mathbf{0}$ — $\mathbf{p} \in \mathbb{R}^{N_{\text{bins}}}$

2: $b = \arg\min(|x - \mathbf{v}_{\text{bins}}|)$

3: $p_b = 1$

4: **return** $\mathbf{p}$

1.6 Evoformer blocks

The network has a two tower architecture with axial self-attention in the MSA stack; triangular multiplicative updates and triangular self-attention in the pair stack; and outer product mean and attention biasing to allow communication between the stacks.

The main trunk of the network consists of $N_{\text{block}} = 48$ Evoformer blocks (**Fig. 1e, 3a**, Algorithm 6). Each block has an MSA representation $\{\mathbf{m}_{si}\}$ and a pair representation $\{\mathbf{z}_{ij}\}$ as its input and output and processes them with several layers. Each layer output is added via a residual connection to the current representations. Some layer outputs are passed through Dropout [97] before they are added (see subsubsection 1.11.6 for details).

The final Evoformer block provides a highly processed MSA representation $\{\mathbf{m}_{si}\}$ and a pair representation $\{\mathbf{z}_{ij}\}$, which contain information required for the structure module (subsection 1.8) and auxiliary network heads (subsection 1.9). The prediction modules are also using a "single" sequence representation $\{\mathbf{s}_i\}$ with $\mathbf{s}_i \in \mathbb{R}^{c_s}$, $c_s = 384$ and $i \in \{1 \dots N_{\text{res}}\}$. This single representation is derived by a linear projection of the first row of the MSA representation.

We now present details of the individual layers.

Algorithm 6 Evoformer stack

def EvoformerStack($\{\mathbf{m}_{si}\}, \{\mathbf{z}_{ij}\}, N_{\text{block}} = 48, c_s = 384$) :

1: **for all** $l \in [1, \dots, N_{\text{block}}]$ **do**

 # MSA stack

2: $\{\mathbf{m}_{si}\}$ += DropoutRowwise$_{0.15}$(MSARowAttentionWithPairBias($\{\mathbf{m}_{si}\}, \{\mathbf{z}_{ij}\}$))

3: $\{\mathbf{m}_{si}\}$ += MSAColumnAttention($\{\mathbf{m}_{si}\}$)

4: $\{\mathbf{m}_{si}\}$ += MSATransition($\{\mathbf{m}_{si}\}$)

 # Communication

5: $\{\mathbf{z}_{ij}\}$ += OuterProductMean($\{\mathbf{m}_{si}\}$)

 # Pair stack

6: $\{\mathbf{z}_{ij}\}$ += DropoutRowwise$_{0.25}$(TriangleMultiplicationOutgoing($\{\mathbf{z}_{ij}\}$))

7: $\{\mathbf{z}_{ij}\}$ += DropoutRowwise$_{0.25}$(TriangleMultiplicationIncoming($\{\mathbf{z}_{ij}\}$))

8: $\{\mathbf{z}_{ij}\}$ += DropoutRowwise$_{0.25}$(TriangleAttentionStartingNode($\{\mathbf{z}_{ij}\}$))

9: $\{\mathbf{z}_{ij}\}$ += DropoutColumnwise$_{0.25}$(TriangleAttentionEndingNode($\{\mathbf{z}_{ij}\}$))

10: $\{\mathbf{z}_{ij}\}$ += PairTransition($\{\mathbf{z}_{ij}\}$)

11: **end for**

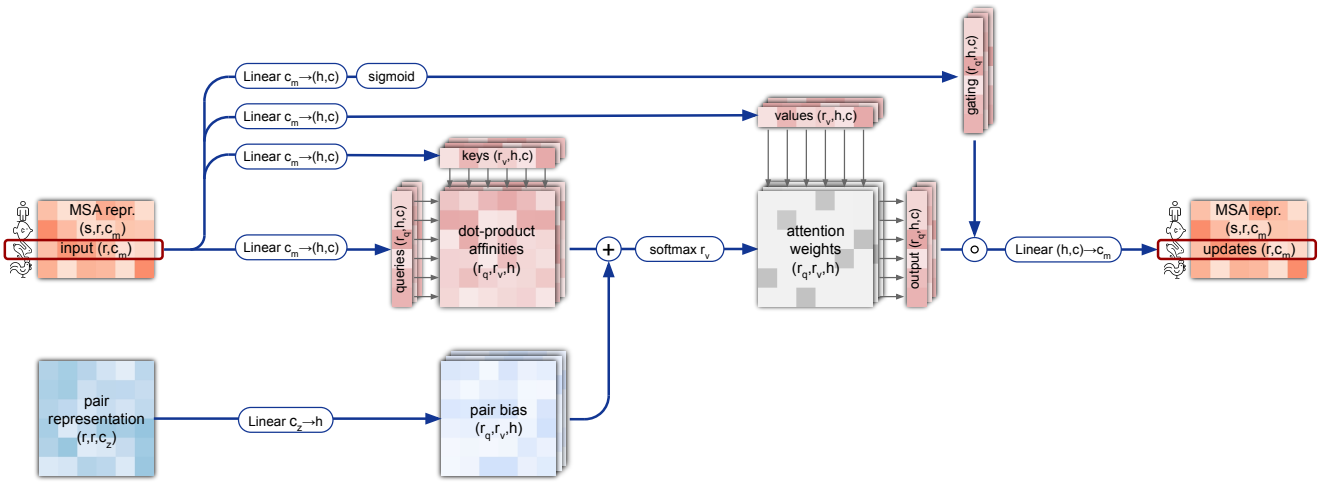
 # Extract the single representation

12: $\mathbf{s}_i = \text{Linear}(\mathbf{m}_{1i})$ $\quad \mathbf{s}_i \in \mathbb{R}^{c_s}$

13: **return** $\{\mathbf{m}_{si}\}, \{\mathbf{z}_{ij}\}, \{\mathbf{s}_i\}$

1.6.1 MSA row-wise gated self-attention with pair bias

MSA representations are processed with consecutive blocks of gated row-wise and column-wise self-attention blocks. The row-wise version (Suppl. Fig. 2 and Algorithm 7) builds attention weights for residue pairs and integrates the information from the pair representation as an additional bias term (line 3). This allows communication from the pair stack into the MSA stack to encourage consistency between them.



Supplementary Figure 2 | MSA row-wise gated self-attention with pair bias. Dimensions: s: sequences, r: residues, c: channels, h: heads.

Algorithm 7 MSA row-wise gated self-attention with pair bias

def $\text{MSARowAttentionWithPairBias}(\{\mathbf{m}_{si}\}, \{\mathbf{z}_{ij}\}, c = 32, N_{\text{head}} = 8)$:

 # Input projections

$\mathbf{m}_{si} \leftarrow \text{LayerNorm}(\mathbf{m}_{si})$

$\mathbf{q}^h_{si}, \mathbf{k}^h_{si}, \mathbf{v}^h_{si} = \text{LinearNoBias}(\mathbf{m}_{si})$ $\qquad \mathbf{q}^h_{si}, \mathbf{k}^h_{si}, \mathbf{v}^h_{si} \in \mathbb{R}^c,\ h \in \{1, \ldots, N_{\text{head}}\}$

$b^h_{ij} = \text{LinearNoBias}(\text{LayerNorm}(\mathbf{z}_{ij}))$

$\mathbf{g}^h_{si} = \text{sigmoid}\left(\text{Linear}(\mathbf{m}_{si})\right)$ $\qquad \mathbf{g}^h_{si} \in \mathbb{R}^c$

 # Attention

$a^h_{sij} = \text{softmax}_j\left(\frac{1}{\sqrt{c}}\, \mathbf{q}^{h\top}_{si} \mathbf{k}^h_{sj} + b^h_{ij}\right)$

$\mathbf{o}^h_{si} = \mathbf{g}^h_{si} \odot \sum_j a^h_{sij} \mathbf{v}^h_{sj}$

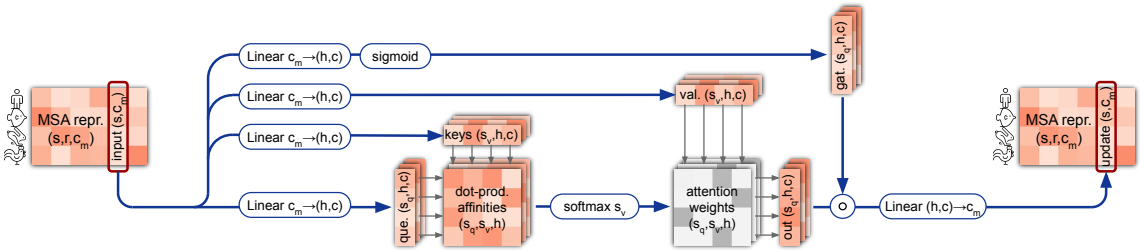
 # Output projection

$\tilde{\mathbf{m}}_{si} = \text{Linear}\left(\text{concat}_h(\mathbf{o}^h_{si})\right)$ $\qquad \tilde{\mathbf{m}}_{si} \in \mathbb{R}^{c_m}$

return $\{\tilde{\mathbf{m}}_{si}\}$

1.6.2 MSA column-wise gated self-attention

The column-wise attention (and Suppl. Fig. 3 and Algorithm 8) lets the elements that belong to the same target residue exchange information. In both attention blocks the number of heads $N_{\text{heads}} = 8$, and dimension of the keys, queries, and values $c = 32$.



Supplementary Figure 3 | MSA column-wise gated self-attention. Dimensions: s: sequences, r: residues, c: channels, h: heads.

Algorithm 8 MSA column-wise gated self-attention

def $\text{MSAColumnAttention}(\{\mathbf{m}_{si}\}, c = 32, N_{\text{head}} = 8)$:

 # Input projections

$\mathbf{m}_{si} \leftarrow \text{LayerNorm}(\mathbf{m}_{si})$

$\mathbf{q}^h_{si}, \mathbf{k}^h_{si}, \mathbf{v}^h_{si} = \text{LinearNoBias}(\mathbf{m}_{si})$ $\qquad \mathbf{q}^h_{si}, \mathbf{k}^h_{si}, \mathbf{v}^h_{si} \in \mathbb{R}^c,\ h \in \{1, \ldots, N_{\text{head}}\}$

$\mathbf{g}^h_{si} = \text{sigmoid}\left(\text{Linear}(\mathbf{m}_{si})\right)$ $\qquad \mathbf{g}^h_{si} \in \mathbb{R}^c$

 # Attention

$a^h_{sti} = \text{softmax}_t\left(\frac{1}{\sqrt{c}}\, \mathbf{q}^{h\top}_{si} \mathbf{k}^h_{ti}\right)$

$\mathbf{o}^h_{si} = \mathbf{g}^h_{si} \odot \sum_t a^h_{sti} \mathbf{v}^h_{st}$

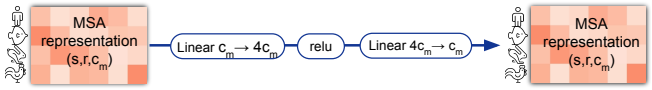
 # Output projection

$\tilde{\mathbf{m}}_{si} = \text{Linear}\left(\text{concat}_h(\mathbf{o}^h_{si})\right)$ $\qquad \tilde{\mathbf{m}}_{si} \in \mathbb{R}^{c_m}$

return $\{\tilde{\mathbf{m}}_{si}\}$

1.6.3 MSA transition

After row-wise and column-wise attention the MSA stack contains a 2-layer MLP as the transition layer (Suppl. Fig. 4 and Algorithm 9). The intermediate number of channels expands the original number of channels by a factor of 4.



Supplementary Figure 4 | MSA transition layer. Dimensions: s: sequences, r: residues, c: channels.

Algorithm 9 Transition layer in the MSA stack

def MSATransition($\{\mathbf{m}_{si}\}, n = 4$) :

$\mathbf{m}_{si} \leftarrow \text{LayerNorm}(\mathbf{m}_{si})$

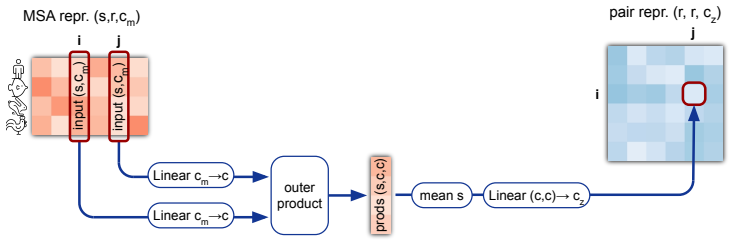
$\mathbf{a}_{si} = \text{Linear}(\mathbf{m}_{si})$ $\quad \mathbf{a}_{si} \in \mathbb{R}^{n \cdot c_m}$

$\mathbf{m}_{si} \leftarrow \text{Linear}(\text{relu}(\mathbf{a}_{si}))$

return $\{\mathbf{m}_{si}\}$

1.6.4 Outer product mean

The "Outer product mean" block transforms the MSA representation into an update for the pair representation (Suppl. Fig. 5 and Algorithm 10). All MSA entries are linearly projected to a smaller dimension $c = 32$ with two independent Linear transforms. The outer products of these vectors from two columns i and j are averaged over the sequences and projected to dimension c_z to obtain an update for entry ij in the pair representation. This is a memory intensive operation, as it requires constructing high-dimensional intermediate tensors. See section 1.11.8 for implementation details.



Supplementary Figure 5 | Outer product mean. Dimensions: s: sequences, r: residues, c: channels.

Algorithm 10 Outer product mean

def OuterProductMean($\{\mathbf{m}_{si}\}, c = 32$) :

$\mathbf{m}_{si} \leftarrow \text{LayerNorm}(\mathbf{m}_{si})$

$\mathbf{a}_{si}, \mathbf{b}_{si} = \text{Linear}(\mathbf{m}_{si})$ $\quad \mathbf{a}_{si}, \mathbf{b}_{si} \in \mathbb{R}^{c}$

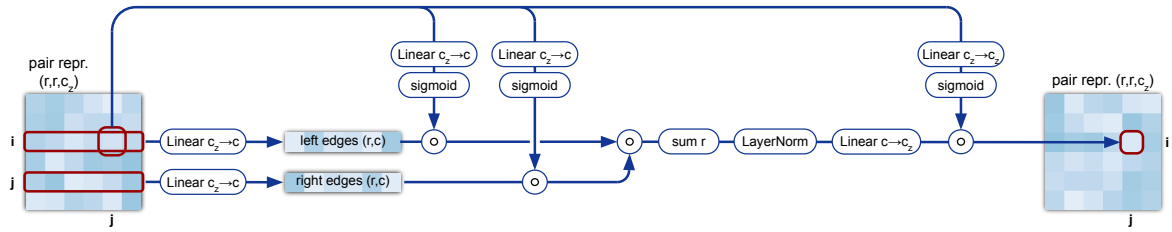
$\mathbf{o}_{ij} = \text{flatten}\left(\text{mean}_s(\mathbf{a}_{si} \otimes \mathbf{b}_{sj})\right)$ $\quad \mathbf{o}_{ij} \in \mathbb{R}^{c \cdot c}$

$\mathbf{z}_{ij} = \text{Linear}(\mathbf{o}_{ij})$ $\quad \mathbf{z}_{ij} \in \mathbb{R}^{c_z}$

return $\{\mathbf{z}_{ij}\}$

1.6.5 Triangular multiplicative update

The triangular multiplicative update (**Fig. 3c**) updates the pair representation in the Evoformer block by combining information within each triangle of graph edges ij, ik, and jk. Each edge ij receives an update from the other two edges of all triangles, where it is involved. There are two symmetric versions, one for the "outgoing" edges (Suppl. Fig. 6 and Algorithm 11) and one for the "incoming" edges (Algorithm 12). The differences are highlighted in yellow.



Supplementary Figure 6 | Triangular multiplicative update using "outgoing" edges. Dimensions: r: residues, c: channels.

Algorithm 11 Triangular multiplicative update using "outgoing" edges

def TriangleMultiplicationOutgoing($\{\mathbf{z}_{ij}\}, c = 128$) :

$\mathbf{z}_{ij} \leftarrow \text{LayerNorm}(\mathbf{z}_{ij})$

$\mathbf{a}_{ij}, \mathbf{b}_{ij} = \text{sigmoid}\left(\text{Linear}(\mathbf{z}_{ij})\right) \odot \text{Linear}(\mathbf{z}_{ij})$ $\quad \mathbf{a}_{ij}, \mathbf{b}_{ij} \in \mathbb{R}^{c}$

$\mathbf{g}_{ij} = \text{sigmoid}\left(\text{Linear}(\mathbf{z}_{ij})\right)$ $\quad \mathbf{g}_{ij} \in \mathbb{R}^{c_z}$

$\tilde{\mathbf{z}}_{ij} = \mathbf{g}_{ij} \odot \text{Linear}(\text{LayerNorm}(\sum_k \mathbf{a}_{ik} \odot \mathbf{b}_{jk}))$ $\quad \tilde{\mathbf{z}}_{ij} \in \mathbb{R}^{c_z}$

return $\{\tilde{\mathbf{z}}_{ij}\}$

Algorithm 12 Triangular multiplicative update using "incoming" edges

def TriangleMultiplicationIncoming($\{\mathbf{z}_{ij}\}, c = 128$) :

$\mathbf{z}_{ij} \leftarrow \text{LayerNorm}(\mathbf{z}_{ij})$

$\mathbf{a}_{ij}, \mathbf{b}_{ij} = \text{sigmoid}\left(\text{Linear}(\mathbf{z}_{ij})\right) \odot \text{Linear}(\mathbf{z}_{ij})$ $\quad \mathbf{a}_{ij}, \mathbf{b}_{ij} \in \mathbb{R}^{c}$

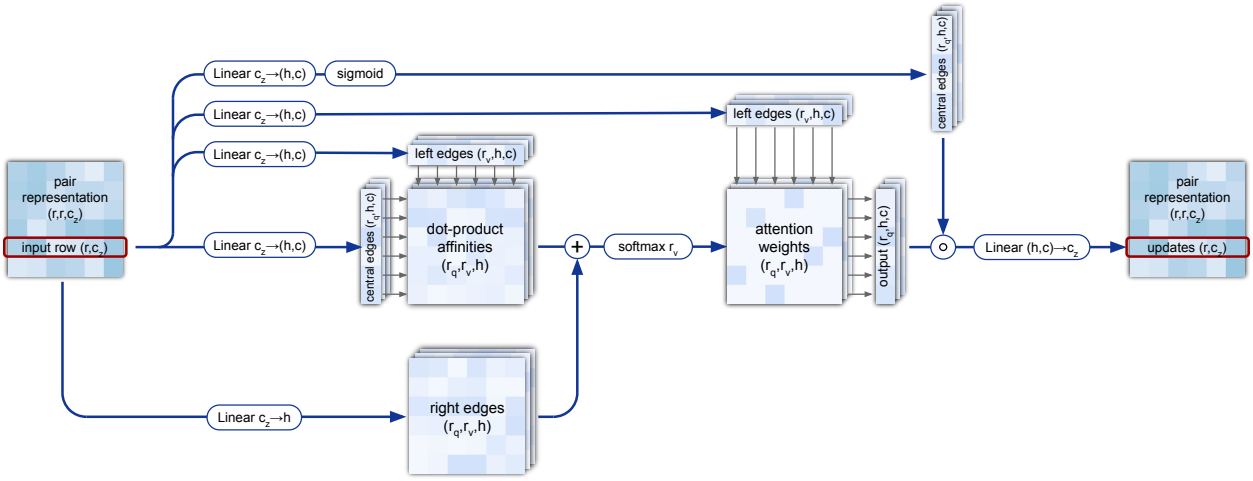
$\mathbf{g}_{ij} = \text{sigmoid}\left(\text{Linear}(\mathbf{z}_{ij})\right)$ $\quad \mathbf{g}_{ij} \in \mathbb{R}^{c_z}$

$\tilde{\mathbf{z}}_{ij} = \mathbf{g}_{ij} \odot \text{Linear}(\text{LayerNorm}(\sum_k \mathbf{a}_{ki} \odot \mathbf{b}_{kj}))$ $\quad \tilde{\mathbf{z}}_{ij} \in \mathbb{R}^{c_z}$

return $\{\tilde{\mathbf{z}}_{ij}\}$

1.6.6 Triangular self-attention

The triangular self-attention (**Fig. 3c**) updates the pair representation in the Evoformer block. The "starting node" version (Suppl. Fig. 7 and Algorithm 13) updates the edge ij with values from all edges that share the same starting node i, (i.e. all edges ik). The decision whether edge ij will receive an update from edge ik is not only determined by their query-key similarity (as in standard attention [95]), but also modulated by the information b_{jk} derived from the third edge jk of this triangle. Furthermore we extend the update with an additional gating g_{ij} derived from edge ij. The symmetric pair of this module operates on the edges around the ending node (Algorithm 14). The differences are highlighted in yellow.



Supplementary Figure 7 | Triangular self-attention around starting node. Dimensions: r: residues, c: channels, h: heads

Algorithm 13 Triangular gated self-attention around starting node

def TriangleAttentionStartingNode($\{\mathbf{z}_{ij}\}, c = 32, N_{\text{head}} = 4$) :

 # Input projections

$\mathbf{z}_{ij} \leftarrow \text{LayerNorm}(\mathbf{z}_{ij})$

$\mathbf{q}_{ij}^h, \mathbf{k}_{ij}^h, \mathbf{v}_{ij}^h = \text{LinearNoBias}(\mathbf{z}_{ij})$ $\qquad \mathbf{q}_{ij}^h, \mathbf{k}_{ij}^h, \mathbf{v}_{ij}^h \in \mathbb{R}^c,\ h \in \{1, \ldots, N_{\text{head}}\}$

$b_{ij}^h = \text{LinearNoBias}(\mathbf{z}_{ij})$

$\mathbf{g}_{ij}^h = \text{sigmoid}\left(\text{Linear}(\mathbf{z}_{ij})\right)$ $\qquad \mathbf{g}_{ij}^h \in \mathbb{R}^c$

 # Attention

$a_{ijk}^h = \text{softmax}_k\left(\frac{1}{\sqrt{c}}\, \mathbf{q}_{ij}^{h\top} \mathbf{k}_{ik}^h + b_{jk}^h\right)$

$\mathbf{o}_{ij}^h = \mathbf{g}_{ij}^h \odot \sum_k a_{ijk}^h \mathbf{v}_{ik}^h$

 # Output projection

$\tilde{\mathbf{z}}_{ij} = \text{Linear}\left(\text{concat}_h(\mathbf{o}_{ij}^h)\right)$ $\qquad \tilde{\mathbf{z}}_{ij} \in \mathbb{R}^{c_z}$

return $\{\tilde{\mathbf{z}}_{ij}\}$

Algorithm 14 Triangular gated self-attention around ending node

def TriangleAttentionEndingNode($\{\mathbf{z}_{ij}\}, c = 32, N_{\text{head}} = 4$) :

 # Input projections

$\mathbf{z}_{ij} \leftarrow \text{LayerNorm}(\mathbf{z}_{ij})$

$\mathbf{q}_{ij}^h, \mathbf{k}_{ij}^h, \mathbf{v}_{ij}^h = \text{LinearNoBias}(\mathbf{z}_{ij})$ $\quad \mathbf{q}_{ij}^h, \mathbf{k}_{ij}^h, \mathbf{v}_{ij}^h \in \mathbb{R}^c,\ h \in \{1, \dots, N_{\text{head}}\}$

$b_{ij}^h = \text{LinearNoBias}(\mathbf{z}_{ij})$

$\mathbf{g}_{ij}^h = \text{sigmoid}\left(\text{Linear}(\mathbf{z}_{ij})\right)$ $\quad \mathbf{g}_{ij}^h \in \mathbb{R}^c$

 # Attention

$a_{ijk}^h = \text{softmax}_k\left(\frac{1}{\sqrt{c}}\, \mathbf{q}_{ij}^{h\top} \mathbf{k}_{kj}^h + b_{ki}^h\right)$

$\mathbf{o}_{ij}^h = \mathbf{g}_{ij}^h \odot \sum_k a_{ijk}^h \mathbf{v}_{kj}^h$

 # Output projection

$\tilde{\mathbf{z}}_{ij} = \text{Linear}\left(\text{concat}_h\left(\mathbf{o}_{ij}^h\right)\right)$ $\quad \tilde{\mathbf{z}}_{ij} \in \mathbb{R}^{c_z}$

return $\{\tilde{\mathbf{z}}_{ij}\}$

1.6.7 Transition in the pair stack

The transition layer in the pair stack (Algorithm 15) is equivalent to that in the MSA stack: a 2-layer MLP where the intermediate number of channels expands the original number of channels by a factor of 4.

Algorithm 15 Transition layer in the pair stack

def PairTransition($\{\mathbf{z}_{ij}\}, n = 4$) :

$\mathbf{z}_{ij} \leftarrow \text{LayerNorm}(\mathbf{z}_{ij})$

$\mathbf{a}_{ij} = \text{Linear}(\mathbf{z}_{ij})$ $\quad \mathbf{a}_{ij} \in \mathbb{R}^{n \cdot c_z}$

$\mathbf{z}_{ij} \leftarrow \text{Linear}(\text{relu}(\mathbf{a}_{ij}))$

return $\{\mathbf{z}_{ij}\}$

1.7 Additional inputs

As outlined in Suppl. Fig. 1, the additional model inputs include templates and unclustered MSA sequences. In this section we explain how they are embedded in more detail.

1.7.1 Template stack

Pairwise template features are linearly projected to form initial template representations $\mathbf{t}_{s_t ij}$, with $\mathbf{t}_{s_t ij} \in \mathbb{R}^{c_t}$, $c_t = 64$, and $i, j \in \{1 \dots N_{\text{res}}\}$, $s_t \in \{1 \dots N_{\text{templ}}\}$. Each template representation is independently processed with the template pair stack (Algorithm 16) with all trainable parameters shared across templates. The output representations are aggregated with the template point-wise attention (Algorithm 17), where the pair representations $\{\mathbf{z}_{ij}\}$ are used to form the queries and attend over the individual templates. The outputs of this module are added to the pair representations (see Suppl. Fig. 1 or line 13 in Algorithm 2).

Furthermore, template torsion angle features are embedded with a small MLP and concatenated with the MSA representations as additional sequence rows (see Suppl. Fig. 1 or lines 7-8 in Algorithm 2). These additional rows are participating in all MSA stack operations, but not in the masked MSA loss (subsubsection 1.9.9). While template torsion angle and MSA features are conceptually different quantities, they are embedded with different sets of weights, thus the learning process presumably drives the embeddings to be comparable since they are processed by the same downstream modules with the same weights.

Algorithm 16 Template pair stack

def TemplatePairStack($\{\mathbf{t}_{ij}\}, N_{\text{block}} = 2$) :

for all $l \in [1, \ldots, N_{\text{block}}]$ **do**
  $\{\mathbf{t}_{ij}\}$ += $\text{DropoutRowwise}_{0.25}(\text{TriangleAttentionStartingNode}(\{\mathbf{t}_{ij}\}, c = 64, N_{\text{head}} = 4))$
  $\{\mathbf{t}_{ij}\}$ += $\text{DropoutColumnwise}_{0.25}(\text{TriangleAttentionEndingNode}(\{\mathbf{t}_{ij}\}, c = 64, N_{\text{head}} = 4))$
  $\{\mathbf{t}_{ij}\}$ += $\text{DropoutRowwise}_{0.25}(\text{TriangleMultiplicationOutgoing}(\{\mathbf{t}_{ij}\}, c = 64))$
  $\{\mathbf{t}_{ij}\}$ += $\text{DropoutRowwise}_{0.25}(\text{TriangleMultiplicationIncoming}(\{\mathbf{t}_{ij}\}, c = 64))$
  $\{\mathbf{t}_{ij}\}$ += $\text{PairTransition}(\{\mathbf{t}_{ij}\}, n = 2)$
end for
return $\text{LayerNorm}\left(\{\mathbf{t}_{ij}\}\right)$

Algorithm 17 Template pointwise attention

def TemplatePointwiseAttention($\{\mathbf{t}_{s_t ij}\}, \{\mathbf{z}_{ij}\}, c = 64, N_{\text{head}} = 4$) :

$\mathbf{q}^h_{ij} = \text{LinearNoBias}(\mathbf{z}_{ij})$ $\quad \mathbf{q}^h_{ij} \in \mathbb{R}^c,\ h \in \{1, \ldots, N_{\text{head}}\}$

$\mathbf{k}^h_{s_t ij}, \mathbf{v}^h_{s_t ij} = \text{LinearNoBias}(\mathbf{t}_{s_t ij})$ $\quad \mathbf{k}^h_{s_t ij}, \mathbf{v}^h_{s_t ij} \in \mathbb{R}^c$

$a^h_{s_t ij} = \text{softmax}_{s_t}\left(\frac{1}{\sqrt{c}}\, {\mathbf{q}^h_{ij}}^\top \mathbf{k}^h_{s_t ij}\right)$

$\mathbf{o}^h_{ij} = \sum_{s_t} a^h_{s_t ij} \mathbf{v}^h_{s_t ij}$

$\{\tilde{\mathbf{z}}_{ij}\} = \text{Linear}\left(\text{concat}_h(\{\mathbf{o}^h_{ij}\})\right)$ $\quad \tilde{\mathbf{z}}_{ij} \in \mathbb{R}^{c_z}$

return $\{\tilde{\mathbf{z}}_{ij}\}$

1.7.2 Unclustered MSA stack

Unclustered MSA sequence features are linearly projected to form initial representations $\{\mathbf{e}_{s_e i}\}$ with $\mathbf{e}_{s_e i} \in \mathbb{R}^{c_e}$, $c_e = 64$, and $s_e \in \{1 \ldots N_{\text{extra\_seq}}\}$, $i \in \{1 \ldots N_{\text{res}}\}$. These representations are processed with the Extra MSA stack containing 4 blocks (Algorithm 18). They are highly similar to the main Evoformer blocks, with the notable difference of using global column-wise self-attention (Algorithm 19) and smaller representation sizes to allow processing of the large number of sequences. The final pair representations are used as inputs for the main Evoformer stack (Algorithm 6), while the final MSA activations are unused (see Suppl. Fig. 1).

Algorithm 18 Extra MSA stack

def ExtraMsaStack($\{\mathbf{e}_{s_e i}\}, \{\mathbf{z}_{ij}\}, N_{\text{block}} = 4$) :

for all $l \in [1, \ldots, N_{\text{block}}]$ **do**

 # MSA stack

$\{\mathbf{e}_{s_e i}\}$ += DropoutRowwise$_{0.15}$(MSARowAttentionWithPairBias($\{\mathbf{e}_{s_e i}\}, \{\mathbf{z}_{ij}\}, c = 8$))

$\{\mathbf{e}_{s_e i}\}$ += MSAColumn Global Attention($\{\mathbf{e}_{s_e i}\}$)

$\{\mathbf{e}_{s_e i}\}$ += MSATransition($\{\mathbf{e}_{s_e i}\}$)

 # Communication

$\{\mathbf{z}_{ij}\}$ += OuterProductMean($\{\mathbf{e}_{s_e i}\}$)

 # Pair stack

$\{\mathbf{z}_{ij}\}$ += DropoutRowwise$_{0.25}$(TriangleMultiplicationOutgoing($\{\mathbf{z}_{ij}\}$))

$\{\mathbf{z}_{ij}\}$ += DropoutRowwise$_{0.25}$(TriangleMultiplicationIncoming($\{\mathbf{z}_{ij}\}$))

$\{\mathbf{z}_{ij}\}$ += DropoutRowwise$_{0.25}$(TriangleAttentionStartingNode($\{\mathbf{z}_{ij}\}$))

$\{\mathbf{z}_{ij}\}$ += DropoutColumnwise$_{0.25}$(TriangleAttentionEndingNode($\{\mathbf{z}_{ij}\}$))

$\{\mathbf{z}_{ij}\}$ += PairTransition($\{\mathbf{z}_{ij}\}$)

end for

return $\{\mathbf{z}_{ij}\}$

Algorithm 19 MSA global column-wise gated self-attention

def MSAColumnGlobalAttention($\{\mathbf{m}_{si}\}, c = 8, N_{\text{head}} = 8$) :

 # Input projections

$\mathbf{m}_{si} \leftarrow \text{LayerNorm}(\mathbf{m}_{si})$

$\mathbf{q}^h_{si}, \mathbf{k}_{si}, \mathbf{v}_{si} = \text{LinearNoBias}(\mathbf{m}_{si})$ $\qquad \mathbf{q}^h_{si}, \mathbf{k}_{si}, \mathbf{v}_{si} \in \mathbb{R}^c,\ h \in \{1, \ldots, N_{\text{head}}\}$

$\mathbf{q}^h_i = \text{mean}_s\, \mathbf{q}^h_{si}$

$\mathbf{g}^h_{si} = \text{sigmoid}\left(\text{Linear}(\mathbf{m}_{si})\right)$ $\qquad \mathbf{g}^h_{si} \in \mathbb{R}^c$

 # Attention

$a^h_{ti} = \text{softmax}_t\left(\frac{1}{\sqrt{c}}\, \mathbf{q}^{h\top}_i \mathbf{k}_{ti}\right)$

$\mathbf{o}^h_{si} = \mathbf{g}^h_{si} \odot \sum_t a^h_{ti} \mathbf{v}_{ti}$

 # Output projection

$\tilde{\mathbf{m}}_{si} = \text{Linear}\left(\text{concat}_h\left(\mathbf{o}^h_{si}\right)\right)$ $\qquad \tilde{\mathbf{m}}_{si} \in \mathbb{R}^{c_m}$

return $\{\tilde{\mathbf{m}}_{si}\}$

1.8 Structure module

The Structure Module (Algorithm 20 and **Fig. 3d** in the main article) maps the abstract representation of the protein structure (created by the Evoformer stack) to concrete 3D atom coordinates. Evoformer's single representation is used as the initial single representation $\{\mathbf{s}_i^{\text{initial}}\}$ with $\mathbf{s}_i^{\text{initial}} \in \mathbb{R}^{c_s}$ and Evoformer's pair representation $\{\mathbf{z}_{ij}\}$ with $\mathbf{z}_{ij} \in \mathbb{R}^{c_z}$ and $i, j \in \{1, ..., N_{\text{res}}\}$ biases the affinity maps in the attention operations. The module has 8 layers with shared weights. Each layer updates the abstract single representation $\{\mathbf{s}_i\}$ as well the concrete 3D representation (the "residue gas") which is encoded as one backbone frame per residue $\{T_i\}$. We represent frames via a tuple $T_i := (R_i, \vec{\mathbf{t}}_i)$. The tuple represents an Euclidean transform from the local frame to a global reference frame. I.e. it transforms a position in local coordinates $\vec{\mathbf{x}}_{\text{local}} \in \mathbb{R}^3$ to a position in global coordinates $\vec{\mathbf{x}}_{\text{global}} \in \mathbb{R}^3$ as

$$\begin{aligned} \vec{\mathbf{x}}_{\text{global}} &= T_i \circ \vec{\mathbf{x}}_{\text{local}} \\ &= R_i \vec{\mathbf{x}}_{\text{local}} + \vec{\mathbf{t}}_i \ . \end{aligned} \tag{2}$$

The backbone frames are initialized to an identity transform (Algorithm 20 line 4). We call this approach the 'black hole initialization'. This initialization means that at the start of the Structure module all residues are located at the same point (the origin of the global frame) with the same orientation. One "layer" of the structure module is composed by the following operations: First, the abstract single representation is updated by the Invariant Point Attention (Algorithm 20 line 6) (see subsubsection 1.8.2 for details) and a transition layer. Then the abstract single representation is mapped to concrete update frames that are composed to the backbone frames (Algorithm 20 line 10). The composition of two Euclidean transforms is denoted as

$$T_{\text{result}} = T_1 \circ T_2$$

In the parameterization with a rotation matrix and translation vector this is:

$$\begin{aligned} (R_{\text{result}}, \vec{\mathbf{t}}_{\text{result}}) &= (R_1, \vec{\mathbf{t}}_1) \circ (R_2, \vec{\mathbf{t}}_2) \\ &= (R_1 R_2, \ \ R_1 \vec{\mathbf{t}}_2 + \vec{\mathbf{t}}_1) \end{aligned}$$

To obtain all atom coordinates, we parameterize each residue by torsion angles. I.e., the torsion angles are the only degrees of freedom, while all bond angles and bond lengths are fully rigid. The individual atoms are identified by their names $\mathcal{S}_{\text{atom names}} = \{\text{N}, \text{C}^\alpha, \text{C}, \text{O}, \text{C}^\beta, \text{C}^\gamma, \text{C}^{\gamma 1}, \text{C}^{\gamma 2}, \dots\}$. The torsions are named as $\mathcal{S}_{\text{torsion names}} = \{\omega, \phi, \psi, \chi_1, \chi_2, \chi_3, \chi_4\}$ We group the atoms according to their dependence on the torsion angles into "rigid groups" (Table 2). E.g., the position of atoms in the χ_2-group depend on χ_1 and χ_2, but do not depend on χ_3 or χ_4. We define a rigid group for every of the 3 backbone and 4 side chain torsion angles $\omega, \phi, \psi, \chi_1, \chi_2, \chi_3, \chi_4$, though with our construction of the backbone frames, no heavy atoms positions depend on the torsion angles ω or ϕ. The χ_5 angle is not included, because it only appears in arginine and only slightly varies around 0 degrees. Atoms that just depend on the backbone frame are in the "backbone rigid group", denoted as "bb".

Table 2 | Rigid groups for constructing all atoms from given torsion angles. Boxes highlight groups that are symmetric under 180° rotations.

aatype	bb	ψ	χ_1	χ_2	χ_3	χ_4
ALA	N, C^α, C, C^β	O	-	-	-	-
ARG	N, C^α, C, C^β	O	C^γ	C^δ	N^ϵ	$N^{\eta 1}$, $N^{\eta 2}$, C^ζ
ASN	N, C^α, C, C^β	O	C^γ	$N^{\delta 2}$, $O^{\delta 1}$	-	-
ASP	N, C^α, C, C^β	O	C^γ	$\boxed{O^{\delta 1}, O^{\delta 2}}$	-	-
CYS	N, C^α, C, C^β	O	S^γ	-	-	-
GLN	N, C^α, C, C^β	O	C^γ	C^δ	$N^{\epsilon 2}$, $O^{\epsilon 1}$	-
GLU	N, C^α, C, C^β	O	C^γ	C^δ	$\boxed{O^{\epsilon 1}, O^{\epsilon 2}}$	-
GLY	N, C^α, C	O	-	-	-	-
HIS	N, C^α, C, C^β	O	C^γ	$C^{\delta 2}$, $N^{\delta 1}$, $C^{\epsilon 1}$, $N^{\epsilon 2}$	-	-
ILE	N, C^α, C, C^β	O	$C^{\gamma 1}$, $C^{\gamma 2}$	$C^{\delta 1}$	-	-
LEU	N, C^α, C, C^β	O	C^γ	$C^{\delta 1}$, $C^{\delta 2}$	-	-
LYS	N, C^α, C, C^β	O	C^γ	C^δ	C^ϵ	N^ζ
MET	N, C^α, C, C^β	O	C^γ	S^δ	C^ϵ	-
PHE	N, C^α, C, C^β	O	C^γ	$\boxed{C^{\delta 1}, C^{\delta 2}, C^{\epsilon 1}, C^{\epsilon 2}, C^\zeta}$	-	-
PRO	N, C^α, C, C^β	O	C^γ	C^δ	-	-
SER	N, C^α, C, C^β	O	O^γ	-	-	-
THR	N, C^α, C, C^β	O	$C^{\gamma 2}$, $O^{\gamma 1}$	-	-	-
TRP	N, C^α, C, C^β	O	C^γ	$C^{\delta 1}$, $C^{\delta 2}$, $C^{\epsilon 2}$, $C^{\epsilon 3}$, $N^{\epsilon 1}$, $C^{\eta 2}$, $C^{\zeta 2}$, $C^{\zeta 3}$	-	-
TYR	N, C^α, C, C^β	O	C^γ	$\boxed{C^{\delta 1}, C^{\delta 2}, C^{\epsilon 1}, C^{\epsilon 2}, O^\eta, C^\zeta}$	-	-
VAL	N, C^α, C, C^β	O	$C^{\gamma 1}$, $C^{\gamma 2}$	-	-	-

A shallow ResNet (Algorithm 20 line 11) predicts the torsion angles $\vec{\alpha}_i^f \in \mathbb{R}^2$. They get mapped to points on the unit circle via normalization whenever they are used as angles. Furthermore we introduce a small auxiliary loss (implemented in Algorithm 27) that encourages a unit norm of the raw vector to avoid degenerate values. In contrast to a $[0, 2\pi]$ angle representation this representation has no discontinuity and can be directly used to construct rotation matrices without the need for trigonometric functions. The predicted torsion angles are converted to frames for the rigid groups of atoms (see subsubsection 1.8.4 for details).

During training, the last step of each layer computes auxiliary losses for the current 3D structure (Algorithm 20 line 17). The intermediate FAPE loss operates only on the backbone frames and C^α atom positions to keep the computational costs low. For the same reason the side chain are here only supervised by their torsion angles. The 180° rotation symmetry of some rigid groups (highlighted by boxes in Table 2) is addressed by providing the alternative angle $\vec{\alpha}_i^{\text{alt truth},f}$ as well (see subsubsection 1.9.1 for details).

We found it helpful to zero the gradients into the orientation component of the rigid bodies between iterations (Algorithm 20 line 20), so any iteration is optimized to find an optimal orientation for the structure in the current iteration, but is not concerned by having an orientation more suitable for the next iteration. Empirically, this improves the stability of training, presumably by removing the lever effects arising in a chained composition frames. After the 8 layers, the final backbone frames and torsion angles are mapped to frames for all rigid groups (backbone and side chain) T_i^f and all atom coordinates $\vec{\mathbf{x}}_i^a$ (Algorithm 20 line 24).

During training, the predicted frames and atom coordinates are compared against the ground truth by the FAPE loss (Algorithm 20 line 28) that assesses all atom coordinates (backbone and side chain) relative to all rigid groups. The 180°-rotation-symmetries of a few rigid groups are handled by a globally consistent renaming of the ambiguous atoms in the ground truth structure (see subsubsection 1.8.5 for details).

Finally the model predicts its confidence in form of a predicted lDDT-Cα score (pLDDT) per residue. This score is trained with the true per-residue lDDT-Cα score computed from the predicted structure and the ground truth. For details see subsubsection 1.9.6.

Algorithm 20 Structure module

def StructureModule $\Big(\{\mathbf{s}_i^{\text{initial}}\}, \{\mathbf{z}_{ij}\}, N_{\text{layer}} = 8, c = 128,$ $\mathbf{s}_i^{\text{initial}} \in \mathbb{R}^{c_s}$
$\{T_i^{\text{true},f}\}, \{T_i^{\text{alt truth},f}\}, \{\vec{\alpha}_i^{\text{true},f}\}, \{\vec{\alpha}_i^{\text{alt truth},f}\}, \{\vec{\mathbf{x}}_i^{\text{true},a}\}, \{\vec{\mathbf{x}}_i^{\text{alt truth},a}\}\Big)$:

$\mathbf{s}_i^{\text{initial}} \leftarrow \text{LayerNorm}(\mathbf{s}_i^{\text{initial}})$

$\mathbf{z}_{ij} \leftarrow \text{LayerNorm}(\mathbf{z}_{ij})$

$\mathbf{s}_i = \text{Linear}(\mathbf{s}_i^{\text{initial}})$ $\mathbf{s}_i \in \mathbb{R}^{c_s}$

$T_i = (\mathbf{I}, \vec{\mathbf{0}})$ $\mathbf{I} \in \mathbb{R}^{3\times3}, \vec{\mathbf{0}} \in \mathbb{R}^3$

for all $l \in [1, \ldots, N_{\text{layer}}]$ **do** *# shared weights*

  $\{\mathbf{s}_i\}$ += InvariantPointAttention$(\{\mathbf{s}_i\}, \{\mathbf{z}_{ij}\}, \{T_i\})$

  $\mathbf{s}_i \leftarrow \text{LayerNorm}(\text{Dropout}_{0.1}(\mathbf{s}_i))$

 # Transition.

  $\mathbf{s}_i \leftarrow \mathbf{s}_i + \text{Linear}(\text{relu}(\text{Linear}(\text{relu}(\text{Linear}(\mathbf{s}_i)))))$ all intermediate activations $\in \mathbb{R}^{c_s}$

  $\mathbf{s}_i \leftarrow \text{LayerNorm}(\text{Dropout}_{0.1}(\mathbf{s}_i))$

 # Update backbone.

  $T_i \leftarrow T_i \circ \text{BackboneUpdate}(\mathbf{s}_i)$

 # Predict side chain and backbone torsion angles $\omega, \phi, \psi, \chi_1, \chi_2, \chi_3, \chi_4$

  $\mathbf{a}_i = \text{Linear}(\mathbf{s}_i) + \text{Linear}(\mathbf{s}_i^{\text{initial}})$ $\mathbf{a}_i \in \mathbb{R}^c$

  $\mathbf{a}_i \leftarrow \mathbf{a}_i + \text{Linear}(\text{relu}(\text{Linear}(\text{relu}(\mathbf{a}_i))))$ all intermediate activations $\in \mathbb{R}^c$

  $\mathbf{a}_i \leftarrow \mathbf{a}_i + \text{Linear}(\text{relu}(\text{Linear}(\text{relu}(\mathbf{a}_i))))$ all intermediate activations $\in \mathbb{R}^c$

  $\vec{\alpha}_i^f = \text{Linear}(\text{relu}(\mathbf{a}_i))$ $\vec{\alpha}_i^f \in \mathbb{R}^2, f \in \mathcal{S}_{\text{torsion names}}$

 # Auxiliary losses in every iteration.

  $(R_i, \vec{\mathbf{t}}_i) = T_i$

  $\vec{\mathbf{x}}_i^{\text{C}\alpha} = \vec{\mathbf{t}}_i$

  $\mathcal{L}_{\text{aux}}^l = \Big(\text{computeFAPE}(\{T_i\}, \{\vec{\mathbf{x}}_i^{\text{C}\alpha}\}, \{T_i^{\text{true}}\}, \{\vec{\mathbf{x}}_i^{\text{true,C}\alpha}\}, \epsilon = 10^{-12})$

    $+ \text{torsionAngleLoss}(\{\vec{\alpha}_i^f\}, \{\vec{\alpha}_i^{\text{true},f}\}, \{\vec{\alpha}_i^{\text{alt truth},f}\})\Big)$

 # No rotation gradients between iterations to stabilize training.

  if $l < N_{\text{layer}}$ **then**

    $T_i \leftarrow (\text{stopgrad}(R_i), \vec{\mathbf{t}}_i)$

  end if

end for

$\mathcal{L}_{\text{aux}} = \text{mean}_l(\{\mathcal{L}_{\text{aux}}^l\})$

$T_i^f, \vec{\mathbf{x}}_i^a = \text{computeAllAtomCoordinates}(T_i, \vec{\alpha}_i^f)$ $a \in \mathcal{S}_{\text{atom names}}$

$T_i^f \leftarrow \text{concat}(T_i, T_i^f)$

 # Final loss on all atom coordinates.

$\{T_i^{\text{true},f}\}, \{\vec{\mathbf{x}}_i^{\text{true},a}\} \leftarrow \text{renameSymmetricGroundTruthAtoms}($

$\{T_i^f\}, \{\vec{\mathbf{x}}_i^a\}, \{T_i^{\text{true},f}\}, \{T_i^{\text{alt truth},f}\}, \{\vec{\mathbf{x}}_i^{\text{true},a}\}, \{\vec{\mathbf{x}}_i^{\text{alt truth},a}\})$

$\mathcal{L}_{\text{FAPE}} = \text{computeFAPE}(\{T_i^f\}, \{\vec{\mathbf{x}}_i^a\}, \{T_i^{\text{true},f}\}, \{\vec{\mathbf{x}}_i^{\text{true},a}\}, \epsilon = 10^{-4})$

 # Predict model confidence.

$\{r_i^{\text{true LDDT}}\} = \text{perResidueLDDT\_Ca}(\{\vec{\mathbf{x}}_i^a\}, \{\vec{\mathbf{x}}_i^{\text{true},a}\})$

$\{r_i^{\text{pLDDT}}\}, \mathcal{L}_{\text{conf}} = \text{predictPerResidueLDDT\_Ca}(\{\mathbf{s}_i\}, \{r_i^{\text{true LDDT}}\})$

return $\{\vec{\mathbf{x}}_i^a\}, \{r_i^{\text{pLDDT}}\}, \mathcal{L}_{\text{FAPE}}, \mathcal{L}_{\text{conf}}, \mathcal{L}_{\text{aux}}$

1.8.1 Construction of frames from ground truth atom positions

We construct frames using the position of three atoms from the ground truth PDB structures using a Gram–Schmidt process (Algorithm 21). Note that the translation vector $\vec{\mathbf{t}}$ is assigned to the centre atom $\vec{\mathbf{x}}_2$. For backbone frames, we use N as $\vec{\mathbf{x}}_1$, Cα as $\vec{\mathbf{x}}_2$ and C as $\vec{\mathbf{x}}_3$, so the frame has Cα at the centre. For the side chain frames, we use the atom before the torsion bond as $\vec{\mathbf{x}}_1$, the atom after the torsion bond as $\vec{\mathbf{x}}_2$ and the next atom after that as $\vec{\mathbf{x}}_3$.

Algorithm 21 Rigid from 3 points using the Gram–Schmidt process

def rigidFrom3Points$(\vec{\mathbf{x}}_1, \vec{\mathbf{x}}_2, \vec{\mathbf{x}}_3)$: $\quad \vec{\mathbf{x}}_1, \vec{\mathbf{x}}_2, \vec{\mathbf{x}}_3 \in \mathbb{R}^3$

$\vec{\mathbf{v}}_1 = \vec{\mathbf{x}}_3 - \vec{\mathbf{x}}_2$

$\vec{\mathbf{v}}_2 = \vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_2$

$\vec{\mathbf{e}}_1 = \vec{\mathbf{v}}_1 / \left\|\vec{\mathbf{v}}_1\right\|$

$\vec{\mathbf{u}}_2 = \vec{\mathbf{v}}_2 - \vec{\mathbf{e}}_1 \left(\vec{\mathbf{e}}_1^\top \vec{\mathbf{v}}_2\right)$

$\vec{\mathbf{e}}_2 = \vec{\mathbf{u}}_2 / \left\|\vec{\mathbf{u}}_2\right\|$

$\vec{\mathbf{e}}_3 = \vec{\mathbf{e}}_1 \times \vec{\mathbf{e}}_2$

$R = \text{concat}(\vec{\mathbf{e}}_1, \vec{\mathbf{e}}_2, \vec{\mathbf{e}}_3)$ $\quad R \in \mathbb{R}^{3\times 3}$

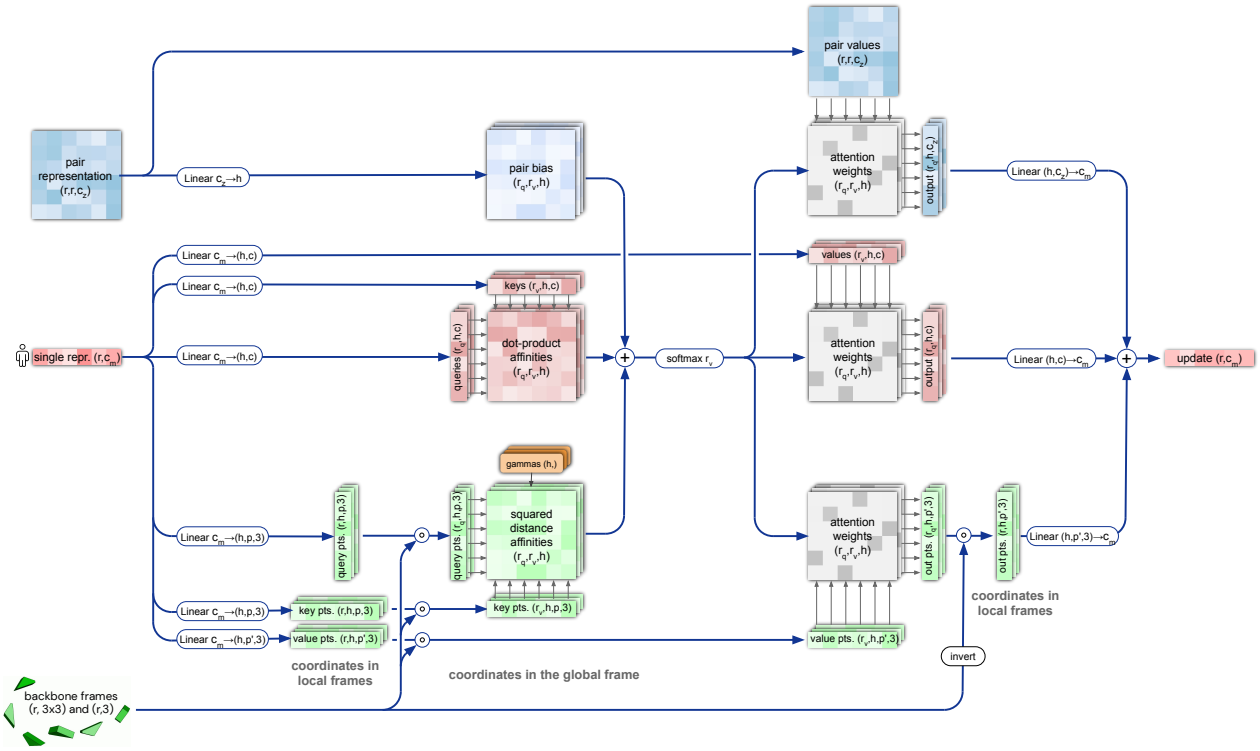
$\vec{\mathbf{t}} = \vec{\mathbf{x}}_2$

return $(R, \vec{\mathbf{t}})$

1.8.2 Invariant point attention (IPA)

Invariant Point Attention (IPA) (Suppl. Fig. 8 and Algorithm 22) is a form of attention that acts on a set of frames (parametrized as Euclidean transforms T_i) and is invariant under global Euclidean transformations T_{global} on said frames. We represent all the coordinates within IPA in nanometres; the choice of units affects the scaling of the point component of the attention affinities.

To define the initial weightings for the different terms, we assume that all queries and keys come iid from a unit normal distribution $N(0, 1)$ and compute the variances of the attention logits. Each scalar pair q, k contributes $\text{Var}[qk] = 1$. Each point pair $(\vec{\mathbf{q}}, \vec{\mathbf{k}})$ contributes $\text{Var}[0.5 \left\|\vec{\mathbf{q}}\right\|^2 - \vec{\mathbf{q}}^\top \vec{\mathbf{k}}] = 9/2$. The weighting factors w_L and w_C are computed such that all three terms contribute equally and that the resulting variance is 1. The weight per head $\gamma^h \in \mathbb{R}$ is the softplus of a learnable scalar.



Supplementary Figure 8 | Invariant Point Attention Module. (**top, blue arrays**) modulation by the pair representation. (**middle, red arrays**) standard attention on abstract features. (**bottom, green arrays**) Invariant point attention. Dimensions: r: residues, c: channels, h: heads, p: points.

Algorithm 22 Invariant point attention (IPA)

def $\text{InvariantPointAttention}(\{\mathbf{s}_i\}, \{\mathbf{z}_{ij}\}, \{T_i\}, N_{\text{head}} = 12, c = 16, N_{\text{query points}} = 4, N_{\text{point values}} = 8)$:

1: $\mathbf{q}_i^h, \mathbf{k}_i^h, \mathbf{v}_i^h = \text{LinearNoBias}(\mathbf{s}_i)$ $\qquad \mathbf{q}_i^h, \mathbf{k}_i^h, \mathbf{v}_i^h \in \mathbb{R}^c,\ h \in \{1, \dots, N_{\text{head}}\}$

2: $\vec{\mathbf{q}}_i^{hp}, \vec{\mathbf{k}}_i^{hp} = \text{LinearNoBias}(\mathbf{s}_i)$ $\qquad \vec{\mathbf{q}}_i^{hp}, \vec{\mathbf{k}}_i^{hp}, \in \mathbb{R}^3,\ p \in \{1, \dots, N_{\text{query points}}\},$ units: nanometres

3: $\vec{\mathbf{v}}_i^{hp} = \text{LinearNoBias}(\mathbf{s}_i)$ $\qquad \vec{\mathbf{v}}_i^{hp} \in \mathbb{R}^3,\ p \in \{1, \dots, N_{\text{point values}}\},$ units: nanometres

4: $b_{ij}^h = \text{LinearNoBias}(\mathbf{z}_{ij})$

5: $w_C = \sqrt{\frac{2}{9N_{\text{query points}}}},$

6: $w_L = \sqrt{\frac{1}{3}}$

7: $a_{ij}^h = \text{softmax}_j\left(w_L\left(\frac{1}{\sqrt{c}}\,\mathbf{q}_i^{h\top}\mathbf{k}_j^h \;+ b_{ij}^h \;- \frac{\gamma^h w_C}{2}\sum_p \left\|T_i \circ \vec{\mathbf{q}}_i^{hp} - T_j \circ \vec{\mathbf{k}}_j^{hp}\right\|^2\right)\right)$

8: $\tilde{\mathbf{o}}_i^h = \sum_j a_{ij}^h \mathbf{z}_{ij}$

9: $\mathbf{o}_i^h = \sum_j a_{ij}^h \mathbf{v}_j^h$

10: $\vec{\mathbf{o}}_i^{hp} = T_i^{-1} \circ \sum_j a_{ij}^h \left(T_j \circ \vec{\mathbf{v}}_j^{hp}\right)$

11: $\tilde{\mathbf{s}}_i = \text{Linear}\left(\text{concat}_{h,p}(\tilde{\mathbf{o}}_i^h, \mathbf{o}_i^h, \vec{\mathbf{o}}_i^{hp}, \left\|\vec{\mathbf{o}}_i^{hp}\right\|)\right)$

12: **return** $\{\tilde{\mathbf{s}}_i\}$

The proof for invariance is straight-forward: The global transformation cancels out in the affinity computation (Algorithm 22 line 7), because the L2-norm of a vector is invariant under rigid transformations:

$$\left\|\left(T_{\text{global}} \circ T_i\right) \circ \vec{\mathbf{q}}_i^{hp} - \left(T_{\text{global}} \circ T_j\right) \circ \vec{\mathbf{k}}_j^{hp}\right\|^2 = \left\|T_{\text{global}} \circ \left(T_i \circ \vec{\mathbf{q}}_i^{hp} - T_j \circ \vec{\mathbf{k}}_j^{hp}\right)\right\|^2 \tag{3}$$

$$= \left\|T_i \circ \vec{\mathbf{q}}_i^{hp} - T_j \circ \vec{\mathbf{k}}_j^{hp}\right\|^2 . \tag{4}$$

In the computation of the output points (Algorithm 22 line 10) it cancels out when mapping back to the local frame:

$$\left(T_{\text{global}} \circ T_i\right)^{-1} \circ \sum_j a_{ij}^h \left(\left(T_{\text{global}} \circ T_j\right) \circ \vec{\mathbf{v}}_j^{hp}\right) = T_i^{-1} \circ T_{\text{global}}^{-1} \circ T_{\text{global}} \circ \sum_j a_{ij}^h \left(T_j \circ \vec{\mathbf{v}}_j^{hp}\right) \tag{5}$$

$$= T_i^{-1} \circ \sum_j a_{ij}^h \left(T_j \circ \vec{\mathbf{v}}_j^{hp}\right) \tag{6}$$

The invariance with respect to the global reference frame in turn implies that applying a shared rigid motion to all residues, while keeping the embeddings fixed, will lead to the same update in the local frames. Therefore, the updated structure will be transformed by the same shared rigid motion showing that this update rule is equivariant under rigid motions. Here and elsewhere, "rigid motion" includes proper rotation and translation but not reflection.

1.8.3 Backbone update

The updates for the backbone frames are created by predicting a quaternion for the rotation and a vector for the translation (Algorithm 23). The first component of the non-unit quaternion is fixed to 1. The three components

defining the Euler axis are predicted by the network. This procedure guarantees a valid normalized quaternion and furthermore favours small rotations over large rotations (quaternion (1,0,0,0) is the identity rotation).

Algorithm 23 Backbone update

def BackboneUpdate($\mathbf{s}_i$) :

1: $b_i, c_i, d_i, \vec{\mathbf{t}}_i = \text{Linear}(\mathbf{s}_i)$ $\qquad b_i, c_i, d_i \in \mathbb{R},\ \vec{\mathbf{t}}_i \in \mathbb{R}^3$

 # Convert (non-unit) quaternion to rotation matrix.

2: $(a_i, b_i, c_i, d_i) \leftarrow (1, b_i, c_i, d_i)/\sqrt{1 + b_i^2 + c_i^2, +d_i^2}$

3: $R_i = \begin{pmatrix} a_i^2 + b_i^2 - c_i^2 - d_i^2 & 2b_ic_i - 2a_id_i & 2b_id_i + 2a_ic_i \\ 2b_ic_i + 2a_id_i & a_i^2 - b_i^2 + c_i^2 - d_i^2 & 2c_id_i - 2a_ib_i \\ 2b_id_i - 2a_ic_i & 2c_id_i + 2a_ib_i & a_i^2 - b_i^2 - c_i^2 + d_i^2 \end{pmatrix}$

4: $T_i = (R_i, \vec{\mathbf{t}}_i)$

5: **return** T_i

1.8.4 Compute all atom coordinates

The Structure Module predicts backbone frames T_i and torsion angles $\vec{\alpha}_i^f$. The atom coordinates are then constructed by applying the torsion angles to the corresponding amino acid structure with idealized bond angles and bond lengths. We attach a local frame to each rigid group (see Table 2), such that the torsion axis is the x-axis, and store the ideal literature atom coordinates [98] for each amino acid relative to these frames in a table $\vec{\mathbf{x}}^{\text{lit}}_{r,f,a}$, where $r \in \{\text{ALA}, \text{ARG}, \text{ASN}, \dots\}$ denotes the residue type, $f \in \mathcal{S}_{\text{torsion names}}$ denotes the frame and a the atom name. We further pre-compute rigid transformations that transform atom coordinates from each frame to the frame that is higher up in the hierarchy. E.g. $T^{\text{lit}}_{r,(\chi_2 \to \chi_1)}$ maps atoms in amino-acid type r from the χ_2-frame to the χ_1-frame. As we are only predicting heavy atoms, the extra backbone rigid groups ω and ϕ do not contain atoms, but the corresponding frames contribute to the FAPE loss for alignment to the ground truth (like all other frames).

Algorithm 24 Compute all atom coordinates

def computeAllAtomCoordinates($T_i, \vec{\alpha}_i^f, F_i^{\text{aatype}}$) :

$\hat{\vec{\alpha}}_i^f = \vec{\alpha}_i^f / \|\vec{\alpha}_i^f\|$

$(\vec{\omega}_i, \vec{\phi}_i, \vec{\psi}_i, \vec{\chi}_{1_i}, \vec{\chi}_{2_i}, \vec{\chi}_{3_i}, \vec{\chi}_{4_i}) = \hat{\vec{\alpha}}_i^f$

 # Make extra backbone frames.

$r_i = F_i^{\text{aatype}}$

$T_{i1} = T_i \circ T^{\text{lit}}_{r_i,(\omega\to\text{bb})} \circ \text{makeRotX}(\vec{\omega}_i)$

$T_{i2} = T_i \circ T^{\text{lit}}_{r_i,(\phi\to\text{bb})} \circ \text{makeRotX}(\vec{\phi}_i)$

$T_{i3} = T_i \circ T^{\text{lit}}_{r_i,(\psi\to\text{bb})} \circ \text{makeRotX}(\vec{\psi}_i)$

 # Make side chain frames (chain them up along the side chain).

$T_{i4} = T_i \circ T^{\text{lit}}_{r_i,(\chi_1\to\text{bb})} \circ \text{makeRotX}(\vec{\chi}_{1_i})$

$T_{i5} = T_{i4} \circ T^{\text{lit}}_{r_i,(\chi_2\to\chi_1)} \circ \text{makeRotX}(\vec{\chi}_{2_i})$

$T_{i6} = T_{i5} \circ T^{\text{lit}}_{r_i,(\chi_3\to\chi_2)} \circ \text{makeRotX}(\vec{\chi}_{3_i})$

$T_{i7} = T_{i6} \circ T^{\text{lit}}_{r_i,(\chi_4\to\chi_3)} \circ \text{makeRotX}(\vec{\chi}_{4_i})$

 # Map atom literature positions to the global frame.

$\vec{\mathbf{x}}_i^a = \text{concat}_{f,a'}\left(\{T_i^f \circ \vec{\mathbf{x}}^{\text{lit}}_{r_i,f,a'}\}\right)$

return $T_i^f, \vec{\mathbf{x}}_i^a$

Algorithm 25 Make a transformation that rotates around the x-axis

def makeRotX($\vec{\alpha}$) : $\quad \vec{\alpha} \in R^2$ with $\|\vec{\alpha}\| = 1$

$$R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \alpha_1 & -\alpha_2 \\ 0 & \alpha_2 & \alpha_1 \end{pmatrix}$$

$$\vec{\mathbf{t}} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$T = (R, \vec{\mathbf{t}})$

return T

1.8.5 Rename symmetric ground truth atoms

The 180°-rotation-symmetry for some of the rigid groups (see Table 2) leads to naming ambiguities for the atoms in this group for all atoms that are not on the rotation axis (see Table 3).

Table 3 | Ambiguous atom names due to $180°$-rotation-symmetry for some of the rigid groups

aatype	renaming swaps
ASP	$O^{\delta 1} \leftrightarrow O^{\delta 2}$
GLU	$O^{\epsilon 1} \leftrightarrow O^{\epsilon 2}$
PHE	$C^{\delta 1} \leftrightarrow C^{\delta 2}$, $C^{\epsilon 1} \leftrightarrow C^{\epsilon 2}$
TYR	$C^{\delta 1} \leftrightarrow C^{\delta 2}$, $C^{\epsilon 1} \leftrightarrow C^{\epsilon 2}$

Algorithm 26 resolves the naming ambiguities in a globally consistent way by renaming the ground truth structure: For each residue, it computes the lDDT of the atoms against all non-ambiguous atoms for both possible namings ("true" and "alt truth") of the ground truth atoms. The set of non-ambiguous atoms $\mathcal{S}_{\text{non-ambiguous atoms}}$ is the set of all (residue-type, atom-type) tuples from Table 2 minus the set of ambiguous atoms from Table 3. Subsequently the algorithm renames the ambiguous ground truth atoms such that they fit best to the predicted structure.

Algorithm 26 Rename symmetric ground truth atoms

def renameSymmetricGroundTruthAtoms($\{T_i^f\}, \{\vec{\mathbf{x}}_i^a\}, \{T_i^{\text{true},f}\}, \{T_i^{\text{alt truth},f}\}, \{\vec{\mathbf{x}}_i^{\text{true},a}\}, \{\vec{\mathbf{x}}_i^{\text{alt truth},a}\}$) :

$d_{(i,a),(j,b)} = \left\| \vec{\mathbf{x}}_i^a - \vec{\mathbf{x}}_j^b \right\|$ $\quad \forall (j,b) \in \mathcal{S}_{\text{non-ambiguous atoms}}$

$d^{\text{true}}_{(i,a),(j,b)} = \left\| \vec{\mathbf{x}}_i^{\text{true},a} - \vec{\mathbf{x}}_j^{\text{true},b} \right\|$ $\quad \forall (j,b) \in \mathcal{S}_{\text{non-ambiguous atoms}}$

$d^{\text{alt truth}}_{(i,a),(j,b)} = \left\| \vec{\mathbf{x}}_i^{\text{alt truth},a} - \vec{\mathbf{x}}_j^{\text{true},b} \right\|$ $\quad \forall (j,b) \in \mathcal{S}_{\text{non-ambiguous atoms}}$

for all $i \in [1 \dots N_{\text{res}}]$ **do**

  if $\text{sum}_{a,(j,b)} \left(\left| d_{(i,a),(j,b)} - d^{\text{alt truth}}_{(i,a),(j,b)} \right| \right) < \text{sum}_{a,(j,b)} \left(\left| d_{(i,a),(j,b)} - d^{\text{true}}_{(i,a),(j,b)} \right| \right)$ **then**

    $\vec{\mathbf{x}}_i^{\text{true},a} \leftarrow \vec{\mathbf{x}}_i^{\text{alt truth},a}$

    $T_i^{\text{true},f} \leftarrow T_i^{\text{alt truth},f}$

  end if

end for

return $\{T_i^{\text{true},f}\}, \{\vec{\mathbf{x}}_i^{\text{true},a}\}$

1.8.6 Amber relaxation

In order to resolve any remaining structural violations and clashes [99], we relax our model predictions by an iterative restrained energy minimization procedure. At each round, we perform minimization of the AMBER99SB [100] force field with additional harmonic restraints that keep the system near its input structure. The restraints are applied independently to heavy atoms, with a spring constant of 10 kcal/mol Å^2. Once the minimizer has converged, we determine which residues still contain violations. We then remove restraints from all atoms within these residues and perform restrained minimization once again, starting from the minimized structure of the previous iteration. This process is repeated until all violations are resolved. In the CASP14 assessment we used a single iteration; targets with unresolved violations were re-run.

Full energy minimization and hydrogen placement was performed using the OpenMM simulation package [101]. The minimization procedure was run with a tolerance of 2.39 kcal/mol and an unbounded maximum number of steps, which are OpenMM's default values.

1.9 Loss functions and auxiliary heads

The network is trained end-to-end, with gradients coming from the main Frame Aligned Point Error (FAPE) loss $\mathcal{L}_{\text{FAPE}}$ and a number of auxiliary losses. The total per-example loss can be defined as follows

$$\mathcal{L} = \begin{cases} 0.5\mathcal{L}_{\text{FAPE}} + 0.5\mathcal{L}_{\text{aux}} + 0.3\mathcal{L}_{\text{dist}} + 2.0\mathcal{L}_{\text{msa}} + 0.01\mathcal{L}_{\text{conf}} & \text{training} \\ 0.5\mathcal{L}_{\text{FAPE}} + 0.5\mathcal{L}_{\text{aux}} + 0.3\mathcal{L}_{\text{dist}} + 2.0\mathcal{L}_{\text{msa}} + 0.01\mathcal{L}_{\text{conf}} + 0.01\mathcal{L}_{\text{exp resolved}} + 1.0\mathcal{L}_{\text{viol}} & \text{fine-tuning} \end{cases}, \tag{7}$$

where $\mathcal{L}_{\text{aux}}$ is the auxiliary loss from the Structure Module (averaged FAPE and torsion losses on the intermediate structures, defined in Algorithm 20 line 23), $\mathcal{L}_{\text{dist}}$ is an averaged cross-entropy loss for distogram prediction, $\mathcal{L}_{\text{msa}}$ is an averaged cross-entropy loss for masked MSA prediction, $\mathcal{L}_{\text{conf}}$ is the model confidence loss defined in subsubsection 1.9.6, $\mathcal{L}_{\text{exp resolved}}$ is the "experimentally resolved" loss defined in subsubsection 1.9.10, and $\mathcal{L}_{\text{viol}}$ is the violation loss defined in subsubsection 1.9.11. The last two losses are only used during fine-tuning.

To decrease the relative importance of short sequences, we multiply the final loss of each training example by the square root of the number of residues after cropping. This implies equal weighting for all proteins that are longer than the crop size, and a square-root penalty for the shorter ones.

The purpose of the FAPE, aux, distogram, and MSA losses is to attach an individual loss to each major subcomponent of the model (including both the pair and MSA final embeddings) as a guide during the training of the "purpose" of each unit. The FAPE and aux are direct structural terms for the Structure module. The distogram loss assures that all entries in the pair representation have a clear relationship to the associated ij residue pair and to make sure that the pair representation will be useful for the structure module (ablations show that this is only a minor effect however). The distogram is also a distributional prediction, such that it is a method by which we interpret the model's confidence in interdomain interactions. The MSA loss is intended to force the network to consider inter-sequence or phylogenetic relationships to complete the BERT task, which we intend as a way to encourage the model to consider co-evolution-like relationships without explicitly encoding covariance statistics (this is the intention but we only observe the outcome that it increases model accuracy). The very small confidence loss allows the construction of the pLDDT value without compromising the accuracy of the structures themselves – we had previously fine-tuned this loss after training but it is just as accurate to train from the beginning with a small loss. Finally, the "violation" losses encourage the model to produce a physically plausible structure with correct bond geometry and avoidance of clashes, even in cases where the model is highly unsure of the structure. This avoids rare failures or accuracy loss in the final AMBER relaxation. Using the violation losses early in training causes a small drop in final accuracy since the model overly optimizes for the avoidance of clashes, so we only use this during fine-tuning.

The various loss weights were hand-selected and only lightly-tuned over the course of AlphaFold development (typically a couple of values for the coefficient of each loss was tried when the loss term was introduced and the weights rarely adjusted after that). We did some tuning early in model development on the ratio of FAPE, distogram, and MSA losses, but did not re-tune much as the model developed. It is possible that automated or more extensive tuning of these weights could improve accuracy, but we have generally not observed strong sensitivity to the precise values that would motivate us to do so.

Below we provide details of the individual losses and transformations that are applied to the Evoformer output representations to obtain auxiliary predictions.

1.9.1 Side chain and backbone torsion angle loss

The predicted side chain and backbone torsion angles are represented as points on the unit circle, i.e., $\hat{\vec{\alpha}}_i^f \in \mathbb{R}^2$ with $\|\hat{\vec{\alpha}}_i^f\| = 1$. They are compared to the true torsion angles $\vec{\alpha}_i^{\text{true},f}$ with an L2 loss in $\mathbb{R}^2$, which is mathematically equivalent to the cosine of the angle difference (see below).

Some side chains parts are $180°$-rotation-symmetric (see Table 2), such that the predicted torsion angle χ and $\chi + \pi$ result in the same physical structure (just the internal atom names are exchanged). We allow the

network to produce either of these torsion angles by providing the alternative angle as $\vec{\alpha}_i^{\text{alt truth},f} = \vec{\alpha}_i^{\text{true},f} + \pi$. For all non-symmetric configurations we set $\vec{\alpha}_i^{\text{alt truth},f} = \vec{\alpha}_i^{\text{true},f}$.

We also introduce a small auxiliary loss $\mathcal{L}_{\text{anglenorm}}$ that keeps the predicted points close to the unit circle. There are two reasons for this: One is to avoid vectors too close to the origin, which can lead to numerically unstable gradients. The other is that while the norm of the vector does not influence the output it does influence the learning dynamics of the network. When looking at how the gradients transform in the backward pass of the normalization the gradients will get rescaled by the norm of the unnormalized vector.

Given that the model is highly nonlinear the length of these vectors can change strongly during training, leading to undesirable learning dynamics. The weighting factor was selected on an ad hoc basis, testing a few values and picking the lowest one that keeps the norm of the vectors stable. We have not observed any strong dependence on the precise value in terms of model performance.

Algorithm 27 Side chain and backbone torsion angle loss

def $\text{torsionAngleLoss}(\{\vec{\alpha}_i^f\}, \{\vec{\alpha}_i^{\text{true},f}\}, \{\vec{\alpha}_i^{\text{alt truth},f}\})$:

1: $\ell_i^f = \|\vec{\alpha}_i^f\|$

2: $\hat{\vec{\alpha}}_i^f = \vec{\alpha}_i^f / \ell_i^f$

3: $\mathcal{L}_{\text{torsion}} = \text{mean}_{i,f}(\text{minimum}(\|\hat{\vec{\alpha}}_i^f - \vec{\alpha}_i^{\text{true},f}\|^2,\ \|\hat{\vec{\alpha}}_i^f - \vec{\alpha}_i^{\text{alt truth},f}\|^2))$

4: $\mathcal{L}_{\text{anglenorm}} = \text{mean}_{i,f}(\{|\ell_i^f - 1|\})$

5: **return** $\mathcal{L}_{\text{torsion}} + 0.02\mathcal{L}_{\text{anglenorm}}$

The comparison of two angles (α and β) represented as points on the unit circle by an L2 norm is mathematically equivalent to the cosine of the angle difference:

$$\begin{aligned}\cos(\alpha-\beta) &= \cos\alpha\cdot\cos\beta + \sin\alpha\cdot\sin\beta \\ &= \begin{pmatrix}\cos\alpha\\ \sin\alpha\end{pmatrix}^\top \begin{pmatrix}\cos\beta\\ \sin\beta\end{pmatrix} \qquad (8)\\ \left\|\begin{pmatrix}\cos\alpha\\ \sin\alpha\end{pmatrix} - \begin{pmatrix}\cos\beta\\ \sin\beta\end{pmatrix}\right\|^2 &= \left\|\begin{pmatrix}\cos\alpha\\ \sin\alpha\end{pmatrix}\right\|^2 + \left\|\begin{pmatrix}\cos\beta\\ \sin\beta\end{pmatrix}\right\|^2 - 2\begin{pmatrix}\cos\alpha\\ \sin\alpha\end{pmatrix}^\top \begin{pmatrix}\cos\beta\\ \sin\beta\end{pmatrix} \\ &= 2 - 2\begin{pmatrix}\cos\alpha\\ \sin\alpha\end{pmatrix}^\top \begin{pmatrix}\cos\beta\\ \sin\beta\end{pmatrix} \\ &= 2 - 2\cos(\alpha-\beta), \qquad (9)\end{aligned}$$

where the first identity is just the normal cosine difference formula.

1.9.2 Frame aligned point error (FAPE)

The Frame Aligned Point Error (FAPE) (see Algorithm 28, and **Fig. 3f** with its corresponding section in the main article) scores a set of predicted atom coordinates $\{\vec{\mathbf{x}}_j\}$ under a set of predicted local frames $\{T_i\}$ against the corresponding ground truth atom coordinates $\{\vec{\mathbf{x}}_j^{\text{true}}\}$ and ground truth local frames $\{T_i^{\text{true}}\}$. The final FAPE loss (Algorithm 20 line 28) scores all atoms in all backbone and side chain frames. Additionally a cheaper version (scoring only all $C\alpha$ atoms in all backbone frames) is used as an auxiliary loss in every layer of the Structure Module (Algorithm 20 line 17).

In order to formulate the loss we compute the atom position $\vec{\mathbf{x}}_j$ relative to frame T_i (Algorithm 28 line 1) and the location of the corresponding true atom position $\vec{\mathbf{x}}_j^{\text{true}}$ relative to the true frame T_i^{true} (Algorithm 28 line 2). The deviation is computed as a robust L2 norm (Algorithm 28 line 3) (ϵ is a small constant added to ensure that gradients are numerically well behaved for small differences. The exact value of this constant does not matter, as long as it is small enough. We used values of 10^{-4} and 10^{-12} in our experiments.). The

resulting $N_{\text{frames}} \times N_{\text{atoms}}$ deviations are penalized with a clamped L1-loss (Algorithm 28 line 4) with a length scale $Z = 10$ Å to make the loss unitless.

In this section, we represent the point positions and the hyperparameters in Å, although the loss is invariant to the choice of units.

We now discuss the behaviour of the loss under global rigid transformations of both the true and the predicted structure. Firstly we should note that $\vec{\mathbf{x}}_{ij}$ is invariant under rigid motions (not including reflection); as such if the predicted structure differs from the ground truth by an arbitrary rotation and translation the loss will stay constant. However, the loss is not invariant under reflections due to the way the local frames are constructed, as the z-axis of the local frames transforms as a pseudo-vector. Furthermore, we can see that the error is invariant when applying the same rigid motion to both target frames and predicted frames. This implies that the way frames are constructed from the structure is not constrained to the precise choice we made, but can differ as long as they are constructed in a consistent manner between predicted and target structure.

Algorithm 28 Compute the Frame aligned point error

def computeFAPE$(\{T_i\}, \{\vec{\mathbf{x}}_j\}, \{T_i^{\text{true}}\}, \{\vec{\mathbf{x}}_j^{\text{true}}\}, Z = 10\text{Å}, d_{\text{clamp}} = 10\text{Å}, \epsilon = 10^{-4}\text{Å}^2)$:

$T_i, T_i^{\text{true}} \in (\mathbb{R}^{3\times3}, \mathbb{R}^3)$
$\vec{\mathbf{x}}_j, \vec{\mathbf{x}}_j^{\text{true}} \in \mathbb{R}^3,$
$i \in \{1, ..., N_{\text{frames}}\}, j \in \{1, ..., N_{\text{atoms}}\}$

1: $\vec{\mathbf{x}}_{ij} = T_i^{-1} \circ \vec{\mathbf{x}}_j$ $\quad \vec{\mathbf{x}}_{ij} \in \mathbb{R}^3$

2: $\vec{\mathbf{x}}_{ij}^{\text{true}} = T_i^{\text{true}-1} \circ \vec{\mathbf{x}}_j^{\text{true}}$ $\quad \vec{\mathbf{x}}_{ij}^{\text{true}} \in \mathbb{R}^3$

3: $d_{ij} = \sqrt{\|\vec{\mathbf{x}}_{ij} - \vec{\mathbf{x}}_{ij}^{\text{true}}\|^2 + \epsilon}$ $\quad d_{ij} \in \mathbb{R}$

4: $\mathcal{L}_{\text{FAPE}} = \frac{1}{Z} \text{mean}_{i,j}(\text{minimum}(d_{\text{clamp}}, d_{ij}))$

5: **return** $\mathcal{L}_{\text{FAPE}}$

1.9.3 Chiral properties of AlphaFold and its loss

In this section we will look in detail at the transformation properties of the various components under global reflections

$$\vec{\mathbf{x}}_i \overset{\text{reflection}}{\mapsto} -\vec{\mathbf{x}}_i \tag{10}$$

Under this global reflection, the coordinates of the frames also change in a non-trivial way (use Algorithm 21 with negated positions). Simple algebra shows that

$$T_i = (R_i, \vec{\mathbf{t}}_i) \overset{\text{reflection}}{\mapsto} \left(R_i \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, -\vec{\mathbf{t}}_i \right), \tag{11}$$

where the non-trivial transformation of the rotation matrix R_i arises from the cross-product in Algorithm 21. The non-trivial transformation of the rotation matrix also means that the local points $T_i^{-1} \circ \vec{\mathbf{x}}_j$ are *not* invariant under reflection,

$$T_i^{-1} \circ \vec{\mathbf{x}}_j = R_i^{-1}(\vec{\mathbf{x}}_j - \vec{\mathbf{t}}_i) \tag{12}$$

$$\overset{\text{reflection}}{\mapsto} \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} R_i^{-1}(-\vec{\mathbf{x}}_j + \vec{\mathbf{t}}_i) \tag{13}$$

$$= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} R_i^{-1}(\vec{\mathbf{x}}_j - \vec{\mathbf{t}}_i) \tag{14}$$

$$= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \left(T_i^{-1} \circ \vec{\mathbf{x}}_j \right). \tag{15}$$

The effect of a global reflection is to reflect only the z-component of the local coordinates $T_i^{-1} \circ \vec{\mathbf{x}}_j$.

In the following we denote a set of frames and points by large Roman letters, for example $X = \left(\{\vec{\mathbf{x}}_i\}, \{T_j\}\right)$.

In particular, this means that both FAPE and IPA can distinguish global reflections of the protein assuming that the rigid frames are related to the underlying points by Algorithm 21, e.g.

$$\mathrm{FAPE}(X, X_{\text{reflection}}) = \sum_{ij} \left\| T_i^{-1} \circ \vec{\mathbf{x}}_j - \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \left(T_i^{-1} \circ \vec{\mathbf{x}}_j \right) \right\| \tag{16}$$

$$= 2 \sum_{ij} \left| (T_i^{-1} \circ \vec{\mathbf{x}}_j)_z \right|, \tag{17}$$

which is a large positive value outside of degenerate cases. For a more general proof that FAPE can distinguish chirality regardless of how the frames are constructed, see the next section.

There are other sources of chirality in AlphaFold. The atom positions are computed from a combination of the backbone frames and the predicted χ angles, and this procedure will always generate a left-handed molecule since it uses ideal values outside of the χ angles (i.e. the CB will always be in the left-handed location). AlphaFold can produce almost exactly the chiral pair for the backbone atoms, but it cannot build the chiral pair of the side chains. The model also has a small loss on the χ-angle values and those values are not invariant under reflection.

In order to test the importance of the chirality of FAPE we trained a model using the dRMSD loss [102] instead of FAPE, we show the results on the CASP14 set in Figure 9. Here we can see that the lDDT-Cα performance is still good, where we note that lDDT-Cα is a metric that cannot distinguish molecules of the opposite chirality.

However GDT shows a bimodal distribution if trained with a dRMSD loss, where one of the two modes is only somewhat worse than baseline AlphaFold and the other mode has very low accuracy (Suppl. Fig. 9). This suggests that the second mode composes molecules of reverse chirality. To test this we compute $\overline{GDT}$, which is the GDT of the mirror reflection of the structure used to compute GDT. These mirror structures also show bimodality of the GDT values. Finally taking the maximum over the GDT of the structure and its mirror produces consistently high GDT. This confirms that AlphaFold trained with a dRMSD loss frequently produces mirror structures, and that FAPE is the main component that ensures the correct chirality of predicted structures.

1.9.4 Configurations with FAPE(X,Y) = 0

To understand the points at which zero FAPE loss is achieved, we will introduce an RMSD-like auxiliary measure that operates only on points and not frames

$$S(\{\vec{\mathbf{x}}_i\}, \{\vec{\mathbf{y}}_i\}) = \min_{T^{\text{align}} \in \mathrm{SE}(3)} \frac{1}{N_{\text{point}}} \sum_j \left\| \vec{\mathbf{x}}_j - T^{\text{align}} \circ \vec{\mathbf{y}}_j \right\| \tag{18}$$

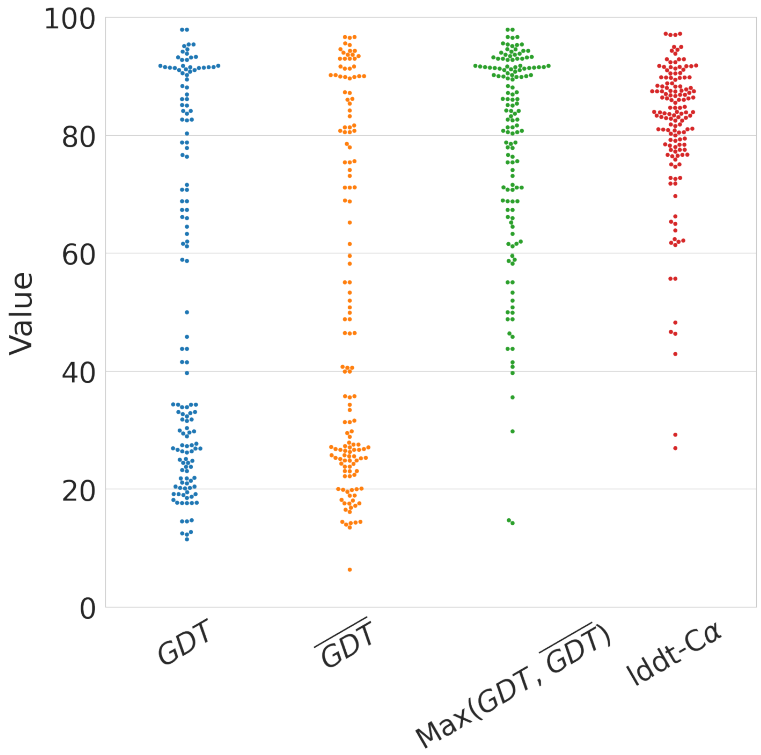
where T is a proper rigid transformation. Then we can show a lower bound of FAPE irrespective of the values of the frames,

$$\mathrm{FAPE}(X, \tilde{X}) = \frac{1}{N_{\text{frames}} N_{\text{point}}} \sum_{ij} \left\| T_i^{-1} \circ \vec{\mathbf{x}}_j - \tilde{T}_i^{-1} \circ \vec{\tilde{\mathbf{x}}}_j \right\| \tag{19}$$

$$= \frac{1}{N_{\text{frames}}} \sum_i \left(\frac{1}{N_{\text{point}}} \sum_j \left\| \vec{\mathbf{x}}_j - (T_i \circ \tilde{T}_i^{-1}) \circ \vec{\tilde{\mathbf{x}}}_j \right\| \right) \tag{20}$$

$$\geq \frac{1}{N_{\text{frames}}} \sum_i S(\{\vec{\mathbf{x}}_j\}, \{\vec{\tilde{\mathbf{x}}}_j\}) \tag{21}$$

$$= S(\{\vec{\mathbf{x}}_j\}, \{\vec{\tilde{\mathbf{x}}}_j\}), \tag{22}$$



Supplementary Figure 9 | Accuracy distribution of a model trained with dRMSD instead of the FAPE loss ($N = 87$ protein domains). Here $\overline{\text{GDT}}$ denotes the GDT score obtained when comparing to the chirally reflected target structure.

since the S function minimizes the quantity in parentheses over all proper rigid transformations and $T_i\tilde{T}_i^{-1}$ is a proper rigid transformation. This inequality simply shows that the point errors averaged over all local frames are no less than the point error for the best single global alignment under the same distance function.

The value $S(\{\vec{\mathbf{x}}_i\}, \{\vec{\tilde{\mathbf{x}}}_i\})$ is zero iff $\text{RMSD}(\{\vec{\mathbf{x}}_i\}, \{\vec{\tilde{\mathbf{x}}}_i\}) = 0$, which shows that $\text{FAPE}(X, \tilde{X}) = 0$ is only possible if the points have $\text{RMSD}(\{\vec{\mathbf{x}}_i\}, \{\vec{\mathbf{y}}_i\}) = 0$. We can characterize all pairs such that $\text{FAPE}(X, \tilde{X}) = 0$ as those where $\text{RMSD}(\{\vec{\mathbf{x}}_i\}, \{\vec{\tilde{\mathbf{x}}}_i\}) = 0$ and $T_i^{-1} \circ \tilde{T}_i$ is one of rigid motions that achieves this zero RMSD (which is unique unless the point sets are degenerate). In particular, the FAPE loss always has non-zero values chiral pairs if the point sets are non-degenerate, regardless of the how the frames are constructed.

1.9.5 Metric properties of FAPE

In this section, we show that idealized FAPE (i.e. using $\epsilon = 0$) has all the properties of a pseudometric, which is a generalized distance function used in topology. To reduce clutter, the proofs are shown without a cutoff but hold equally well when $\|\cdot\|$ is treated as a clipped vector norm. The reader uninterested in the mathematical details may skip this section.

It is simple to show from the defining equation that

$$\text{FAPE}(X, Y) \geq 0 \tag{23}$$
$$\text{FAPE}(X, X) = 0 \tag{24}$$
$$\text{FAPE}(X, Y) = \text{FAPE}(Y, X), \tag{25}$$

so the only remaining pseudometric property to prove is the triangle inequality, $\text{FAPE}(X, Z) \leq \text{FAPE}(X, Y) + \text{FAPE}(Y, Z)$. The triangle inequality is proven below, using tildes and overbars to designate the different con-

figurations.

$$\text{FAPE}(X, \tilde{X}) = \sum_{i,j} \left\| T_i^{-1} \circ \vec{\mathbf{x}}_j - \tilde{T}_i^{-1} \circ \vec{\tilde{\mathbf{x}}}_j \right\| \tag{26}$$

$$= \sum_{i,j} \left\| T_i^{-1} \circ \vec{\mathbf{x}}_j - \bar{T}_i^{-1} \circ \vec{\bar{\mathbf{x}}}_j + \bar{T}_i^{-1} \circ \vec{\bar{\mathbf{x}}}_j - \tilde{T}_i^{-1} \circ \vec{\tilde{\mathbf{x}}}_j \right\| \tag{27}$$

$$\leq \sum_{i,j} \left(\left\| T_i^{-1} \circ \vec{\mathbf{x}}_j - \bar{T}_i^{-1} \circ \vec{\bar{\mathbf{x}}}_j \right\| + \left\| \bar{T}_i^{-1} \circ \vec{\bar{\mathbf{x}}}_j - \tilde{T}_i^{-1} \circ \vec{\tilde{\mathbf{x}}}_j \right\| \right) \tag{28}$$

$$= \sum_{i,j} \left\| T_i^{-1} \circ \vec{\mathbf{x}}_j - \bar{T}_i^{-1} \circ \vec{\bar{\mathbf{x}}}_j \right\| + \sum_{i,j} \left\| \bar{T}_i^{-1} \circ \vec{\bar{\mathbf{x}}}_j - \tilde{T}_i^{-1} \circ \vec{\tilde{\mathbf{x}}}_j \right\| \tag{29}$$

$$= \text{FAPE}(X, \bar{X}) + \text{FAPE}(\bar{X}, \tilde{X}). \tag{30}$$

1.9.6 Model confidence prediction (pLDDT)

Our model provides intrinsic model accuracy estimates by predicting the per-residue lDDT-Cα [99] scores. We call this confidence measure pLDDT. Algorithm 29 computes it by taking the final single representation from the structure module (Algorithm 20 line 30), and projecting it into 50 bins (each bin covering a 2 lDDT-Cα range).

For training, we score the final predicted structure with per-residue lDDT-Cα against the ground truth structure. This score is discretized into 50 bins and used as the target value for the cross-entropy loss $\mathcal{L}_{\text{conf}}$ averaged over the residues. Additionally, we only train on the examples with resolution between 0.1 Å and 3.0 Å, and ignoring any NMR structure, to ensure high-quality ground truth data.

For evaluation, we compute the expected value of the per-residue pLDDT distribution (Algorithm 29 line 5). The *chain* pLDDT is obtained as an average of the per-residue pLDDT scores within the chain.

Algorithm 29 Predict model confidence pLDDT

def predictPerResidueLDDT_Ca($\{\mathbf{s}_i\}, \mathbf{v}_{\text{bins}} = [1, 3, 5, \ldots, 99]^\top, \{r_i^{\text{true LDDT}}\}, c = 128$) :

1: $\mathbf{a}_i = \text{relu}(\text{Linear}(\text{relu}(\text{Linear}(\text{LayerNorm}(\mathbf{s}_i)))))$ — $\mathbf{a}_i$, and intermediate activations $\in \mathbb{R}^c$

2: $\mathbf{p}_i^{\text{pLDDT}} = \text{softmax}(\text{Linear}(\mathbf{a}_i))$ — $\mathbf{p}_i^{\text{pLDDT}} \in \mathbb{R}^{|\mathbf{v}_{\text{bins}}|}$

3: $\mathbf{p}_i^{\text{true LDDT}} = \text{one\_hot}(r_i^{\text{true LDDT}}, \mathbf{v}_{\text{bins}})$

4: $\mathcal{L}_{\text{conf}} = \text{mean}_i(\mathbf{p}_i^{\text{true LDDT}^\top} \log \mathbf{p}_i^{\text{pLDDT}})$

5: $r_i^{\text{pLDDT}} = \mathbf{p}_i^{\text{pLDDT}^\top} \mathbf{v}_{\text{bins}}$ — $r_i^{\text{pLDDT}} \in \mathbb{R}$

6: **return** $\{r_i^{\text{pLDDT}}\}, \mathcal{L}_{\text{conf}}$

1.9.7 TM-score prediction

The pLDDT head above predicts the value of lDDT-Cα, which is a local error metric that operates pairwise but by design is not sensitive to what fraction of the residues can be aligned using a single global rotation and translation. This can be disadvantageous for assessing whether the model is confident in its overall domain packing for large chains. In this section we develop a predictor of the global superposition metric TM-score [103].

We denote the ground truth structure's Cα atoms $X^{\text{true}} = \{\vec{\mathbf{x}}_j^{\text{true}}\}$, its backbone frames $\{T_i^{\text{true}}\}$, the predicted structure's Cα atoms via $X = \{\vec{\mathbf{x}}_j\}$, and the corresponding backbone frames $\{T_i\}$. Let the number of residues be N_{res}, and assume all residues are resolved in the ground truth. Denoting $T^{\text{align}} = (R^{\text{align}}, \vec{\mathbf{t}}^{\text{align}})$ an arbitrary

rigid transformation, the TM-score is defined as

$$\mathrm{TM}(X, X^{\text{true}}) = \max_{T^{\text{align}} \in \mathrm{SE}(3)} \frac{1}{N_{\text{res}}} \sum_{j=1}^{N_{\text{res}}} f\left(\|\vec{\mathbf{x}}_j - T^{\text{align}} \circ \vec{\mathbf{x}}_j^{\text{true}}\|\right), \tag{31}$$

$$f(d) = \frac{1}{1 + \left(\frac{d}{d_0(N_{\text{res}})}\right)^2}, \tag{32}$$

$$d_0(N_{\text{res}}) = 1.24 \sqrt[3]{\text{maximum}(N_{\text{res}}, 19) - 15} - 1.8 \tag{33}$$

In the last equation, N_{res} is clipped to avoid negative or undefined values for very short proteins. In practice, global optimisation over T^{align} is infeasible, so computational methods rely on approximations, e.g. TM-align [104] uses an iterative procedure of choosing a best-fit alignment on a subset of residues and then re-computing this set of residues.

We now develop an approximation to the TM-score that relies on a pairwise accuracy measure which can be predicted from a pair representation. Crucially, this approximation allows prediction of the TM-score for an arbitrary subset of residues (such as a domain) without rerunning the structure prediction.

$$\mathrm{TM}(X, X^{\text{true}}) = \max_{T^{\text{align}} \in \mathrm{SE}(3)} \frac{1}{N_{\text{res}}} \sum_j f\left(\|\vec{\mathbf{x}}_j - T^{\text{align}} \circ \vec{\mathbf{x}}_j^{\text{true}}\|\right) \tag{34}$$

$$\geq \max_{i \in [1,\ldots,N_{\text{res}}]} \frac{1}{N_{\text{res}}} \sum_j f\left(\|\vec{\mathbf{x}}_j - (T_i T_i^{\text{true}\,-1}) \circ \vec{\mathbf{x}}_j^{\text{true}}\|\right) \tag{35}$$

$$= \max_i \frac{1}{N_{\text{res}}} \sum_j f\left(\|T_i^{-1} \circ \vec{\mathbf{x}}_j - T_i^{\text{true}\,-1} \circ \vec{\mathbf{x}}_j^{\text{true}}\|\right). \tag{36}$$

Above, we replace the continuous maximum over all possible alignments with a discrete maximum set of N_{res} single-residue alignments ($T_i T_i^{\text{true}\,-1}$). This clearly lower-bounds the original maximum, with the bound becoming tight when the global superposition exactly aligns the backbone of any residue. Then, we symmetrize the vector difference by applying the rigid transformation T_i. Note that the expression in Equation 36 is closely related to FAPE, except that the f function is somewhat different and FAPE has a mean instead of maximum (i.e. FAPE considers *all* 1-residue alignments instead of only the best one).

Next, we assume that the correct structure X^{true} is unknown and that we have a distribution of probable structures for which we seek the expected value of the TM-score for our prediction X:

$$\mathbb{E}\left[\mathrm{TM}(X, X^{\text{true}})\right] \geq \mathbb{E}\left[\max_i \frac{1}{N_{\text{res}}} \sum_j f\left(\|T_i^{-1} \circ \vec{\mathbf{x}}_j - T_i^{\text{true}\,-1} \circ \vec{\mathbf{x}}_j^{\text{true}}\|\right)\right] \tag{37}$$

$$\geq \max_i \frac{1}{N_{\text{res}}} \sum_j \mathbb{E}\left[f\left(\|T_i^{-1} \circ \vec{\mathbf{x}}_j - T_i^{\text{true}\,-1} \circ \vec{\mathbf{x}}_j^{\text{true}}\|\right)\right]. \tag{38}$$

In the final line, we replace the expectation-of-maximum with a maximum-of-expectation (a lower bound due to Jensen's inequality), and use the linearity of expectation.

The pairwise matrix $e_{ij} = \|T_i^{-1} \circ \vec{\mathbf{x}}_j - T_i^{\text{true}\,-1} \circ \vec{\mathbf{x}}_j^{\text{true}}\|$ is a non-symmetric matrix that captures the error in the position of the Cα atom of residue j when the predicted and true structures are aligned using the backbone frame of residue i. The probability distribution of its elements can be readily predicted by a neural network. To do that, we discretize the distribution of e_{ij} into 64 bins, covering the range from 0 to 31.5 Å with 0.5 Å bin width. During training, the final bin also captures any larger errors. We compute e_{ij} as a linear projection of the pair representation $\mathbf{z}_{ij}$, followed by a softmax. We fine-tune the CASP models to additionally predict e_{ij} using the average categorical cross-entropy loss, with weight 0.1 (weights 0.01 and 1.0 were also tried but those weights lowered either pTM accuracy or structure prediction accuracy, respectively). Just as for

the pLDDT prediction, we only train this prediction module on non-NMR examples with resolution between 0.1 Å and 3.0 Å. The fine-tuning took roughly 16 hours and processed $3 \cdot 10^5$ samples.

Using e_{ij}, we approximate the TM-score as

$$\text{pTM} = \max_i \frac{1}{N_{\text{res}}} \sum_j \mathbb{E}\left[f(e_{ij})\right], \tag{39}$$

where the expectation is taken over the probability distribution defined by e_{ij}. We find that pTM is a pessimistic predictor of the true TM-score, which is as expected, given that both approximations used in the derivation of pTM are lower bounds.

Note that given a full-chain prediction of e_{ij}, we can obtain a TM-score prediction for any subset of residues $\mathcal{D}$ (such as a particular domain) simply by restricting the range of residues considered:

$$\text{pTM}(\mathcal{D}) = \max_{i \in \mathcal{D}} \frac{1}{|\mathcal{D}|} \sum_{j \in \mathcal{D}} \mathbb{E} \frac{1}{1 + \left(\frac{e_{ij}}{d_0(|\mathcal{D}|)}\right)^2}, \tag{40}$$

where $|\mathcal{D}|$ is the number of residues in the set $\mathcal{D}$. By small modifications, a similar expression could be derived to estimate GDT, FAPE, or RMSD from the e_{ij} matrix, though we do not pursue this further. Additionally, the 2-D image of the expected values of $f(e_{ij})$ serves as a good visualization of confident domain packings within the structure.

1.9.8 Distogram prediction

We linearly project the symmetrized pair representations $(\mathbf{z}_{ij} + \mathbf{z}_{ji})$ into 64 distance bins and obtain the bin probabilities p^b_{ij} with a softmax. The bins cover the range from 2 Å to 22 Å; and they have equal width apart from the last bin, which also includes any more distant residue pairs. For prediction targets y^b_{ij} we use one-hot encoded binned residue distances, which are computed from the ground-truth beta carbon positions for all amino acids except glycine where use alpha carbon instead. We use the cross-entropy loss averaged over all residue pairs:

$$\mathcal{L}_{\text{dist}} = -\frac{1}{N^2_{\text{res}}} \sum_{i,j} \sum_{b=1}^{64} y^b_{ij} \log p^b_{ij} \,. \tag{41}$$

1.9.9 Masked MSA prediction

Similarly to the common masked language modelling objectives [105], we use the final MSA representations to reconstruct MSA values that have been previously masked out (see subsubsection 1.2.7). We consider 23 classes, which include 20 common amino acid types, an unknown type, a gap token, and a mask token. MSA representations $\{\mathbf{m}_{si}\}$ are linearly projected into the output classes, passed through a softmax, and scored with the cross-entropy loss:

$$\mathcal{L}_{\text{msa}} = -\frac{1}{N_{\text{mask}}} \sum_{s,i \in \text{mask}} \sum_{c=1}^{23} y^c_{si} \log p^c_{si} \,, \tag{42}$$

where p^c_{si} are predicted class probabilities, y^c_{si} are one-hot encoded ground-truth values, and averaging happens over the masked positions.

1.9.10 "Experimentally resolved" prediction

The model contains a head that predicts if an atom is experimentally resolved in a high-resolution structure. The input for this head is the single representation $\{\mathbf{s}_i\}$ produced by the Evoformer stack (subsection 1.6). The single representation is projected with a Linear layer and a sigmoid to atom-wise probabilities $\{p_i^{\text{exp resolved},a}\}$

with $i \in [1, \ldots, N_{\text{res}}]$ and $a \in \mathcal{S}_{\text{atom names}}$. We train this head during fine-tuning (see subsubsection 1.12.1) on high-resolution X-ray crystals and cryo-EM structures (resolution better than 3Å) with standard cross-entropy,

$$\mathcal{L}_{\text{exp resolved}} = \text{mean}_{(i,a)} \left(-y_i^a \log p_i^{\text{exp resolved},a} - (1 - y_i^a) \log(1 - p_i^{\text{exp resolved},a}) \right) , \tag{43}$$

where $y_i^a \in \{0, 1\}$ denotes the ground truth, i.e. if atom a in residue i was resolved in the experiment.

1.9.11 Structural violations

The construction of the atom coordinates from independent backbone frames and torsion-angles (**Fig. 3e**), subsubsection 1.8.4) produces idealized bond lengths and bond angles for most of the atom bonds, but the geometry for inter-residue bonds (peptide bonds) and the avoidance of atom clashes need to be learned. We introduce extra losses that penalize these structural violations to ensure these constraints everywhere, i.e. also in regions where no ground truth atom coordinates are available. We constructed the losses in a way that loss-free structures will pass the stereochemical quality checks in the lDDT metric [99].

The losses use a flat-bottom L1 loss that penalizes violations above a certain tolerance threshold τ. The bond length violation loss is computed as

$$\mathcal{L}_{\text{bondlength}} = \frac{1}{N_{\text{bonds}}} \sum_{i=1}^{N_{\text{bonds}}} \max \left(|\ell_{\text{pred}}^i - \ell_{\text{lit}}^i| - \tau,\ 0 \right) , \tag{44}$$

where ℓ_{pred}^i is a bond length in the predicted structure and ℓ_{lit}^i is the literature value [98] for this bond length. N_{bonds} is the number of all bonds in this structure. We set the tolerance τ to $12\sigma_{\text{lit}}$, where σ_{lit} is the literature standard deviation of this bond length. We chose the factor 12 to ensure that the produced bond lengths pass the stereochemical quality checks lDDT metric [99], which also uses by default a tolerance factor of 12.

The bond angle violation loss uses the cosine of the angle computed from the dot-product of the unit vectors of the bonds, $\cos \alpha = \hat{\vec{\mathbf{v}}}_1^\top \hat{\vec{\mathbf{v}}}_2$ and also applies a flat-bottom L1 loss on the deviations:

$$\mathcal{L}_{\text{bondangle}} = \frac{1}{N_{\text{angles}}} \sum_{i=1}^{N_{\text{angles}}} \max \left(|\cos \alpha_{\text{pred}}^i - \cos \alpha_{\text{lit}}^i| - \tau,\ 0 \right) \tag{45}$$

where α_{pred}^i is a bond angle in the predicted structure and α_{lit}^i is the literature value for this bond angle. N_{angles} is the number of all bond angles in this structure. The tolerance τ is computed such that the flat bottom extends from -12 to 12 times the literature standard deviation of this bond angle.

The clash loss uses a one-sided flat-bottom-potential, that penalizes too short distances only,

$$\mathcal{L}_{\text{clash}} = \sum_{i=1}^{N_{\text{nbpairs}}} \max \left(d_{\text{lit}}^i - \tau - d_{\text{pred}}^i,\ 0 \right) , \tag{46}$$

where d_{pred}^i is the distance of two non-bonded atoms in the predicted structure and d_{lit}^i is the "clashing distance" of these two atoms according to their literature Van der Waals radii. N_{nbpair} is the number of all non-bonded atom pairs in this structure. The tolerance τ is set to 1.5 Å.

All these losses together build the violation loss

$$\mathcal{L}_{\text{viol}} = \mathcal{L}_{\text{bondlength}} + \mathcal{L}_{\text{bondangle}} + \mathcal{L}_{\text{clash}} . \tag{47}$$

We apply this violation loss only during the fine-tuning training phase. Switching it on in the early training leads to strong instabilities in the training dynamics.

1.10 Recycling iterations

We find it helpful to execute the network multiple times, each time embedding the previous outputs as additional inputs. We call this technique "recycling", and we find it relatively important in the ablation studies (subsection 1.13). In this section we first explain how recycling operates in general, both at training and the inference time. Then, we describe the specific features being recycled in the AlphaFold model. Algorithm 2 shows the full inference of AlphaFold with recycling and ensembling (MSA resampling and ensembling is further explained in subsubsection 1.11.2).

Algorithm 30 Generic recycling inference procedure

def RecyclingInference($\{\text{inputs}_c\}, N_{\text{cycle}}$) :

outputs $= \mathbf{0}$

 # Recycling iterations

for all $c \in [1, \ldots, N_{\text{cycle}}]$ **do** *# shared weights*
  outputs $\leftarrow$ Model(inputs_c, outputs)
end for
return outputs

Algorithm 31 Generic recycling training procedure

def RecyclingTraining($\{\text{inputs}_c\}, N_{\text{cycle}}$) :

$N' = \text{uniform}(1, N_{\text{cycle}})$ *# shared value across the batch*

outputs $= \mathbf{0}$

 # Recycling iterations

for all $c \in [1, \ldots, N']$ **do** *# shared weights*
  outputs $\leftarrow$ stopgrad(outputs) *# no gradients between iterations*
  outputs $\leftarrow$ Model(inputs_c, outputs)
end for
return loss(outputs) *# only the final iteration's outputs are used*

Recycling allows to make the network deeper and to process multiple versions of the input features (e.g. incorporate the MSA resampling) without significantly increasing the training time or the number of parameters. At the inference time, recycling yields a recurrent network with shared weights, where each iteration c takes in the input features inputs_c and the previous iteration's outputs, and produces the new refined outputs, see Algorithm 30. The initial "outputs" are constant zeros, since we find that networks can easily accommodate this special case. We will denote the number of iterations of this network via N_{cycle}. We use $N_{\text{cycle}} = 4$ in AlphaFold.

While it is possible to train recycling by simply unrolling the N_{cycle} iterations, the computational and memory cost of that may be prohibitive. Instead, we introduce an approximate training scheme, which empirically works well and significantly reduces the extra cost of recycling during training, Algorithm 31. We first define an objective, the average loss of the model over all iterations:

$$\frac{1}{N_{\text{cycle}}} \sum_{c=1}^{N_{\text{cycle}}} \text{loss}(\text{outputs}_c) \tag{48}$$

where outputs$_i$ are the model's outputs at iteration c. Next, we construct an unbiased Monte Carlo estimate of this objective by uniformly sampling the number of iterations N' between 1 an N_{cycle}, and only train the loss for that step. This means that any following iterations can be skipped. The random choice of the number of iterations N' is synchronized across the batch (i.e. every batch element trains the same iteration on each step) to increase efficiency. Finally, we stop the gradients flowing from the chosen iteration to the previous ones, allowing to skip the backward pass (backpropagation) for the previous iterations $c < N'$. As an example, if we sampled $N' = 3$, and $N_{\text{cycle}} = 4$, then we perform just the forward pass for the first two iterations; both the forward and the backward pass for the third iteration; and the fourth iteration is completely skipped.

We now explain the motivation behind this procedure, compared to simple unrolling. Randomly sampling the iteration N' improves the efficiency, since we now perform $\frac{1}{N_{\text{cycle}}}\sum_{c=1}^{N_{\text{cycle}}} c = \frac{N_{\text{cycle}}+1}{2}$ iterations on average instead of N_{cycle}. Importantly, it also acts as auxiliary loss for the model, requiring to provide plausible outputs mid-way through the inference. We hypothesize that this allows the network to refine its own predictions multiple times. Stopping the gradients of the intermediate outputs makes the procedure much cheaper, both memory-wise and computationally. Memory-wise, it allows to eschew storing the intermediate activations of the previous iterations. Computationally, this procedure requires a single backward pass and N' forward passes. With rematerialization (see subsubsection 1.11.8), a forward pass is approximately 4 times cheaper than the full forward and backward round, thus each forward pass adds only 25% of the baseline training time. Therefore, training a model with N_{cycle} iterations has an additional average cost of $(12.5\,(N_{\text{cycle}} - 1))\%$. For $N_{\text{cycle}} = 4$ used in AlphaFold, this corresponds to just 37.5% extra training time, compared to 300% with unrolling. While stopping the gradients between iterations leads to a bias in the gradients, we find that does not hamper training.

In AlphaFold, we recycle the predicted backbone atom coordinates from the Structure module (Algorithm 20), and output pair and first row MSA representations from the Evoformer (Algorithm 6). Algorithm 32 shows their embedding details. Both types of representations are passed through LayerNorm to prepare updates for the corresponding input representations $\{\mathbf{z}_{ij}\}$ and $\{\mathbf{m}_{1i}\}$. Predicted coordinates of beta carbon atoms (alpha carbon for glycine) are used to compute pairwise distances, which are then discretized into 15 bins of equal width 1.25Å that span the range to approx. 20Å (Due to historical reasons the precise bin values range from 3⅜ Å to 21⅜ Å). The resulting one-hot distogram is linearly projected and added to the pair representation update. The recycling updates are injected to the network as shown in line 6 of Algorithm 2 (for a visual scheme, see R sign in Suppl. Fig. 1). This is the only mechanism employed for recycling previous predictions, and the rest of the network is identical on all recycling iterations.

Algorithm 32 Embedding of Evoformer and Structure module outputs for recycling

def RecyclingEmbedder($\{\mathbf{m}_{1i}\}, \{\mathbf{z}_{ij}\}, \{\vec{\mathbf{x}}_i^{C^\beta}\}$) :

 # Embed pair distances of backbone atoms:

1: $d_{ij} = \left\| \vec{\mathbf{x}}_i^{C^\beta} - \vec{\mathbf{x}}_j^{C^\beta} \right\|$ — C^α used for glycin

2: $\mathbf{d}_{ij} = \text{Linear}(\text{one\_hot}(d_{ij}, \mathbf{v}_{\text{bins}} = [3⅜\,\text{Å},\ 5⅛\,\text{Å},\ \ldots, 21⅜\,\text{Å}]))$ — $\mathbf{d}_{ij} \in \mathbb{R}^{c_z}$

 # Embed output Evoformer representations:

3: $\tilde{\mathbf{z}}_{ij} = \mathbf{d}_{ij} + \text{LayerNorm}(\mathbf{z}_{ij})$

4: $\tilde{\mathbf{m}}_{1i} = \text{LayerNorm}(\mathbf{m}_{1i})$

5: **return** $\{\tilde{\mathbf{m}}_{1i}\}, \{\tilde{\mathbf{z}}_{ij}\}$

1.11 Training and inference details

Table 4 | AlphaFold training protocol. We train each stage until convergence with the approximate timings and number of samples provided.

Model	Initial training	Fine-tuning
Number of templates N_{templ}	4	4
Sequence crop size N_{res}	256	384
Number of sequences N_{seq}	128	512
Number of extra sequences $N_{\text{extra\_seq}}$	1024	5120
Parameters initialized from	Random	Initial training
Initial learning rate	10^{-3}	$5 \cdot 10^{-4}$
Learning rate linear warm-up samples	128000	0
Structural violation loss weight	0.0	1.0
Training samples ($\cdot 10^6$)	≈ 10	≈ 1.5
Training time	≈ 7 days	≈ 4 days

1.11.1 Training stages

The training proceeds in two stages outlined in Table 4. We train with sequence crop size 256 and then fine-tune with the larger crop size 384, where the second stage starts from the pre-trained model weights and uses the lower learning rate and no warm-up steps. Additionally, we use the increased number of MSA clusters as well as unclustered (extra) sequences. We also apply the structural violation loss.

1.11.2 MSA resampling and ensembling

There are several sources of stochastic behaviour in the network. The most important one is MSA pre-processing, which includes MSA block deletion (Algorithm 1) and MSA clustering procedure (subsubsection 1.2.7). To reduce the stochastic effects of these steps, we perform them multiple times, resulting in a set of sampled MSA and extra MSA features. Note that the residue cropping (subsubsection 1.2.8), which follows these steps, is done consistently across all samples, i.e. the same residue ranges are taken. Also note that we do not perform template re-sampling, i.e. the same templates (up to 4) are used for one training example.

To take advantage of the multiple samples of these features, we embed them multiple times with the Evoformer network and average the output embeddings. Concretely, at each recycling iteration we perform multiple passes of the Evoformer (including all preceding input embedding transformations) for the different instances of "msa_feat" and "extra_msa_feat", and we average the output pair and single representations before providing them to the structure module or any other heads of the network (see Algorithm 2). We call this technique ensembling (although it does not ensemble final predictions) and we employ it during inference, but not during training. Finally, since we also use different samples of the MSA and extra MSA features at each recycling iteration, the full system processes the total of $N_{\text{cycle}} \times N_{\text{ensemble}}$ samples.

We use $N_{\text{ensemble}} = 3$ for the best checkpoint selection in the evaluator (subsubsection 1.11.7) and also for the inference on CASP14 targets performed for this paper. However, with the recycling and pLDDT-based selection from 5 model runs, we have found that the additional ensembling has only a minor impact on the accuracy of the system (see subsubsection 1.12.2 for details). Given that N_{ensemble} is almost a linear multiplier for the inference time, we are gradually deprecating it in our systems, and the inference on the recent PDB dataset for this paper has been performed with $N_{\text{ensemble}} = 1$.

1.11.3 Optimization details

For optimization we use Adam [106] with the base learning rate 10^{-3}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-6}$. We linearly increase (warm-up) the learning rate over the first $0.128 \cdot 10^6$ samples, and further decrease the learning rate by a factor of 0.95 after $6.4 \cdot 10^6$ samples. During the fine-tuning stage we have no learning rate warm-up, but we reduce the base learning rate by half.

We train using the mini-batch size of 128, one example per TPU-core. To stabilize the training, we apply gradient clipping by the global norm [107] on the parameters independently to every training example in a mini-batch, with a clipping value 0.1.

1.11.4 Parameters initialization

Linear layers. By default, the weights of the Linear layers are initialized using the LeCun (fan-in) initialization strategy [108] with a truncated normal distribution. By default, the biases of the Linear layers are initialized by zero. For the layers immediately followed by a ReLU activations, we use He initializer [109]. The queries, keys and values projection layers in self-attention layers are initialized using the 'fan-average' Glorot uniform scheme [110].

To improve stability of training, we zero-initialize the weights of the *final* weight layers [111] in every residual layer. This ensures that every residual layer acts as an identity operation at initialization. Furthermore, we zero-initialize all the final projection weight layers of the network: masked MSA prediction logits, residue distance prediction logits, model confidence prediction logits. We also identity-initialize the rigid frame updates in the Structure module.

Gating Linear layers, i.e. the Linear layers immediately followed by a $\mathrm{sigmoid}$, are initialized with zero weights. The biases are initialized with a constant value of 1, ensuring that at initialisation, the gates are mostly-opened with a value of $\mathrm{sigmoid}(1) = 1/(1 + \exp(-1)) \approx 0.73$.

Layer normalization layers are initialized using the standard scheme: gains set to 1 and biases set to 0.

Point weights in Invariant Point Attention γ^h are initialized to be 1 after the softplus transformation.

1.11.5 Loss clamping details

In 90% of training mini-batches the FAPE backbone loss is clamped by $e_{\max} = 10$ Å, in the remaining 10% it is not clamped, $e_{\max} = +\infty$. For side-chains it is always clamped by $e_{\max} = 10$ Å.

1.11.6 Dropout details

We denote the standard Dropout [97] with operator $\mathrm{Dropout}_x$, where x is the dropout rate, i.e. the probability to zero an entry. In residual updates for the self-attention operations, we modify it such that dropout masks are shared across one of the dimensions. We use $\mathrm{DropoutRowwise}_x$ operator to denote the version with the mask of shape $[1,\ N_{\text{res}},\ N_{\text{channel}}]$, which is shared across rows. It is used in the residual updates following the MSA row-wise self-attention and the triangular self-attention around starting node. For a given residue (column) the same channels are set to zero across all rows in these updates. Similarly, we use $\mathrm{DropoutColumnwise}_x$ operator to denote the version with the mask of the shape $[N_{\text{res}},\ 1,\ N_{\text{channel}}]$, which is shared across columns. It is used in the residual update following the triangular self-attention around ending node. In the main Evoformer stack we also apply the row-wise Dropout when performing the residual updates for the triangular multiplicative operations in the pair stack. We keep the same configuration for the unclustered MSA stack and the Template pair stack. In the Structure module, we apply Dropout to the results of the Invariant Point Attention and the Transition layer. Please refer to Algorithm 6, Algorithm 16, Algorithm 18, and Algorithm 20 for the exact locations of all Dropout operations and their rates.

1.11.7 Evaluator setup

During evaluation, we use an exponential moving average of the trained parameters [112] with the decay 0.999. To select the best model during training, we monitor lDDT-Cα performance on a validation set of targets collected from CAMEO [113] over 3 months period (2018-12-14 to 2019-03-09). It is filtered to a maximum sequence length of 700, and there is no residue cropping at evaluation time.

1.11.8 Reducing the memory consumption

AlphaFold model has high memory consumption, both in training and during inference. This is largely driven by the logits in the self-attention layers and the large pair and MSA activiations. For example, a^h_{ijk} in Algorithm 13 and Algorithm 14 consists of $(N^3_{\text{res}} \cdot N_{\text{head}})$ floating point numbers. We now describe the techniques we employ to perform memory-efficient training and inference of this model.

Training. We store all intermediate activations of the model in `bfloat16` format, which only requires two bytes per floating point number. Training is performed on relatively short sequence crop sizes, up to $N_{\text{res}} = 384$. The cubic scaling of the self-attention logits makes it impractical to store them all for backpropagation, as is done in naive backpropagation. Specifically, storing just the attention logits of the largest sublayer type for all blocks requires more memory than the 16 GiB available on TPUv3 core:

$$\underset{N^3_{\text{res}}}{384^3} \cdot \underset{N_{\text{head}}}{4} \cdot \underset{N_{\text{block}}}{48} \cdot \underset{\text{bytes per bfloat16}}{2} = 20.25 \text{ GiB}. \tag{49}$$

To solve this, we leverage a standard technique called gradient checkpointing, or rematerialization [114, 115]. Concretely, during the forward pass, we store the activations that are passed between the $N_{\text{block}} = 48$ Evoformer blocks. During the backward pass, we recompute all activations within the blocks. This means we need to store the logits for only one layer at a time, bringing down the memory consumption to just 0.4 GiB. Storing the intermediate pair activations requires an extra

$$\underset{N^2_{\text{res}}}{384^2} \cdot \underset{c_z}{128} \cdot \underset{N_{\text{block}}}{48} \cdot \underset{\text{bytes per bfloat16}}{2} = 1.7 \text{ GiB}. \tag{50}$$

Rematerialization's computational cost is equivalent to an extra forward pass through the network. Empirically, the backward pass is twice as expensive as the forward pass, so rematerialization increases the training time by 33%.

Inference. The intermediate activations are stored in `float32` format, requiring four bytes per floating point number. During inference, we do not need to store the intermediate activations, which reduces the memory consumption. However, we also consider proteins with dramatically larger N_{res} than during training. For example, T1044 has $N_{\text{res}} = 2180$, so a single layer's logits have the size of

$$\underset{N^3_{\text{res}}}{2180^3} \cdot \underset{N_{\text{head}}}{4} \cdot \underset{\text{bytes per float32}}{4} = 154.4 \text{ GiB}, \tag{51}$$

which again exceeds the memory size of modern accelerators. To reduce this burden, for each type of layer in the network, we identify a 'batch-like' dimension where the computation is independent along that dimension. We then execute the layer one 'chunk' at a time, meaning that only the intermediate activations for that chunk need to be stored in memory at a given time. Thus, reducing the chunk size improves memory efficiency, but at a cost of performance, as small chunks cannot fully exploit the parallelism of the hardware. In practice, we often use a chunk size of 4, resulting in much lower memory consumption for the logits:

$$\underset{N^2_{\text{res}}}{2180^2} \cdot \underset{N_{\text{head}}}{4} \cdot \underset{\text{chunk size}}{4} \cdot \underset{\text{bytes per float32}}{4} = 0.3 \text{ GiB}. \tag{52}$$

The same technique was used in a similar context in [116].

1.12 CASP14 assessment

1.12.1 Training procedure

Table 5 | Training protocol for CASP14 models. The models in **bold** (i.e. **1.1.1** – **1.2.3**) were used in the assessment. We report the number of training samples and the training time (in days and hours) until the best validation score. Three dots (· · ·) indicate the same value as in the former column.

	initial training	first fine-tuning		second fine-tuning				
Model	1	1.1	1.2	**1.1.1**	**1.1.2**	**1.2.1**	**1.2.2**	**1.2.3**
Parameters initialized from	Random	Model 1	· · ·	Model 1.1	· · ·	Model 1.2	· · ·	· · ·
Number of templates N_{templ}	4	4	0	4	· · ·	0	· · ·	· · ·
Sequence crop size N_{res}	256	· · ·	· · ·	384	· · ·	· · ·	· · ·	· · ·
Number of sequences N_{seq}	128	512	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·
Number of extra sequences $N_{\text{extra\_seq}}$	1024	· · ·	· · ·	5120	1024	5120	· · ·	1024
Initial learning rate	10^{-3}	$5 \cdot 10^{-4}$	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·
Learning rate linear warm-up samples	128000	0	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·
Structural violation loss weight	0.0	1.0	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·
"Experimentally resolved" loss weight	0.0	0.01	· · ·	· · ·	· · ·	· · ·	· · ·	· · ·
Training samples ($\cdot 10^6$)	9.2	1.1	1.7	0.3	0.6	1.4	1.1	2.4
Training time	6d 6h	1d 10h	2d 3h	20h	1d 13h	4d 1h	3d	5d 12h

We now describe the exact training process for the models used in the CASP14 assessment. The training of the model proceeds in three stages with exact details presented in Table 5. First, we train model 1 with a lower crop size and number of sequences for about a week. Then, we fine-tune this model with more sequences, enabled violation losses, and a lower learning rate. For historical reasons, we also added the "experimentally resolved" predictor (subsubsection 1.9.10) at this stage. We fine-tune two models: model 1.1 uses templates, while model 1.2 does not (the parameters pertaining to templates are dropped in this case). To obtain the final five models, we perform a second round of fine-tuning with larger sequence crop size and more extra MSA sequences from these two models. All CASP14 models are trained with distillation (subsection 1.3) from a slightly earlier version of the model. Empirically, we find that small changes in the model details have little effect on the distillation procedure. The training process, excluding the preparation of the distillation dataset, takes about two weeks. Since the assessment, we have found that the intermediate fine-tuning stage can be omitted without degradation of quality, and we follow the simplified training protocol (Table 4) in our ablation studies.

We continued to improve the models during the CASP14 assessment, thus some of our submissions used slightly different details, and some submissions had manual interventions. For more details please refer to the short description in [117] and also an upcoming publication in "Proteins: Structure, Function, and Bioinformatics". For the accuracy comparison of the systems see subsubsection 1.12.2 below.

1.12.2 Inference and scoring

At inference time, the five models are independently executed on the same set of inputs. Then, the predictions are re-ranked according to the chain pLDDT confidence measure. The original CASP14 submissions were inferenced with $N_{\text{ensemble}} = 8$, which has been later reduced in our system.

After the CASP14 assessment we have re-evaluated the final system described in subsubsection 1.12.1 with $N_{\text{ensemble}} = 3$ and obtained a mean domain GDT of 87.65, compared to 87.66 GDT that our actual submissions get on the same set of domains. The median Cα RMSD at 95% coverage is 0.96 for both the described system and the actual submissions. Furthermore, the described system without ensembling ($N_{\text{ensemble}} = 1$) obtains GDT 87.56, which is only 0.1 points less than the full system, while yielding 8 times speedup for the inference.

For this evaluation, we are predicting the structure of all non-cancelled, non-server targets as full chains (except T1085 and T1086, for which the structures are embargoed). Note that the long (2180 residues) target T1044 structure has also been predicted as a whole chain and then split into the domain targets (T1031, T1033, T1035, T1037, T1039–T1043) for the evaluation. We used all FM, TBM-easy, TBM-hard and FM/TBM domains (87 in total):

T1024-D1, T1024-D2, T1026-D1, T1027-D1, T1029-D1, T1030-D1, T1030-D2, T1031-D1, T1032-D1, T1033-D1, T1034-D1, T1035-D1, T1037-D1, T1038-D1, T1038-D2, T1039-D1, T1040-D1, T1041-D1, T1042-D1, T1043-D1, T1045s2-D1, T1046s1-D1, T1046s2-D1, T1047s1-D1, T1047s2-D1, T1047s2-D2, T1047s2-D3, T1049-D1, T1050-D1, T1050-D2, T1050-D3, T1052-D1, T1052-D2, T1052-D3, T1053-D1, T1053-D2, T1054-D1, T1055-D1, T1056-D1, T1057-D1, T1058-D1, T1058-D2, T1060s2-D1, T1060s3-D1, T1061-D1, T1061-D2, T1061-D3, T1064-D1, T1065s1-D1, T1065s2-D1, T1067-D1, T1068-D1, T1070-D1, T1070-D2, T1070-D3, T1070-D4, T1073-D1, T1074-D1, T1076-D1, T1078-D1, T1079-D1, T1080-D1, T1082-D1, T1083-D1, T1084-D1, T1087-D1, T1089-D1, T1090-D1, T1091-D1, T1091-D2, T1091-D3, T1091-D4, T1092-D1, T1092-D2, T1093-D1, T1093-D2, T1093-D3, T1094-D1, T1094-D2, T1095-D1, T1096-D1, T1096-D2, T1099-D1, T1100-D1, T1100-D2, T1101-D1, T1101-D2.

We use this set for all CASP14-related results reported in the paper.

1.13 Ablation studies

1.13.1 Architectural details

We estimate the relative importance of key components of the architecture by training and evaluating a number of ablation models:

Baseline. Baseline model as described in the paper without noisy-student self-distillation. All other ablations should be understood relative to this baseline model.

With self-distillation training. Full model as described in the paper including noisy-student self-distillation training.

No templates. We remove the template stack and the template torsion angle features, see subsubsection 1.7.1 and Suppl. Fig. 1. The number of MSA clusters N_{clust} is increased by $N_{\text{templ}} = 4$ to keep the overall size of the MSA representation.

No raw MSA (use MSA pairwise frequencies). AlphaFold builds representations by performing attention on individual MSA sequences. Here we ablate whether this is important and modify the system to provide only first and second order MSA statistics.

We denote the raw MSA data as M, where M_{si} is the amino acid of sequence s at position i. We also denote the total number of sequences as S. We define the MSA profile for position i as the empirical distribution of amino acids occurring at this position:

$$p_i(A) = \frac{1}{S}\sum_{s=1}^{S}[M_{si} = A], \tag{53}$$

where $[\,]$ denotes the Iverson bracket and is equal to 1 if the expression inside is true and 0 otherwise. Furthermore we define the pairwise frequency table as the probability of observing amino acid A at position i and amino acid B at position j:

$$f_{ij}(A, B) = \frac{1}{S}\sum_{s=1}^{S}[M_{si} = A][M_{sj} = B]. \tag{54}$$

We note that these type of MSA features are commonly used in the literature, e.g. by the trRosetta pipeline [118] (with further down-weighting of redundant sequences in the averages). Usually these first and second order statistics are transformed by computing the inverse covariation matrix or fitting a statistical model like

a Potts Model before the features are passed to the network. Here we will just use the raw features without sequence nonlocal preprocessing and allow the model to process them as needed.

This ablation will have access to the following MSA information:

- MSA profile, i.e. the marginal distribution $p_i(A)$;
- Expected number of deletions for a given position;
- MSA pairwise frequency table $f_{ij}(A, B)$;
- Second order statistics for the number of deletions at position i and amino acid types at position j;
- Second order statistics for the number of deletions at different positions.

We implement this ablation in terms of a set of small interventions in our system. We will show that with these interventions all provided MSA information can be directly derived from the features listed above.

In the data pipeline, we set the number of clusters $N_{\text{clust}} = 1$ so that the MSA representation only has a single row. The first cluster is always set to the sequence of the modelled protein. Furthermore in the clustering stage all sequences in the MSA will be assigned to this single cluster, this means the cluster profile feature is just the full MSA profile.

We also remove the template torsion angle features that otherwise would be concatenated as additional rows. Since columnwise MSA attention would be trivial in this setup, we remove this operation from the Evoformer stack.

Furthermore we replace the extra MSA stack (Algorithm 18) with a single outer product mean (Algorithm 10) on the extra MSA features $\{\mathbf{f}_{s_e i}^{\text{extra\_msa\_feat}}\}$ and use it to initialize the pair representation.

We denote the set of features (MSA sequences and deletions) usually passed to the extra MSA stack by $\mathbf{f}_{si}$ and analyse what information of the MSA can be accessed by the model. Applying the outer product mean operation on $\mathbf{f}_{si}$ yields linear projections of the following kind of terms: $\left\{\sum_s \mathbf{f}_{si} \otimes \mathbf{f}_{sj}, \sum_s \mathbf{f}_{si},\ \sum_s \mathbf{f}_{sj}\right\}$. The first kind of terms comes from the outer product on the linearly projected features, the other two come from the mixed bias and features terms in the outer product, since we use linear transformations with a bias. When writing out the MSA features explicitly, we see that the first term contains the pairwise frequencies. Additionally, we have the second order statistics involving deletions. The other two terms result in an outer sum of profiles and expected number of deletions.

No triangles, biasing, or gating (use axial attention). First, we remove the triangle inductive bias. For this we replace the pair bias b_{ij}^h in Algorithm 13 and Algorithm 14 with a projection of the relative distances along the chain, i.e. relpos_{ij} from Algorithm 4. We also replace the two triangular multiplicative updates (Algorithm 11 and Algorithm 12) with two self-attention layers identical to the ones described above.

Furthermore, we replace the pair bias in the row-wise MSA attention Algorithm 7 with the projected relative distances relpos_{ij} and remove the gating in all attention operations. After these modifications, the attention used in the main part of the model closely matches the standard axial (or criss-cross) attention [119].

These modifications also imply that the pair representation does not influence the MSA representation directly. Though it is important to note that we keep Algorithm 10 without any modifications, so the MSA representation still influences the pair representation. We also do not modify the structure module, as such the pair representation still participates in building the structure.

No recycling. We remove recycling (see subsection 1.10) during training and inference, i.e. we only make a single pass through the model.

No invariant point attention (use direct projection). We replace the prediction of backbone frames, including all of the IPA Algorithm 22, with a linear direct projection of the backbone frame similar to Algorithm 23. The direct projection consists of linear layer applied to the final Evoformer single embedding to produce a $N_{\text{res}} \times 7$ matrix. The seven coordinates are converted into a backbone frame in the following way: The first 4 coordinates (q_w, q_x, q_y, q_z) are converted to the rotation matrix by creating a normalized quaternion

$$(a, b, c, d) = (1 + q_w, q_x, q_y, q_z)/\sqrt{(1 + q_w)^2 + q_x^2 + q_y^2 + q_z^2}\,, \tag{55}$$

and applying the standard formula for converting a quaternion to a rotation matrix:

$$R = \begin{pmatrix} a^2+b^2-c^2-d^2 & 2bc-2ad & 2bd+2ac \\ 2bc+2ad & a^2-b^2+c^2-d^2 & 2cd-2ab \\ 2bd-2ac & 2cd+2ab & a^2-b^2-c^2+d^2 \end{pmatrix}. \tag{56}$$

The last 3 predicted coordinates (t_x, t_y, t_z) are used as the translation vector. We also apply the side chain torsion module directly to the single embedding in the same manner as in the full model. In this ablation, there is no iterative process in the Structure module. Additionally, the final pair representation is not used in any way in the Structure module in this ablation but is used for the distogram loss.

No invariant point attention and no recycling. This ablation simply combines the changes of the two individual ablations described above.

No end-to-end structure gradients (keep auxiliary heads). We stop the gradients from the structure module into the main part of the network (i.e. the structure losses have no effect on Evoformer parameters), which means the Evoformer embeddings are trained only using the distogram loss and other auxiliary losses.

No auxiliary distogram head. We remove the auxiliary loss that predicts distograms (see subsubsection 1.9.8).

No auxiliary masked MSA head. We remove the auxiliary BERT-style loss that imputes the masked values in the MSA (see subsubsection 1.9.9). The ablation still uses the same MSA processing, i.e. the MSA is still randomly masked.

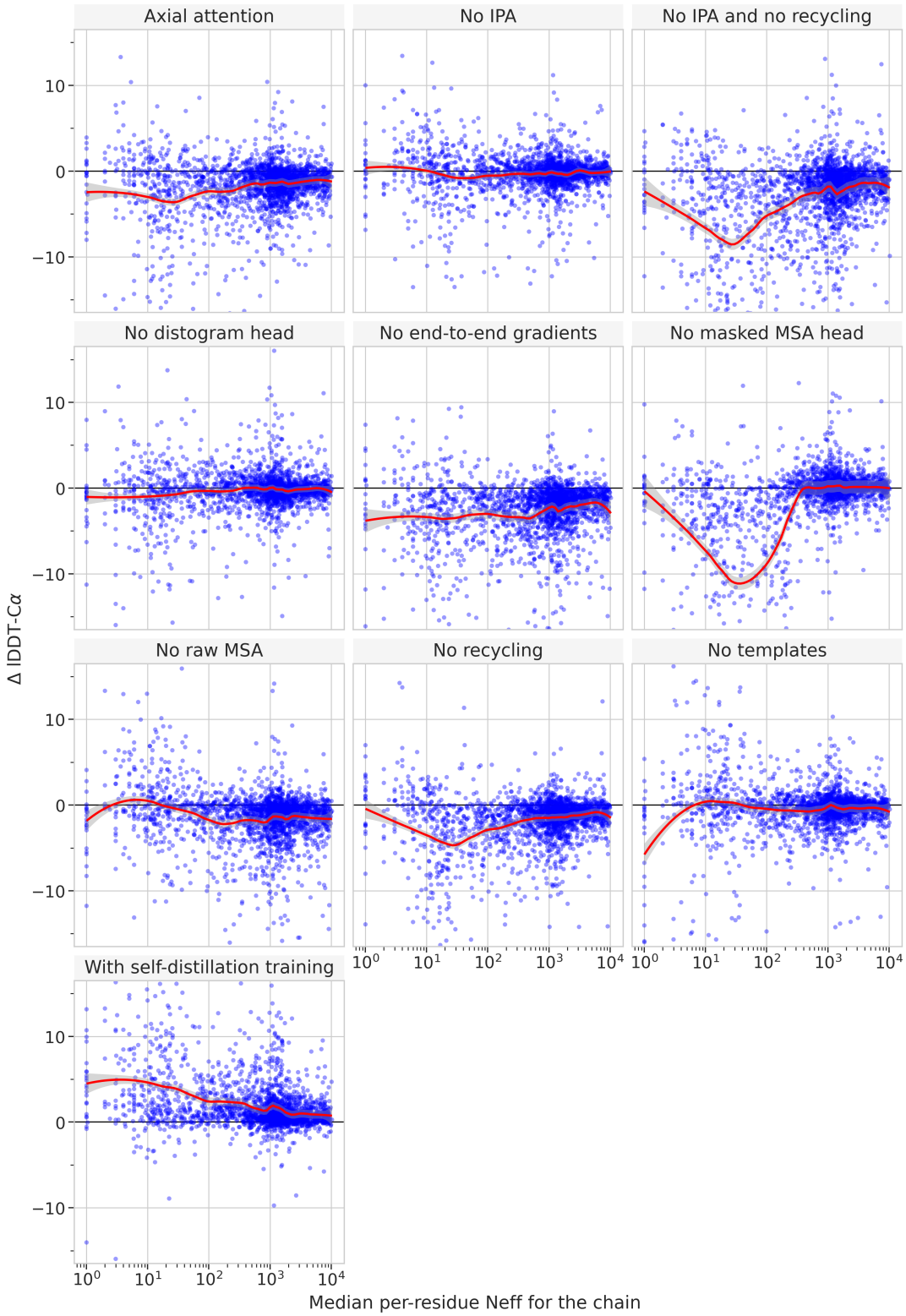
1.13.2 Procedure

Models training. For all ablations we kept hyperparameters from the main model configuration, which we have not re-tuned. We trained the models in two stages following the main training protocol shown in Table 4. All models apart from one ablation were trained without the distillation dataset. Including examples from the full AlphaFold distillation dataset would mean leaking architectural choices of the models that were used to create this dataset, while re-creating the dataset for each individual ablation would be too computationally expensive. For each ablation we trained 3 or 4 identical models with different random seeds and discarded any unstable runs using the validation set from subsubsection 1.11.7.

Datasets and metrics. We used two different test sets for ablations analysis. Firstly, we used the set of CASP14 domains described in subsubsection 1.12.2 and evaluated individual domain predictions with the Global Distance Test (GDT) [120]. Secondly, we used the redundancy-reduced set of recent PDB chains described in Methods, further restricted to protein chains with template coverage $\leq 30\%$ at 30% identity ($N = 2261$ targets). We scored the full chain predictions with lDDT-Cα [99].

1.13.3 Results

Ablation results are presented in Fig. 4b of the main paper. We show the difference in mean backbone accuracy for each ablation relative to an average of baseline seeds using both the CASP14 targets and the redundancy- and template-reduced set from recent PDB. Confidence intervals for each training seed are given by bootstrap over domains for CASP and full chains for PDB. Additionally, in Suppl. Fig. 10 we find that the different ablations have quite non-uniform dependence on the MSA depth. Some ablations like No Masked MSA head primarily affect shallow MSAs whereas others such as No Recycling have effects at all MSA depths.



Supplementary Figure 10 | Ablations accuracy relative to the baseline for different values of the MSA depth on the recent PDB set of chains, filtered by template coverage $\leq 30\%$ at 30% identity ($N = 2261$ protein chains). MSA depth is computed by counting the number of non-gap residues for each position in the MSA (using the Neff weighting scheme with a threshold of 80% identity measured on the region that is non-gap in either sequence) and taking the median across residues. Plots are restricted to the range [-15, 15]; the red lines represent LOESS mean estimates with 95% confidence intervals for the LOESS.

The presented ablations remove relatively independent components of the system, such as end-to-end training, iterative structure refinement, triangular inductive bias, and others from the no-self-distillation baseline. While this analysis helps determine the relative importance of different ideas in the model, there are two major limitations. First, hyperparameters were not re-tuned when performing the ablations which could make ablations appear more significant than in a properly tuned model. Second, interactions between different components may be strongly non-linear as the components of the AlphaFold system can have overlapping or interacting roles. As a dramatic example, removal of Invariant Point Attention (IPA) from the model has only a tiny effect on accuracy for recent PDB, but removing both recycling and IPA from the model has a much larger effect than removing recycling alone. This double-ablation must be interpreted with caution, however, as it is not clear which missing features of recycling (embedding of intermediate structures, much deeper network, intermediate losses, or weight tying) explain the severe ablation.

In general, the ablations confirm that many architecture and training features have an effect on final AlphaFold accuracy and the relative magnitude of effects are broadly consistent whether measured on the human-curated CASP set or the much larger recent PDB set.

The strong contributions of self-distillation training and masked MSA losses suggest that these techniques are making effective use of unlabelled data within the supervised training framework (and despite the lack of unsupervised pre-training). The various architecture ablations confirm that many different architectural innovations contribute to the final AlphaFold accuracy. Finally, careful handling of the structure and intermediate losses (using both end-to-end structure gradients and providing intermediate loss gradients in recycling) is important to achieving full accuracy. This is likely due to pushing the network towards having a concrete representation of the structure as appears to be present in the trajectories of intermediate structure predictions.

A surprisingly small role is played by the distogram head, suggesting that it is not a necessary output to obtain high-accuracy structures. Another surprise is that equivariance in the network is not essential as the single ablation of IPA, which removes all equivariant components, is still a very accurate structure prediction network. Similarly, the single ablation of raw MSA leads to a drop in accuracy but it is not catastrophic. We hypothesize though that each of these features would show much larger effects within multiple ablations similar to the effect seen in ablating both IPA and recycling.

1.14 Network probing details

This paper includes supplementary videos with structures evolving over 4 recycling iterations and 48 Evoformer blocks. Static versions of those videos are also presented as GDT curves per $4 \cdot 48 = 192$ intermediate points and discussed in the main part of the paper. In this section we provide additional details about how these results were produced.

We started with a full network architecture and added additional probing modules after each Evoformer block. Concretely, we incorporated copies of the Structure module (subsection 1.8) with the same architecture and configuration and optimized the probing loss $\mathcal{L}_{\text{probe}}$, defined as the average of the losses of these modules. For inputs, we provided the current versions of the MSA and pair representations produced up to this point in the network. As with the main network, we had different trainable parameters across Evoformer blocks, but tied across recycling iterations. In other words, we trained 48 additional sets of weights for these modules. Note that we have not attached any probing modules to the template pair stack or the extra MSA stack. Importantly, when training the probing network, we stopped all gradients coming to the main network parameters both from the probing and the main losses. Outputs of the probing modules have not been recycled in any way, i.e. they were used for the analysis only.

In the presented analysis, we were probing one of the five CASP14 models (model 1.1.2 from Table 5 to be precise). We used the same data sizes as in this model, i.e. $N_{\text{templ}} = 4$, $N_{\text{res}} = 384$, $N_{\text{seq}} = 512$, and $N_{\text{extra\_msa}} = 1024$. For the learning rate, we used the standard schedule from our training stage (see subsubsection 1.11.3). Following the main training stage setup, we have not used violation losses (subsubsection 1.9.11) for the probing. In the evaluator (subsubsection 1.11.7) we used the validation probing loss $\mathcal{L}_{\text{probe}}$ for the model selection and we changed the exponential moving average decay to 0.99 for a faster training

startup. We trained the probing network for about 5 days until convergence. We used $N_{\text{ensemble}} = 1$ both for the build-in evaluator and for the presented inference to simplify setup and reduce memory usage.

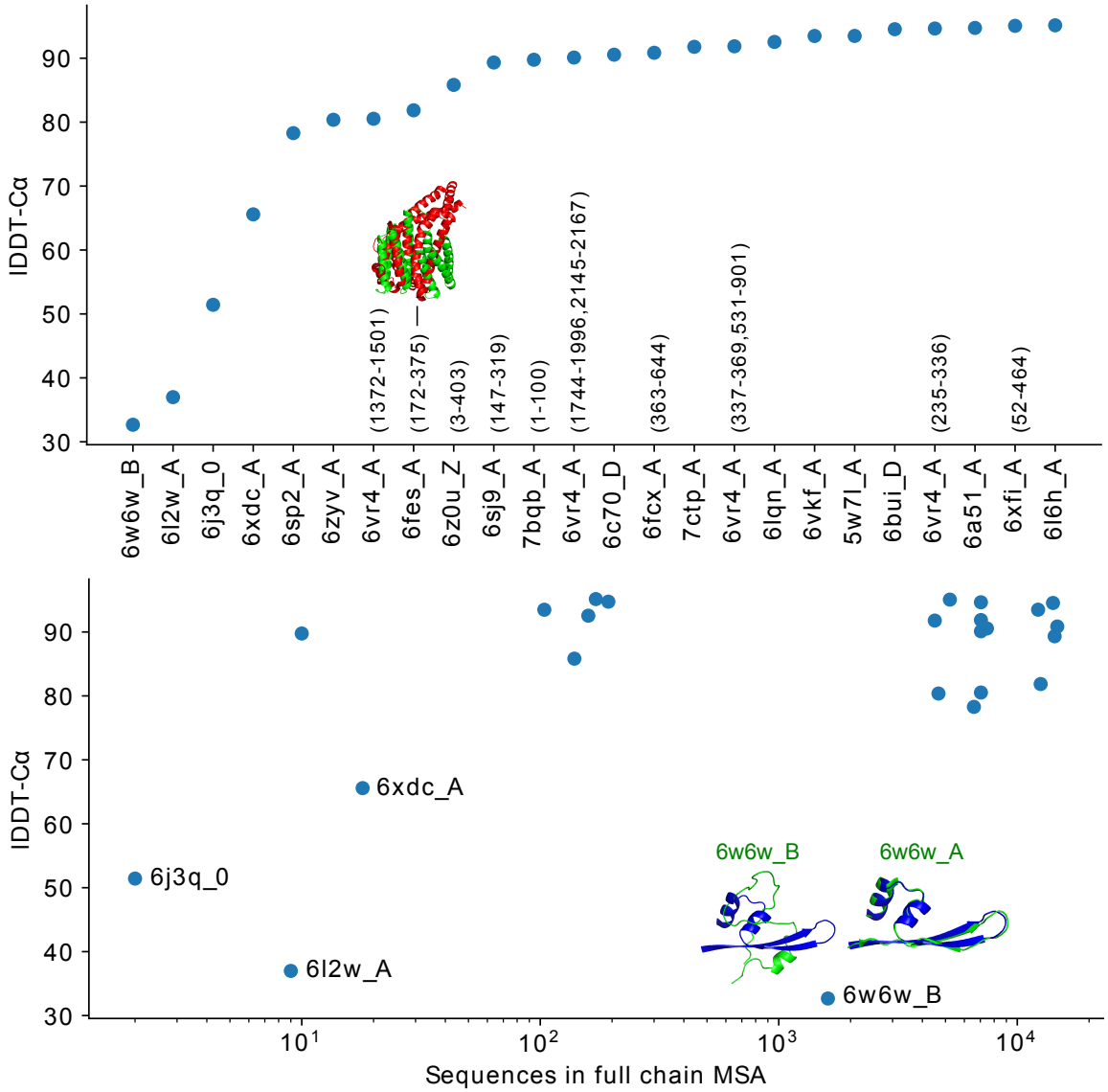
1.15 Novel fold performance

Across CASP14 and CASP_Commons, five of the assessed targets were sufficiently distinct from prior PDB entries to be called novel folds. Targets T1035, T1037, T1040 and T1042 (PDB:6vr4) were highlighted as new folds by the CASP14 assessors [121], while C1905 (PDB:6xdc) is called novel in its primary citation [122]. Here we evaluate AlphaFold's performance on an expanded test set of recent structures selected for their novelty.

Three heuristics were used to identify recent PDB chains that might be novel with respect to AlphaFold's training set:

1. The PDB abstract or structure title contains the phrase "novel fold".
2. The chain contains a Pfam domain from a clade not associated with any structure released before 2018-04-30.
3. The chain maps to a SCOP fold not associated with any structure released before 2018-04-30.

The SIFTS resource was used to map PDB chains to Pfam and SCOP [123], with clade and fold annotations downloaded from the respective project websites [124, 125]. Where multiple similar structures were surfaced from the same study only one was kept, and trivial structures consisting of a single helix were also discarded. Where it was clear from the literature that a specific domain was novel, or where the chain consisted of multiple domains joined by an unresolved linker, the structure was cropped to a more reasonable unit for TM-align comparison. The final test set was selected by TM-align comparison against all chains in the training set [104]. Any target with a TM-score greater than 0.6 was eliminated (normalising with respect to the length of the candidate novel structure). Of the remaining 24 targets, 15 have a TM-score of less than 0.5 to the closest training set chain.



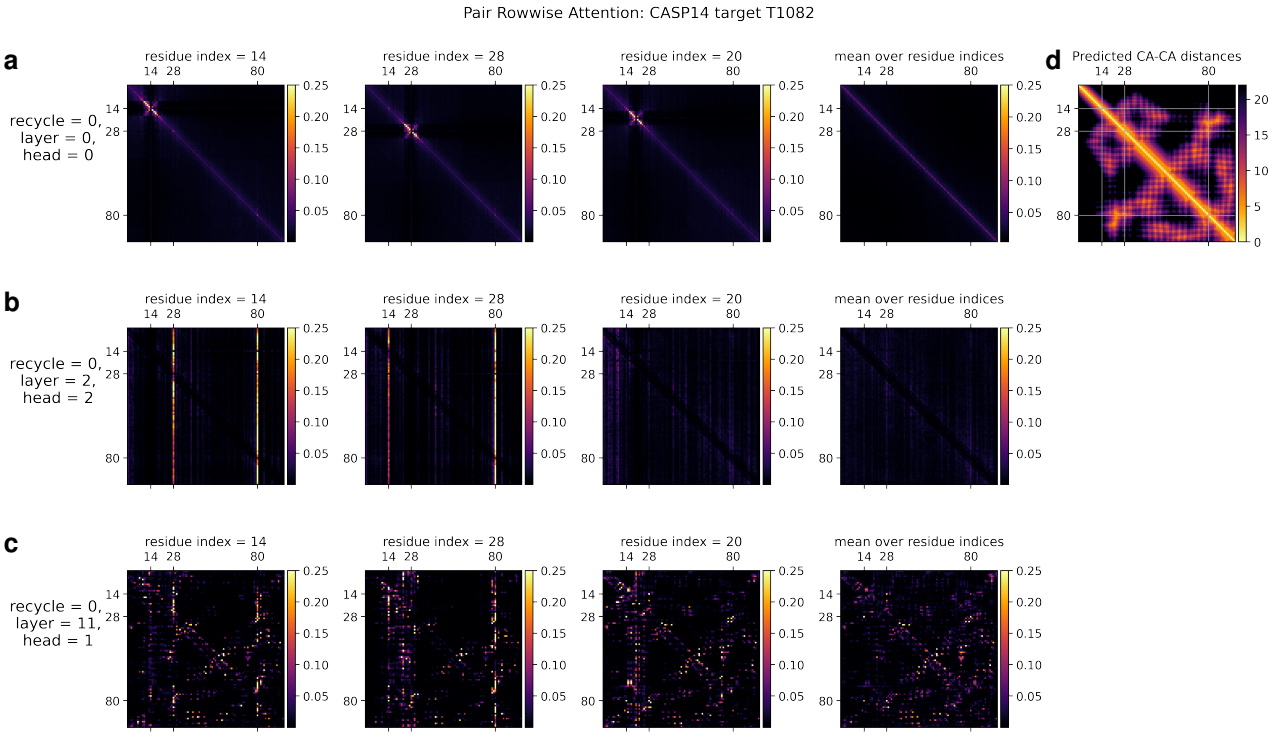
Supplementary Figure 11 | Performance on a set of novel structures ($N = 24$ protein chains). Where the target is not a full chain, the evaluated region is given as residue ranges in PDB numbering. 6fes_A has the highest TM-score to a training set chain (0.57); the structure comparison shows this target (green) aligned to its closest match with cealign [126] (5jxf_B, red, cropped for easier comparison). The bottom graph shows the relationship between performance and full chain MSA depth, with low lDDT-Cα outliers labelled. For outlier 6w6w_B, we also show the prediction (blue) aligned to two possible ground truth structures (green): 6w6w_B and the corresponding subsequence of 6w6w_A which strongly suggests that AlphaFold has simply produced an alternative conformation of this sequence.

Prediction used $N_{\text{ensemble}} = 3$ and disallowed templates released after 2018-04-30, but otherwise followed our CASP14 procedure. The input to AlphaFold was the full chain fasta sequence for each target, extracted from the relevant mmCIF file. Figure 11 shows the results. The median lDDT-Cα on this set was 90.3, with mean lDDT-Cα 82.6. Investigating the outliers, we found that almost all could be explained by a small MSA containing fewer than 20 sequences. In the case of 6w6w_B, the ground truth structure may be an alternate conformation adopted by this subsequence in complex. The AlphaFold prediction closely matches the conformation seen in 6w6w_A, which forms part of a larger domain. We conclude that AlphaFold is able to predict novel structures well, subject to the same constraints described elsewhere in this work (a sufficiently large MSA, and minimal dependence of the conformation on missing context from surrounding chains).

1.16 Visualization of attention

In this section we are going to analyse a small subset of the attention patterns we see in the main part of the model. This will be restricted to relatively short proteins with fairly shallow MSA's in order for easier visualization.

We are going to examine the attention weights in the attention on the pair representation in the "Triangular gated self-attention around starting node" (Algorithm 13). The attention pattern for a given head h is a third order tensor a^h_{ijk} (line 5 of Algorithm 13), here we will investigate different slices along the i axis as well as averages along this axis, and display the jk array as a heat map. We specify the attention pattern for a specific head by the recycling index r, layer index l and head index h. We show the attention patterns in Suppl. Fig. 12



Supplementary Figure 12 | Visualization of row-wise pair attention. (**a**) Attention patterns in layer 0, head 0. (**b**) Attention patterns in layer 2, head 2. (**c**) Attention patterns in layer 11, head 1. (**d**) Predicted Cα – Cα distances.

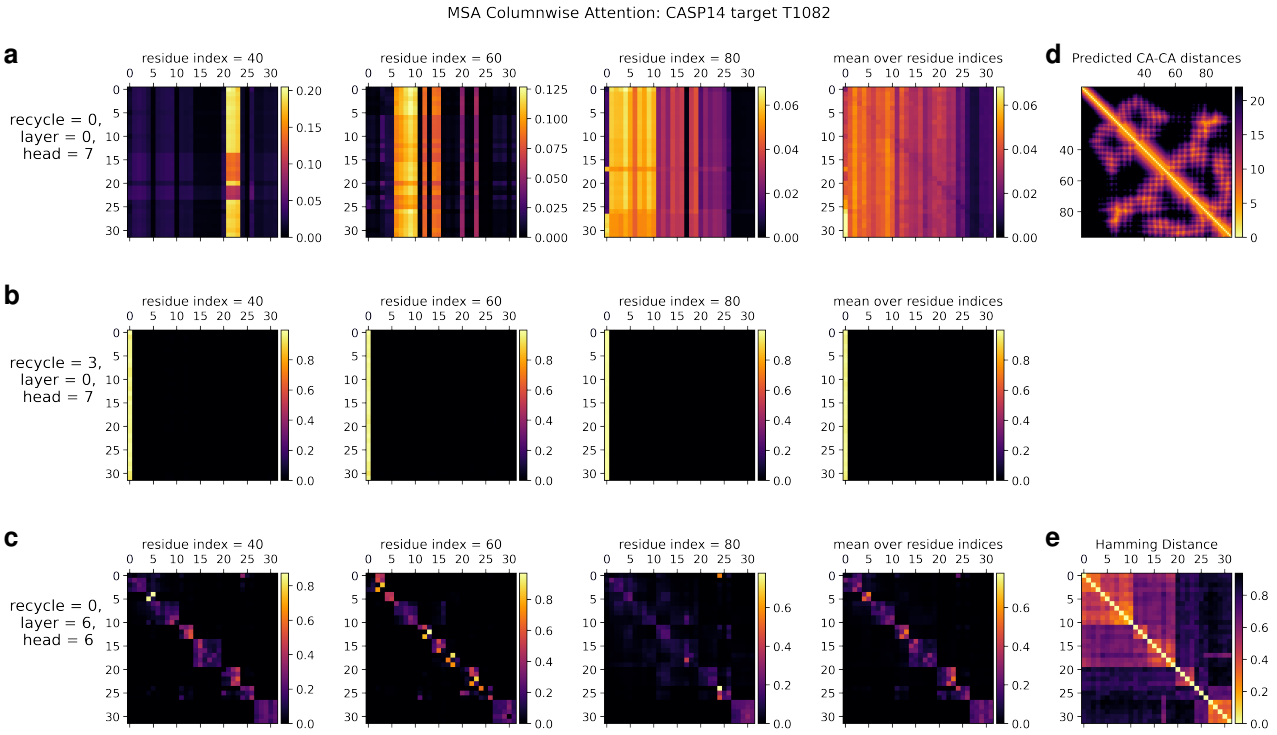
In layer 0, head 0 (Suppl. Fig. 12a) we see a pattern that is similar to a convolution, i.e. the pair representation at $(i, i+j)$ attends to the pair representation $(i, i+k)$, where j and k are relatively small and the pattern is fairly independent of i. The radius of the pattern is 4 residues, we note that the hydrogen bonded residues in an alpha helix are exactly 4 residues apart. Patterns like this are expected in early layers due to the relative position encoding, however one would naively expect it to be wider, as the relative positions run to 32.

In layer 2, head 2 (Suppl. Fig. 12b) we see a very specific attention pattern. We see that positions $14, 28$ and 80 play a special role. These correspond to the positions of cysteine residues in the protein. We see that at the cysteine positions the attention is localized to the other cysteine positions as exemplified by the bands in the first two panels. Meanwhile away at positions different from the cysteine like position 20, we see that the attention is devoid of features suggesting that the model does not use the attention at these positions. We found this behaviour to be consistent across different positions and proteins, this possibly indicates finding possible disulfides as a key feature of some heads.

A head later in the network (Suppl. Fig. 12c) shows a rich, non-local attention pattern resembling the

distance between pairs in the structure (Suppl. Fig. 12d).

In Suppl. Fig. 13 we show a visualization of the attention pattern in the MSA along the columns a^h_{sti} (line 4 of Algorithm 8). We slice along the last axis i and display the st array as heat map.



Supplementary Figure 13 | Visualization of attention in the MSA along sequences. (**a**) Attention patterns in layer 0, head 7. (**b**) Attention patterns in layer 0, head 7 in the last recycling iteration. (**c**) Attention patterns in layer 6, head 6. (**d**) Predicted Cα – Cα distances (**e**) Hamming distances between the sequences of the MSA.

The original MSA subset shown to the main part of the model is randomly sampled. As such the order is random except for the first row. The first row is special because it contains the target sequence and is recycled in consecutive iterations of the model. Due to the shallow MSA of this protein, the random subset leads to a random permutation of the sequences. In order to facilitate easier interpretation of the attention patterns here we reorder the attention tensor by using a more suitable order for the MSA. We perform a hierarchical clustering using the Ward method with simple Hamming distance as a metric and use the output to re-index the sequence dimension in the MSA attention tensor. We resort the indices from the hierarchical clustering manually to keep the target sequence in the first row. This manual sorting is done in such a way as to keep the tree structure valid. The Hamming distances between the reordered sequences (see Suppl. Fig. 13e) show a block-like structure quite clearly after the reordering.

The attention pattern in the first layer of the network in the first recycling iteration (e.g. layer 0, head 7 in Suppl. Fig. 13a) is not very informative and is largely averaging as can be seen by looking at the range of the attention weights.

In the same head, but in the last recycling iteration (Suppl. Fig. 13b) we see that all sequences at all positions attend to the first row. Therefore this head behaves differently upon recycling and is presumably important for distribution the information in the recycled first row to the rest of the MSA representation.

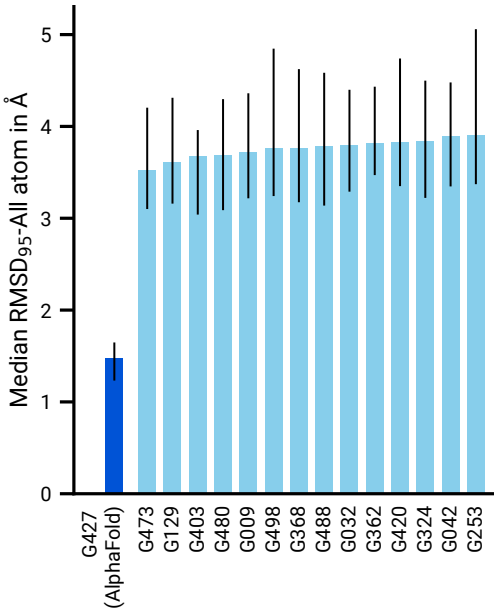
Layer 6, head 6 (Suppl. Fig. 13c) shows a pattern that is fairly common in the column-wise MSA attention, here the pattern only varies lightly as one goes along the sequence and there is a clear structure in blocks of sequences that attend to each other. We note that these seem somewhat similar to the blocks derived from

hierarchical clustering using Hamming distance.

Whether attention patterns provide a good explanation for the behaviour of a given model or are predictive of interventions into the model is a topic of debate in the community, see [127], [128], [129]. A detailed analysis of the generality and predictivity of these attention patterns is beyond the scope of this paper.

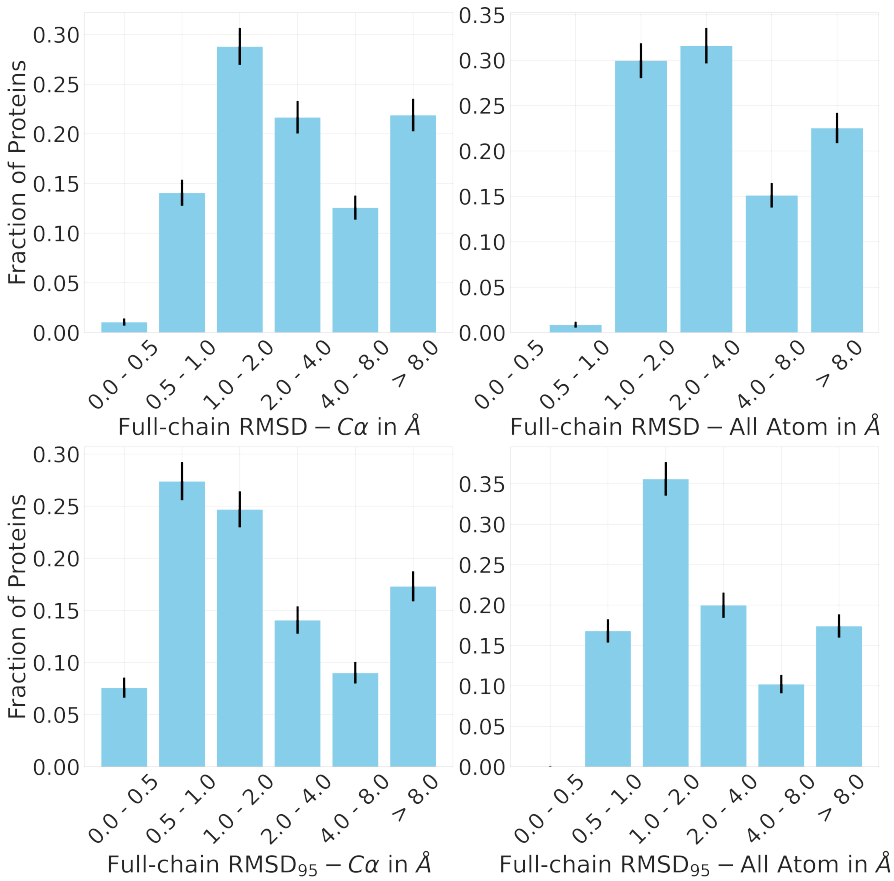
1.17 Additional results

We present median all-atom RMSD_{95} on the CASP14 set in Suppl. Fig. 14. Figure 1 of the main paper shows the corresponding results on just the Cα atoms (RMSD_{95}-Cα).



Supplementary Figure 14 | Median all-atom RMSD_{95} on the CASP14 set of protein domains ($N = 87$ protein domains) relative to the top-15 entries (out of 146), group numbers correspond to the numbers assigned to entrants by CASP; error bars represent the 95% confidence interval of the median, estimated with 10,000 bootstrap samples.

In addition to the main Figure 2a, we show all-atom RMSD and RMSD for 100% coverage in Suppl. Fig. 15. We also report the quartiles of the distributions in Table 6. For this analysis, the recent PDB set of 10795 chains described in Methods was further filtered to exclude proteins with a template (identified by hmmsearch) from the training set with more than 40% sequence identity covering more than 1% of the chain.



Supplementary Figure 15 | Histograms of all-atom and backbone RMSD at 95% and 100% coverage for full-chain predictions on the template-reduced set ($N = 3144$ protein chains). Error bars are 95% confidence intervals (Poisson).

Table 6 | Quartiles of distributions of all-atom and backbone RMSD at 95% and 100% coverage for full-chain predictions on the template-reduced set ($N = 3144$ protein chains).

Quantity (in Å)	lower quart.	median	upper quart.
RMSD - Cα	1.31	2.32	6.49
RMSD - All Atom.	1.80	2.80	6.78
RMSD_{95} - Cα	0.79	1.46	4.33
RMSD_{95} - All Atom	1.17	1.89	4.72

References

[85] Jimmy Ba, Jamie R Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[86] L Steven Johnson, Sean R Eddy, and Elon Portugaly. Hidden markov model speed heuristic and iterative hmm search procedure. *BMC Bioinformatics*, 11(1):1–8, 2010.

[87] Michael Remmert, Andreas Biegert, Andreas Hauser, and Johannes Söding. Hhblits: lightning-fast iterative protein sequence searching by hmm-hmm alignment. *Nature Methods*, 9(2):173–175, 2012.

[88] Alex L Mitchell, Alexandre Almeida, Martin Beracochea, Miguel Boland, Josephine Burgin, Guy Cochrane, Michael R Crusoe, Varsha Kale, Simon C Potter, Lorna J Richardson, Ekaterina Sakharova, Maxim Scheremetjew, Anton Korobeynikov, Alex Shlemov, Olga Kunyavskaya, Alla Lapidus, and Robert D Finn. MGnify: the microbiome analysis resource in 2020. *Nucleic Acids Research*, 48(D1):D570–D578, 11 2019.

[89] Baris E Suzek, Yuqi Wang, Hongzhan Huang, Peter B McGarvey, Cathy H Wu, and UniProt Consortium. Uniref clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics*, 31(6):926–932, 2015.

[90] Milot Mirdita, Lars von den Driesch, Clovis Galiez, Maria J Martin, Johannes Söding, and Martin Steinegger. Uniclust databases of clustered and deeply annotated protein sequences and alignments. *Nucleic Acids Research*, 45(D1):D170–D176, 2017.

[91] Martin Steinegger, Markus Meier, Milot Mirdita, Harald Vöhringer, Stephan J Haunsberger, and Johannes Söding. Hh-suite3 for fast remote homology detection and deep protein annotation. *BMC Bioinformatics*, 20(1):1–15, 2019.

[92] Timo Lassmann, Oliver Frings, and Erik L L Sonnhammer. Kalign2: high-performance multiple alignment of protein and nucleotide sequences allowing external features. *Nucleic Acids Research*, 37:858–865, 2009.

[93] Martin Steinegger and Johannes Söding. Clustering huge protein sequence sets in linear time. *Nature Communications*, 9(1):1–8, 2018.

[94] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10687–10698, 2020.

[95] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *31st Conference on Neural Information Processing Systems (NeurIPS 2017)*, pages 6000–6010, 2017.

[96] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy, July 2019. Association for Computational Linguistics.

[97] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[98] RA Engh and R Huber. Structure quality and target parameters. In *International Tables for Crystallography, Vol. F*. John Wiley & Sons, Ltd, 2006.

[99] Valerio Mariani, Marco Biasini, Alessandro Barbato, and Torsten Schwede. lddt: A local superposition-free score for comparing protein structures and models using distance difference tests. *Bioinformatics*, 29(21):2722–2728, 2013.

[100] Viktor Hornak, Robert Abel, Asim Okur, Bentley Strockbine, Adrian Roitberg, and Carlos Simmerling. Comparison of multiple amber force fields and development of improved protein backbone parameters. *Proteins: Structure, Function, and Bioinformatics*, 65(3):712–725, 2006.

[101] Peter Eastman, Jason Swails, John D Chodera, Robert T McGibbon, Yutong Zhao, Kyle A Beauchamp, Lee-Ping Wang, Andrew C Simmonett, Matthew P Harrigan, Chaya D Stern, Rafal P Wiewiora, Bernard R Brooks, and Vijay S Pande. Openmm 7: Rapid development of high performance algorithms for molecular dynamics. *PLOS Computational Biology*, 13(7):1–17, 07 2017.

[102] Mohammed AlQuraishi. End-to-End Differentiable Learning of Protein Structure. *Cell Systems*, 8(4):292–301.e3, 24 April 2019.

[103] Yang Zhang and Jeffrey Skolnick. Scoring function for automated assessment of protein structure template quality. *Proteins: Structure, Function, and Bioinformatics*, 57(4):702–710, 2004.

[104] Yang Zhang and Jeffrey Skolnick. TM-align: a protein structure alignment algorithm based on the TM-score. *Nucleic acids research*, 33(7):2302–2309, 2005.

[105] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[106] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, 2015.

[107] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the International Conference on Machine Learning*, pages 1310–1318, 2013.

[108] Y. LeCun, L. Bottou, G. Orr, and K. Muller. Efficient backprop. In *Neural Networks: Tricks of the Trade*. Springer, 1998.

[109] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE Conference on Computer Vision*, pages 1026–1034, 2015.

[110] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[111] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[112] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

[113] Jürgen Haas, Alessandro Barbato, Dario Behringer, Gabriel Studer, Steven Roth, Martino Bertoni, Khaled Mostaguir, Rafal Gumienny, and Torsten Schwede. Continuous automated model evaluation (cameo) complementing the critical assessment of structure prediction in casp12. *Proteins: Structure, Function, and Bioinformatics*, 86:387–398, 2018.

[114] Andreas Griewank and Andrea Walther. Algorithm 799: revolve: an implementation of checkpointing for the reverse or adjoint mode of computational differentiation. *ACM Transactions on Mathematical Software (TOMS)*, 26(1):19–45, 2000.

[115] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

[116] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *Proceedings of the International Conference on Learning Representations*, 2020.

[117] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Kathryn Tunyasuvunakool, Olaf Ronneberger, Russ Bates, Augustin Žídek, Alex Bridgland, Clemens Meyer, Simon A A Kohl, Anna Potapenko, Andrew J Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Martin Steinegger, Michalina Pacholska, David Silver, Oriol Vinyals, Andrew W Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. High accuracy protein structure prediction using deep learning. In *Fourteenth Critical Assessment of Techniques for Protein Structure Prediction (Abstract Book)*, pages 22–24, 2020.

[118] Jianyi Yang, Ivan Anishchenko, Hahnbeom Park, Zhenling Peng, Sergey Ovchinnikov, and David Baker. Improved protein structure prediction using predicted interresidue orientations. *Proceedings of the National Academy of Sciences of the United States of America*, 117(3):1496–1503, 21 January 2020.

[119] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 603–612, 2019.

[120] Adam Zemla. LGA – a method for finding 3d similarities in protein structures. *Nucleic Acids Research*, 31:3370–4, 08 2003.

[121] Lisa Kinch, Andriy Kryshtafovych, and Nick Grishin. Target classification in the 14th round of the critical assessment of protein structure prediction (CASP14). https://predictioncenter.org/casp14/doc/presentations/2020_11_30_TargetClassification_Kinch_Updated.pdf, 2020. [Online; accessed 08-June-2021].

[122] David M. Kern, Ben Sorum, Sonali S. Mali, Christopher M. Hoel, Savitha Sridharan, Jonathan P. Remis, Daniel B. Toso, Abhay Kotecha, Diana M. Bautista, and Stephen G. Brohawn. Cryo-EM structure of the SARS-CoV-2 3a ion channel in lipid nanodiscs. *bioRxiv preprint bioRxiv:10.1101/2020.06.17.156554v3*, 2021.

[123] Jose M Dana, Aleksandras Gutmanas, Nidhi Tyagi, Guoying Qi, Claire O'Donovan, Maria Martin, and Sameer Velankar. SIFTS: updated structure integration with function, taxonomy and sequences resource allows 40-fold increase in coverage of structure-based annotations for proteins. *Nucleic acids research*, 47(D1):D482–D489, 2019.

[124] Jaina Mistry, Sara Chuguransky, Lowri Williams, Matloob Qureshi, Gustavo A Salazar, Erik LL Sonnhammer, Silvio CE Tosatto, Lisanna Paladin, Shriya Raj, Lorna J Richardson, et al. Pfam: The protein families database in 2021. *Nucleic Acids Research*, 49(D1):D412–D419, 2021.

[125] Antonina Andreeva, Eugene Kulesha, Julian Gough, and Alexey G Murzin. The SCOP database in 2020: expanded classification of representative family and superfamily domains of known protein structures. *Nucleic acids research*, 48(D1):D376–D382, 2020.

[126] Ilya N Shindyalov and Philip E Bourne. Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. *Protein engineering*, 11(9):739–747, 1998.

[127] Sarthak Jain and Byron C Wallace. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[128] Sofia Serrano and Noah A Smith. Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2931–2951, Florence, Italy, July 2019. Association for Computational Linguistics.

[129] Sarah Wiegreffe and Yuval Pinter. Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20, Hong Kong, China, November 2019. Association for Computational Linguistics.

Article

Improved protein structure prediction using potentials from deep learning

https://doi.org/10.1038/s41586-019-1923-7

Received: 2 April 2019

Accepted: 10 December 2019

Published online: 15 January 2020

Andrew W. Senior[1,4]*, Richard Evans[1,4], John Jumper[1,4], James Kirkpatrick[1,4], Laurent Sifre[1,4], Tim Green[1], Chongli Qin[1], Augustin Žídek[1], Alexander W. R. Nelson[1], Alex Bridgland[1], Hugo Penedones[1], Stig Petersen[1], Karen Simonyan[1], Steve Crossan[1], Pushmeet Kohli[1], David T. Jones[2,3], David Silver[1], Koray Kavukcuoglu[1] & Demis Hassabis[1]

Protein structure prediction can be used to determine the three-dimensional shape of a protein from its amino acid sequence[1]. This problem is of fundamental importance as the structure of a protein largely determines its function[2]; however, protein structures can be difficult to determine experimentally. Considerable progress has recently been made by leveraging genetic information. It is possible to infer which amino acid residues are in contact by analysing covariation in homologous sequences, which aids in the prediction of protein structures[3]. Here we show that we can train a neural network to make accurate predictions of the distances between pairs of residues, which convey more information about the structure than contact predictions. Using this information, we construct a potential of mean force[4] that can accurately describe the shape of a protein. We find that the resulting potential can be optimized by a simple gradient descent algorithm to generate structures without complex sampling procedures. The resulting system, named AlphaFold, achieves high accuracy, even for sequences with fewer homologous sequences. In the recent Critical Assessment of Protein Structure Prediction[5] (CASP13)—a blind assessment of the state of the field—AlphaFold created high-accuracy structures (with template modelling (TM) scores[6] of 0.7 or higher) for 24 out of 43 free modelling domains, whereas the next best method, which used sampling and contact information, achieved such accuracy for only 14 out of 43 domains. AlphaFold represents a considerable advance in protein-structure prediction. We expect this increased accuracy to enable insights into the function and malfunction of proteins, especially in cases for which no structures for homologous proteins have been experimentally determined[7].

Proteins are at the core of most biological processes. As the function of a protein is dependent on its structure, understanding protein structures has been a grand challenge in biology for decades. Although several experimental structure determination techniques have been developed and improved in accuracy, they remain difficult and time-consuming[2]. As a result, decades of theoretical work has attempted to predict protein structures from amino acid sequences.

CASP[5] is a biennial blind protein structure prediction assessment run by the structure prediction community to benchmark progress in accuracy. In 2018, AlphaFold joined 97 groups from around the world in entering CASP13[8]. Each group submitted up to 5 structure predictions for each of 84 protein sequences for which experimentally determined structures were sequestered. Assessors divided the proteins into 104 domains for scoring and classified each as being amenable to template-based modelling (TBM, in which a protein with a similar sequence has a known structure, and that homologous structure is modified in accordance with the sequence differences) or requiring free modelling (FM, in cases in which no homologous structure is available), with an intermediate (FM/TBM) category. Figure 1a shows that AlphaFold predicts more FM domains with high accuracy than any other system, particularly in the 0.6–0.7 TM-score range. The TM score—ranging between 0 and 1—measures the degree of match of the overall (backbone) shape of a proposed structure to a native structure. The assessors ranked the 98 participating groups by the summed, capped *z*-scores of the structures, separated according to category. AlphaFold achieved a summed *z*-score of 52.8 in the FM category (best-of-five) compared with 36.6 for the next closest group (322). Combining FM and TBM/FM categories, AlphaFold scored 68.3 compared with 48.2. AlphaFold is able to predict previously unknown folds to high accuracy (Fig. 1b). Despite using only FM techniques and not using templates, AlphaFold also scored well in the TBM category according to the assessors' formula 0-capped *z*-score, ranking fourth for the top-one model or first for the best-of-five models. Much of the accuracy of AlphaFold is due to the accuracy of the distance predictions, which is evident from the high precision of the corresponding contact predictions (Fig. 1c and Extended Data Fig. 2a).

[1]DeepMind, London, UK. [2]The Francis Crick Institute, London, UK. [3]University College London, London, UK. [4]These authors contributed equally: Andrew W. Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre. *e-mail: andrewsenior@google.com

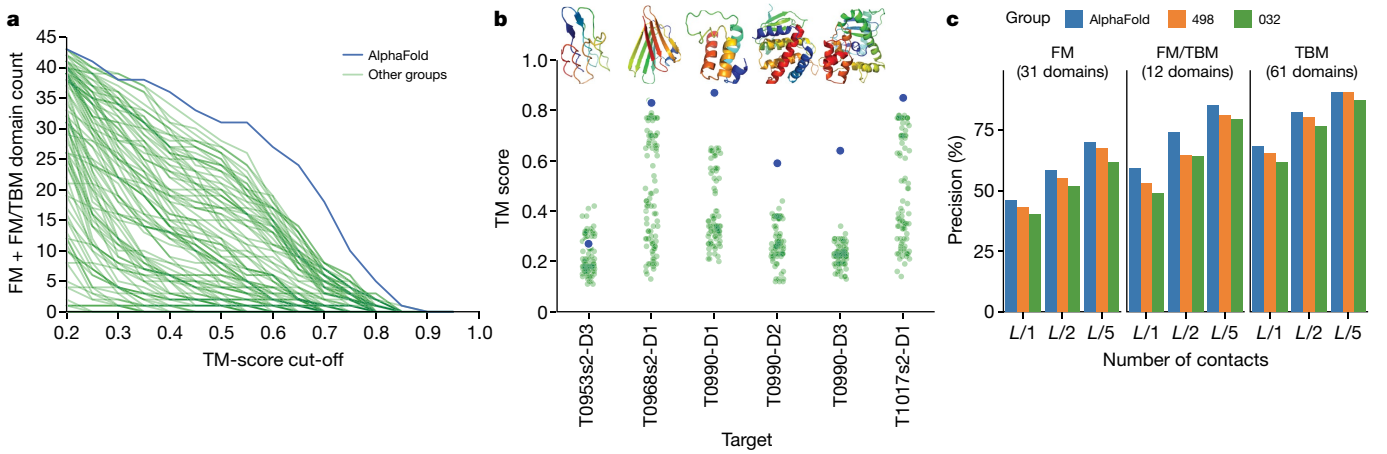


Fig. 1 | The performance of AlphaFold in the CASP13 assessment. a, Number of FM (FM + FM/TBM) domains predicted for a given TM-score threshold for AlphaFold and the other 97 groups. **b**, For the six new folds identified by the CASP13 assessors, the TM score of AlphaFold was compared with the other groups, together with the native structures. The structure of T1017s2-D1 is not available for publication. **c**, Precisions for long-range contact prediction in CASP13 for the most probable L, $L/2$ or $L/5$ contacts, where L is the length of the domain. The distance distributions used by AlphaFold in CASP13, thresholded to contact predictions, are compared with the submissions by the two best-ranked contact prediction methods in CASP13: 498 (RaptorX-Contact[26]) and 032 (TripletRes[32]) on 'all groups' targets, with updated domain definitions for T0953s2.

The most-successful FM approaches thus far[9–11] have relied on fragment assembly. In these approaches, a structure is created through a stochastic sampling process—such as simulated annealing[12]—that minimizes a statistical potential that is derived from summary statistics extracted from structures in the Protein Data Bank (PDB)[13]. In fragment assembly, a structure hypothesis is repeatedly modified, typically by changing the shape of a short section while retaining changes that lower the potential, ultimately leading to low potential structures. Simulated annealing requires many thousands of such moves and must be repeated many times to have good coverage of low-potential structures.

In recent years, the accuracy of structure predictions has improved through the use of evolutionary covariation data[14] that are found in sets of related sequences. Sequences that are similar to the target sequence are found by searching large datasets of protein sequences derived from DNA sequencing and aligned to the target sequence to generate a multiple sequence alignment (MSA). Correlated changes in the positions of two amino acid residues across the sequences of the MSA can be used to infer which residues might be in contact. Contacts are typically defined to occur when the β-carbon atoms of 2 residues are within 8 Å of one another. Several methods[15–18], including neural networks[19–22], have been used to predict the probability that a pair of residues is in contact based on features computed from MSAs. Contact predictions are incorporated in structure predictions by modifying the statistical potential to guide the folding process to structures that satisfy more of the predicted contacts[11,23]. Other studies[24,25] have used predictions of the distance between residues, particularly for distance geometry approaches[26–28]. Neural network distance predictions without covariation features were used to make the evolutionary pairwise distance-dependent statistical potential[25], which was used to rank structure hypotheses. In addition, the QUARK pipeline[11] used a template-based distance-profile restraint for TBM.

In this study, we present a deep-learning approach to protein structure prediction, the stages of which are illustrated in Fig. 2a. We show that it is possible to construct a learned, protein-specific potential by training a neural network (Fig. 2b) to make accurate predictions about the structure of the protein given its sequence, and to predict the structure itself accurately by minimizing the potential by gradient descent (Fig. 2c). The neural network predictions include backbone torsion angles and pairwise distances between residues. Distance predictions provide more specific information about the structure than contact predictions and provide a richer training signal for the neural network. By jointly predicting many distances, the network can propagate distance information that respects covariation, local structure and residue identities of nearby residues. The predicted probability distributions can be combined to form a simple, principled protein-specific potential. We show that with gradient descent, it is simple to find a set of torsion angles that minimizes this protein-specific potential using only limited sampling. We also show that whole chains can be optimized simultaneously, avoiding the need to segment long proteins into hypothesized domains that are modelled independently as is common practice (see Methods).

The central component of AlphaFold is a convolutional neural network that is trained on PDB structures to predict the distances d_{ij} between the C_β atoms of pairs, ij, of residues of a protein. On the basis of a representation of the amino acid sequence, S, of a protein and features derived from the MSA(S) of that sequence, the network, which is similar in structure to those used for image-recognition tasks[29], predicts a discrete probability distribution $P(d_{ij}|S, \mathrm{MSA}(S))$ for every ij pair in any 64 × 64 region of the $L \times L$ distance matrix, as shown in Fig. 2b. The full set of distance distribution predictions constructed by combining such predictions that covers the entire distance map is termed a distogram (from distance histogram). Example distogram predictions for one CASP protein, T0955, are shown in Fig. 3c, d. The modes of the distribution (Fig. 3c) can be seen to closely match the true distances (Fig. 3b). Example distributions for all distances to one residue (residue 29) are shown in Fig. 3d. We found that the predictions of the distance correlate well with the true distance between residues (Fig. 3e). Furthermore, the network also models the uncertainty in its predictions (Fig. 3f). When the s.d. of the predicted distribution is low, the predictions are more accurate. This is also evident in Fig. 3d, in which more confident predictions of the distance distribution (higher peak and lower s.d. of the distribution) tend to be more accurate, with the true distance close to the peak. Broader, less-confidently predicted distributions still assign probability to the correct value even when it is not close to the peak. The high accuracy of the distance predictions and consequently the contact predictions (Fig. 1c) comes from a combination of factors in the design of the neural network and its training, data augmentation, feature representation, auxiliary losses, cropping and data curation (see Methods).

To generate structures that conform to the distance predictions, we constructed a smooth potential V_{distance} by fitting a spline to the negative log probabilities, and summing across all of the residue pairs

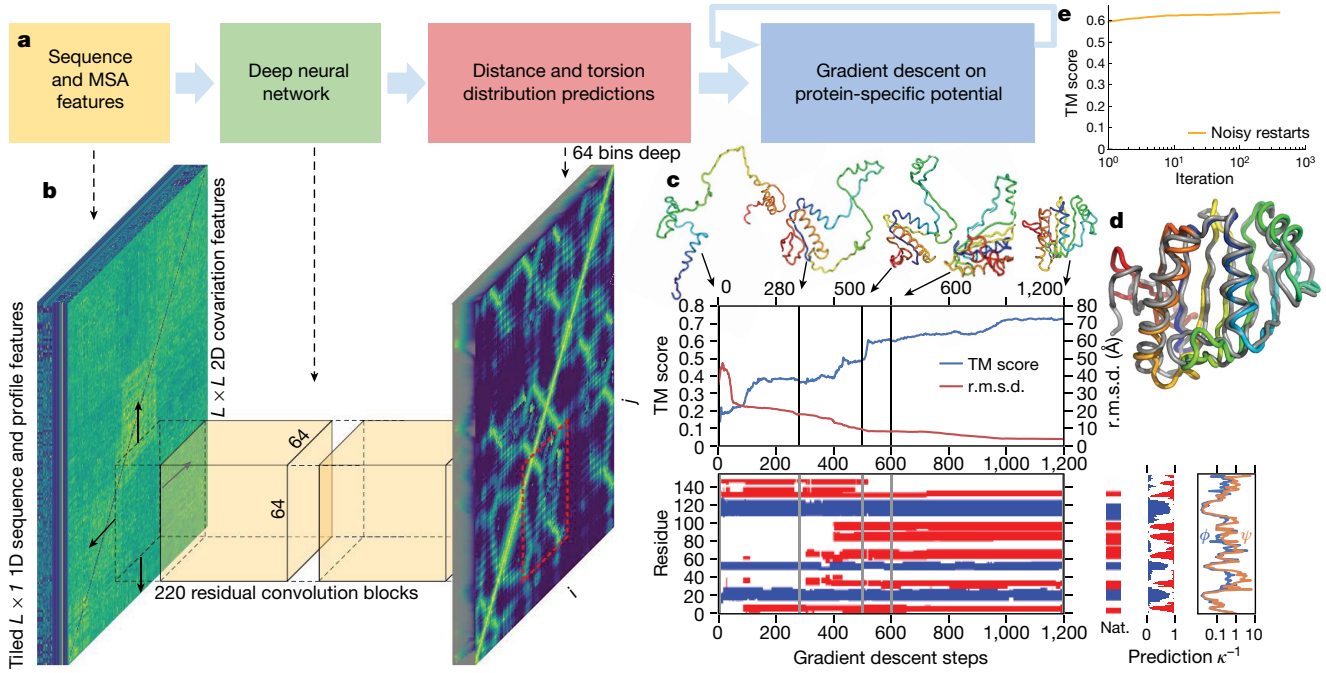


Fig. 2 | The folding process illustrated for CASP13 target T0986s2. CASP target T0986s2, $L = 155$, PDB: 6N9V. **a**, Steps of structure prediction. **b**, The neural network predicts the entire $L \times L$ distogram based on MSA features, accumulating separate predictions for 64 × 64-residue regions. **c**, One iteration of gradient descent (1,200 steps) is shown, with the TM score and root mean square deviation (r.m.s.d.) plotted against step number with five snapshots of the structure. The secondary structure (from SST[33]) is also shown (helix in blue, strand in red) along with the native secondary structure (Nat.), the secondary structure prediction probabilities of the network and the uncertainty in torsion angle predictions (as κ^{-1} of the von Mises distributions fitted to the predictions for φ and ψ). While each step of gradient descent greedily lowers the potential, large global conformation changes are effected, resulting in a well-packed chain. **d**, The final first submission overlaid on the native structure (in grey). **e**, The average (across the test set, $n = 377$) TM score of the lowest-potential structure against the number of repeats of gradient descent per target (log scale).

(see Methods). We parameterized protein structures by the backbone torsion angles (φ, ψ) of all residues and build a differentiable model of protein geometry $\mathbf{x} = G(\varphi, \psi)$ to compute the C_β coordinates, $\mathbf{x}_i$ for all residues i and thus the inter-residue distances, $d_{ij} = ||\mathbf{x}_i - \mathbf{x}_j||$, for each structure, and express V_{distance} as a function of φ and ψ. For a protein with L residues, this potential accumulates L^2 terms from marginal distribution predictions. To correct for the overrepresentation of the prior, we subtract a reference distribution[30] from the distance potential in the log domain. The reference distribution models the distance distributions $P(d_{ij}|\text{length})$ independent of the protein sequence and is computed by training a small version of the distance prediction neural network on the same structures, without sequence or MSA input features. A separate output head of the contact prediction network is trained to predict discrete probability distributions of backbone torsion angles $P(\varphi_i, \psi_i|S, \text{MSA}(S))$. After fitting a von Mises distribution, this is used to add a smooth torsion modelling term, V_{torsion}, to the potential. Finally, to prevent steric clashes, we add the $V_{\text{score2\_smooth}}$ score of Rosetta[9] to the potential, as this incorporates a van der Waals term. We used multiplicative weights for each of the three terms in the potential; however, no combination of weights noticeably outperformed equal weighting.

As all of the terms in the combined potential $V_{\text{total}}(\varphi, \psi)$ are differentiable functions of (φ, ψ), it can be optimized with respect to these variables by gradient descent. Here we use L-BFGS[31]. Structures are initialized by sampling torsion values from $P(\varphi_i, \psi_i|S, \text{MSA}(S))$. Figure 2c illustrates a single gradient descent trajectory that minimizes the potential, showing how this greedy optimization process leads to increasing accuracy and large-scale conformation changes. The secondary structure is partly set by the initialization from the predicted torsion angle distributions. The overall accuracy (TM score) improves quickly and after a few hundred steps of gradient descent the accuracy of the structure has converged to a local optimum of the potential.

We repeated the optimization from sampled initializations, leading to a pool of low-potential structures from which further structure initializations are sampled, with added backbone torsion noise ('noisy restarts'), leading to more structures to be added to the pool. After only a few hundred cycles, the optimization converges and the lowest potential structure is chosen as the best candidate structure. Figure 2e shows the progress in the accuracy of the best-scoring structures over multiple restarts of the gradient descent process, showing that after a few iterations the optimization has converged. Noisy restarts enable structures with a slightly higher TM score to be found than when continuing to sample from the predicted torsion distributions (average of 0.641 versus 0.636 on our test set, shown in Extended Data Fig. 4).

Figure 4a shows that the distogram accuracy (measured using the local distance difference test (lDDT_{12}) of the distogram; see Methods) correlates well with the TM score of the final realized structures. Figure 4b shows the effect of changing the construction of the potential. Removing the distance potential entirely gives a TM score of 0.266. Reducing the resolution of the distogram representation below six bins by averaging adjacent bins causes the TM score to degrade. Removing the torsion potential, reference correction or $V_{\text{score2\_smooth}}$ degrades the accuracy only slightly. A final 'relaxation' (side-chain packing interleaved with gradient descent) with Rosetta[9], using a combination of the Talaris2014 potential and a spline fit of our reference-corrected distance potential adds side-chain atom coordinates, and yields a small average improvement of 0.007 TM score.

We show that a carefully designed deep-learning system can provide accurate predictions of inter-residue distances and can be used to construct a protein-specific potential that represents the protein structure. Furthermore, we show that this potential can be optimized with gradient descent to achieve accurate structure predictions.

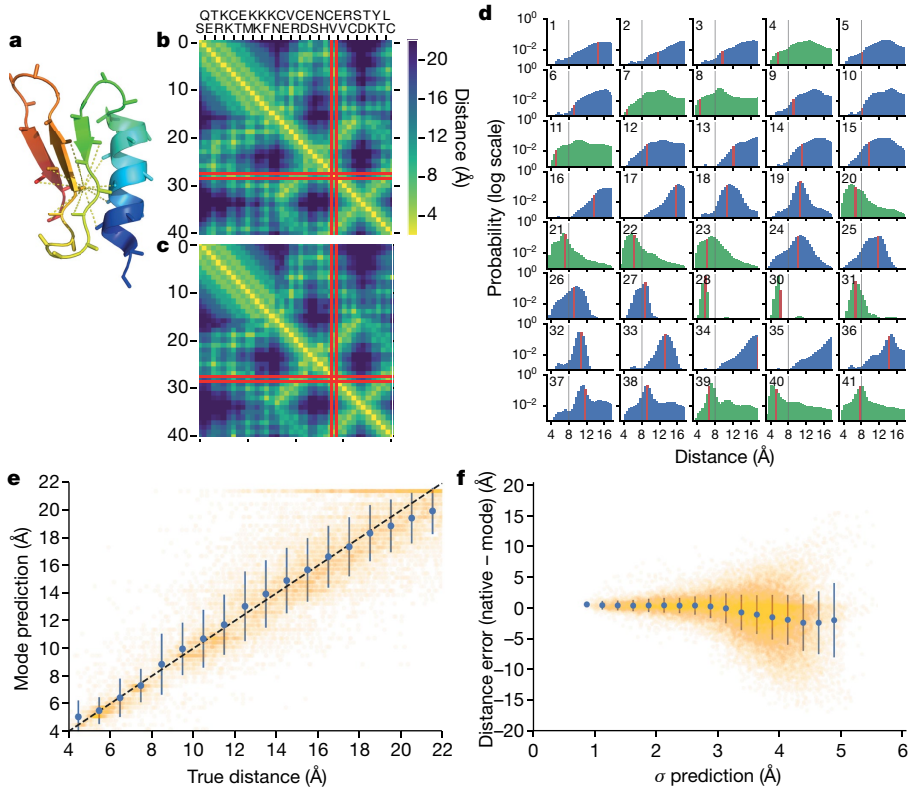


Fig. 3 | Predicted distance distributions compared with true distances. **a**–**d**, CASP target T0955, $L = 41$, PDB 5W9F. **a**, Native structure showing distances under 8 Å from the C_β of residue 29. **b**, **c**, Native inter-residue distances (**b**) and the mode of the distance predictions (**c**), highlighting residue 29. **d**, The predicted probability distributions for distances of residue 29 to all other residues. The bin corresponding to the native distance is highlighted in red, 8 Å is drawn in black. The distributions of the true contacts are plotted in green, non-contacts in blue. **e**, **f**, CASP target T0990, $L = 552$, PDB 6N9V. **e**, The mode of the predicted distance plotted against the true distance for all residue pairs with distances ≤22 Å, excluding distributions with s.d. > 3.5 Å ($n = 28{,}678$). Data are mean ± s.d. calculated for 1 Å bins. **f**, The error of the mode distance prediction versus the s.d. of the distance distributions, excluding pairs with native distances >22 Å ($n = 61{,}872$). Data are mean ± s.d. are shown for 0.25 Å bins. The true distance matrix and distogram for T0990 are shown in Extended Data Fig. 2b, c.

Whereas FM predictions only rarely approach the accuracy of experimental structures, the CASP13 assessment shows that the AlphaFold system achieves unprecedented FM accuracy and that this FM method can match the performance of template-modelling approaches without using templates and is starting to reach the accuracy needed to provide biological insights (see Methods). We hope that the methods we have

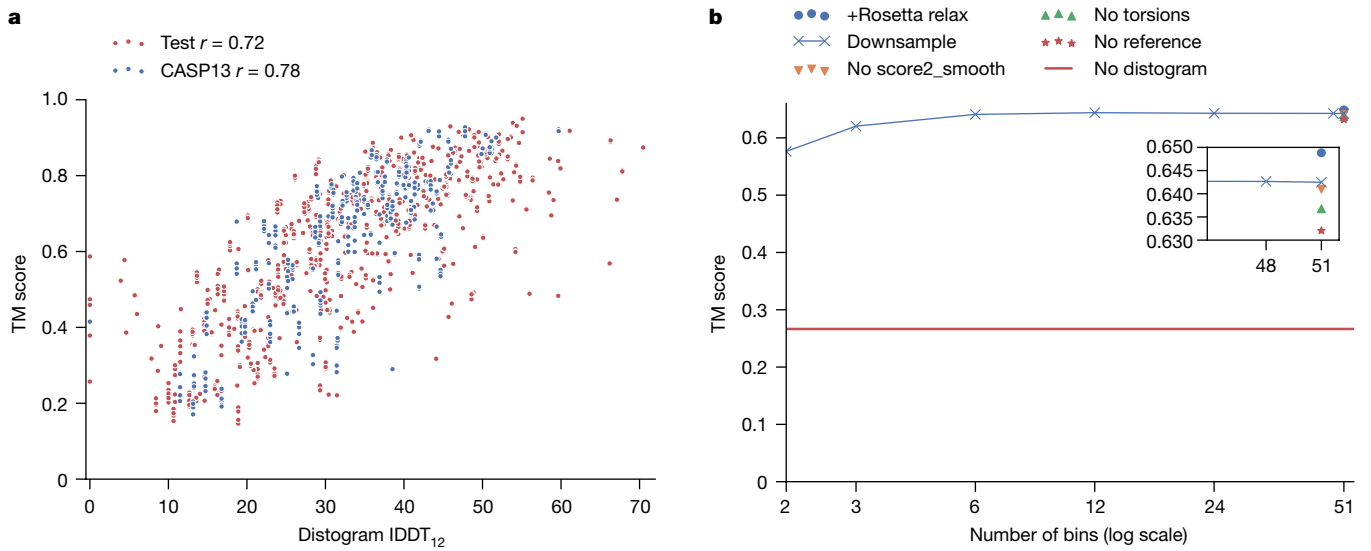


Fig. 4 | TM scores versus the accuracy of the distogram, and the dependency of the TM score on different components of the potential. **a**, TM score versus distogram $lDDT_{12}$ with Pearson's correlation coefficients, for both CASP13 ($n = 500$: 5 decoys for all domains, excluding T0999) and test ($n = 377$) datasets. **b**, Average TM score over the test set ($n = 377$) versus the number of histogram bins used when downsampling the distogram, compared with removing different components of the potential, or adding Rosetta relaxation.

described can be developed further and applied to benefit all areas of protein science with more accurate predictions for sequences of unknown structure.

Online content

Any methods, additional references, Nature Research reporting summaries, source data, extended data, supplementary information, acknowledgements, peer review information; details of author contributions and competing interests; and statements of data and code availability are available at https://doi.org/10.1038/s41586-019-1923-7.

1. Dill, K. A., Ozkan, S. B., Shell, M. S. & Weikl, T. R. The protein folding problem. *Annu. Rev. Biophys.* **37**, 289–316 (2008).
2. Dill, K. A. & MacCallum, J. L. The protein-folding problem, *50 years on*. *Science* **338**, 1042–1046 (2012).
3. Schaarschmidt, J., Monastyrskyy, B., Kryshtafovych, A. & Bonvin, A. M. J. J. Assessment of contact predictions in CASP12: co-evolution and deep learning coming of age. *Proteins* **86**, 51–66 (2018).
4. Kirkwood, J. Statistical mechanics of fluid mixtures. *J. Chem. Phys.* **3**, 300–313 (1935).
5. Kryshtafovych, A., Schwede, T., Topf, M., Fidelis, K. & Moult, J. Critical assessment of methods of protein structure prediction (CASP)—Round XIII. *Proteins* **87**, 1011–1020 (2019).
6. Zhang, Y. & Skolnick, J. Scoring function for automated assessment of protein structure template quality. *Proteins* **57**, 702–710 (2004).
7. Zhang, Y. Protein structure prediction: when is it useful? *Curr. Opin. Struct. Biol.* **19**, 145–155 (2009).
8. Senior, A. W. et al. Protein structure prediction using multiple deep neural networks in the 13th Critical Assessment of Protein Structure Prediction (CASP13). *Proteins* **87**, 1141–1148 (2019).
9. Das, R. & Baker, D. Macromolecular modeling with Rosetta. *Annu. Rev. Biochem.* **77**, 363–382 (2008).
10. Jones, D. T. Predicting novel protein folds by using FRAGFOLD. *Proteins* **45**, 127–132 (2001).
11. Zhang, C., Mortuza, S. M., He, B., Wang, Y. & Zhang, Y. Template-based and free modeling of I-TASSER and QUARK pipelines using predicted contact maps in CASP12. *Proteins* **86**, 136–151 (2018).
12. Kirkpatrick, S., Gelatt, C. D. Jr & Vecchi, M. P. Optimization by simulated annealing. *Science* **220**, 671–680 (1983).
13. Berman, H. M. et al. The Protein Data Bank. *Nucleic Acids Res.* **28**, 235–242 (2000).
14. Altschuh, D., Lesk, A. M., Bloomer, A. C. & Klug, A. Correlation of co-ordinated amino acid substitutions with function in viruses related to tobacco mosaic virus. *J. Mol. Biol.* **193**, 693–707 (1987).
15. Ovchinnikov, S., Kamisetty, H. & Baker, D. Robust and accurate prediction of residue–residue interactions across protein interfaces using evolutionary information. *eLife* **3**, e02030 (2014).
16. Seemayer, S., Gruber, M. & Söding, J. CCMpred—fast and precise prediction of protein residue–residue contacts from correlated mutations. *Bioinformatics* **30**, 3128–3130 (2014).
17. Morcos, F. et al. Direct-coupling analysis of residue coevolution captures native contacts across many protein families. *Proc. Natl Acad. Sci. USA* **108**, E1293–E1301 (2011).
18. Jones, D. T., Buchan, D. W., Cozzetto, D. & Pontil, M. PSICOV: precise structural contact prediction using sparse inverse covariance estimation on large multiple sequence alignments. *Bioinformatics* **28**, 184–190 (2012).
19. Skwark, M. J., Raimondi, D., Michel, M. & Elofsson, A. Improved contact predictions using the recognition of protein like contact patterns. *PLOS Comput. Biol.* **10**, e1003889 (2014).
20. Jones, D. T., Singh, T., Kosciolek, T. & Tetchner, S. MetaPSICOV: combining coevolution methods for accurate prediction of contacts and long range hydrogen bonding in proteins. *Bioinformatics* **31**, 999–1006 (2015).
21. Wang, S., Sun, S., Li, Z., Zhang, R. & Xu, J. Accurate de novo prediction of protein contact map by ultra-deep learning model. *PLOS Comput. Biol.* **13**, e1005324 (2017).
22. Jones, D. T. & Kandathil, S. M. High precision in protein contact prediction using fully convolutional neural networks and minimal sequence features. *Bioinformatics* **34**, 3308–3315 (2018).
23. Ovchinnikov, S. et al. Improved de novo structure prediction in CASP11 by incorporating coevolution information into Rosetta. *Proteins* **84**, 67–75 (2016).
24. Aszódi, A. & Taylor, W. R. Estimating polypeptide α-carbon distances from multiple sequence alignments. *J. Math. Chem.* **17**, 167–184 (1995).
25. Zhao, F. & Xu, J. A position-specific distance-dependent statistical potential for protein structure and functional study. *Structure* **20**, 1118–1126 (2012).
26. Xu, J. & Wang, S. Analysis of distance-based protein structure prediction by deep learning in CASP13. *Proteins* **87**, 1069–1081 (2019).
27. Aszódi, A., Gradwell, M. J. & Taylor, W. R. Global fold determination from a small number of distance restraints. *J. Mol. Biol.* **251**, 308–326 (1995).
28. Kandathil, S. M., Greener, J. G. & Jones, D. T. Prediction of interresidue contacts with DeepMetaPSICOV in CASP13. *Proteins* **87**, 1092–1099 (2019).
29. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition* 770–778 (2016).
30. Simons, K. T., Kooperberg, C., Huang, E. & Baker, D. Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and Bayesian scoring functions. *J. Mol. Biol.* **268**, 209–225 (1997).
31. Liu, D. C. & Nocedal, J. On the limited memory BFGS method for large scale optimization. *Math. Program.* **45**, 503–528 (1989).
32. Li, Y., Zhang, C., Bell, E. W., Yu, D.-J. & Zhang, Y. Ensembling multiple raw coevolutionary features with deep residual neural networks for contact-map prediction in CASP13. *Proteins* **87**, 1082–1091 (2019).
33. Konagurthu, A. S., Lesk, A. M. & Allison, L. Minimum message length inference of secondary structure from protein coordinate data. *Bioinformatics* **28**, i97–i105 (2012).

Methods

Extended Data Figure 1a shows the steps involved in MSA construction, feature extraction, distance prediction, potential construction and structure realization.

Tools

The following tools and dataset versions were used for the CASP system and for subsequent experiments: PDB 15 March 2018; CATH 16 March 2018; HHblits based on v.3.0-beta.3 (three iterations, $E=1\times10^{-3}$); HHpred web server; Uniclust30 2017-10; PSI-BLAST v.2.6.0 nr dataset (as of 15 December 2017) (three iterations, $E=1\times10^{-3}$); SST web server (March 2019); BioPython v.1.65; Rosetta v.3.5; PyMol 2.2.0 for structure visualization; TM-align 20160521.

Data

Our models are trained on structures extracted from the PDB[13]. We extract non-redundant domains by utilizing the CATH[34] 35% sequence similarity cluster representatives. This generated 31,247 domains, which were split into train and test sets (29,427 and 1,820 proteins, respectively), keeping all domains from the same homologous superfamily (H-level in the CATH classification) in the same partition. The CATH superfamilies of FM domains from CASP11 and CASP12 were also excluded from the training set. From the test set, we took—at random—a single domain per homologous superfamily to create the 377 domain subset used for the results presented here. We note that accuracies for this set are higher than for the CASP13 test domains.

CASP13 submission results are drawn from the CASP13 results pages with additional results shown for the CASP13 dataset for 'all groups' chains, scored on CASP13 PDB files, by CASP domain definitions. Contact prediction accuracies were recomputed from the group 032 and 498 submissions (as RR files), compared with the distogram predictions used by AlphaFold for CASP13 submissions. Contact prediction probabilities were obtained from the distograms by summing the probability mass in each distribution below 8 Å.

For each training sequence, we searched for and aligned to the training sequence similar protein sequences in the Uniclust30[35] dataset with HHblits[36] and used the returned MSA to generate profile features with the position-specific substitution probabilities for each residue as well as covariation features—the parameters of a regularized pseudolikelihood-trained Potts model similar to CCMpred[16]. CCMPred uses the Frobenius norm of the parameters, but we feed both this norm (1 feature) and the raw parameters (484 features) into the network for each residue pair *ij*. In addition, we provide the network with features that explicitly represent gaps and deletions in the MSA. To make the network better able to make predictions for shallow MSAs, and as a form of data augmentation, we take a sample of half the sequences from the the HHblits MSA before computing the MSA-based features. Our training set contains 10 such samples for each domain. We extract additional profile features using PSI-BLAST[37].

The distance prediction neural network was trained with the following input features (with the number of features indicated in brackets).

- Number of HHblits alignments (scalar).
- Sequence-length features: 1-hot amino acid type (21 features); profiles: PSI-BLAST (21 features), HHblits profile (22 features), non-gapped profile (21 features), HHblits bias, HMM profile (30 features), Potts model bias (22 features); deletion probability (1 feature); residue index (integer index of residue number, consecutive except for multi-segment domains, encoded as 5 least-significant bits and a scalar).
- Sequence-length-squared features: Potts model parameters (484 features, fitted with 500 iterations of gradient descent using Nesterov momentum 0.99, without sequence reweighting); Frobenius norm (1 feature); gap matrix (1 feature).

The *z*-scores were taken from the results CASP13 assessors (http://predictioncenter.org/casp13/zscores_final.cgi?formula=assessors).

Distogram prediction. The inter-residue distances are predicted by a deep neural network. The architecture is a deep two-dimensional dilated convolutional residual network. Previously, a two-dimensional residual network was used that was preceded by one-dimensional embedding layers for contact prediction[21]. Our network is two-dimensional throughout and uses 220 residual blocks[29] with dilated convolutions[38]. Each residual block, illustrated in Extended Data Fig. 1b, consists of a sequence of neural network layers[39] that interleave three batchnorm layers; two 1×1 projection layers; a 3×3 dilated convolution layer and exponential linear unit (ELU)[40] nonlinearities. Successive layers cycle through dilations of 1, 2, 4, 8 pixels to allow propagation of information quickly across the cropped region. For the final layer, a position-specific bias was used, such that the biases were indexed by residue-offset (capped at 32) and bin number.

The network is trained with stochastic gradient descent using a cross-entropy loss. The target is a quantification of the distance between the C_β atoms of the residues (or C_α for glycine). We divide the range 2–22 Å into 64 equal bins. The input to the network consists of a two-dimensional array of features in which each *i*,*j* feature is the concatenation of the one-dimensional features for both *i* and *j* as well as the two-dimensional features for *i*,*j*.

Individual training runs were cross-validated with early stopping using 27 CASP11 FM domains as a validation set. Models were selected by cross-validation on 27 CASP12 FM domains.

Neural network hyperparameters

- 7 groups of 4 blocks with 256 channels, cycling through dilations 1, 2, 4, 8.
- 48 groups of 4 blocks with 128 channels, cycling through dilations 1, 2, 4, 8.
- Optimization: synchronized stochastic gradient descent
- Batch size: batch of 4 crops on each of 8 GPU workers.
- 0.85 dropout keep probability.
- Nonlinearity: ELU.
- Learning rate: 0.06.
- Auxiliary loss weights: secondary structure: 0.005; accessible surface area: 0.001. These auxiliary losses were cut by a factor 10 after 100 000 steps.
- Learning rate decayed by 50% at 150,000, 200,000, 250,000 and 350,000 steps.
- Training time: about 5 days for 600,000 steps.

Cropped distograms. To constrain memory usage and avoid overfitting, the network was always trained and tested on 64×64 regions of the distance matrix, that is, the pairwise distances between 64 consecutive residues and another group of 64 consecutive residues. For each training domain, the entire distance matrix was split into non-overlapping 64×64 crops. By training off-diagonal crops, the interaction between residues that are further apart than 64 residues could be modelled. Each crop consisted of the distance matrix that represented the juxtaposition of two 64-residue fragments. It has previously been shown[22] that contact prediction needs only a limited context window. We note that the distance predictions close to the diagonal $i=j$, encode predictions of the local structure of the protein, and for any cropped region the distances are governed by the local structure of the two fragments represented by the *i* and *j* ranges of the crop. Augmenting the inputs with the on-diagonal two-dimensional input features that correspond to both the *i* and *j* ranges provides additional information to predict the structure of each fragment and thus the distances between them. It can be seen that if the fragment structures can be well predicted (for instance, if they are confidently predicted as helices or sheets), then the prediction of a single contact

between the fragments will strongly constrain the distances between all other pairs.

Randomizing the offset of the crops each time a domain is used in training leads to a form of data augmentation in which a single protein can generate many thousands of different training examples. This is further enhanced by adding noise proportional to the ground-truth resolution to the atom coordinates, leading to variation in the target distances. Data augmentation (MSA subsampling and coordinate noise), together with dropout[41], prevents the network from overfitting to the training data.

To predict the distance distribution for all $L \times L$ residue pairs, many 64×64 crops are combined. To avoid edge effects, several such tilings are produced with different offsets and averaged together, with a heavier weighting for the predictions near the centre of the crop. To improve accuracy further, predictions from an ensemble of four separate models, trained independently with slightly different hyperparameters, are averaged together. Extended Data Figure 2b, c shows examples of the true distances and the mode of the distogram predictions for a three-domain CASP13 target, T0990.

As the network has a rich representation capable of incorporating both profile and covariation features of the MSA, we argue that the network can be used to predict the secondary structure directly. By mean- and max- pooling the two-dimensional activations of the penultimate layer of the network separately in both i and j, we add an additional one-dimensional output head to the network that predicts eight-class secondary structure labels as computed by DSSP[42] for each residue in j and i. The resulting accuracy of the Q3 (distinguishing the three helix/sheet/coil classes) predictions is 84%, which is comparable to the state-of-the-art predictions[43]. The relative accessible surface area (ASA) of each residue can also be predicted.

The one-dimensional pooled activations are also used to predict the marginal Ramachandran distributions, $P(\varphi_i, \psi_i|S,\mathrm{MSA}(S))$, independently for each residue, as a discrete probability distribution approximated to 10° (1,296 bins). In practice during CASP13 we used distograms from a network that was trained to predict distograms, secondary structure and ASA. Torsion predictions were taken from a second similar network trained to predict distograms, secondary structure, ASA and torsions, as the former had been more thoroughly validated.

Extended Data Figure 3b shows that an important factor in the accuracy of the distograms (as has previously been found with contact prediction systems) is N_{eff}, the effective number of sequences in the MSA[20]. This is the number of sequences found in the MSA, discounting redundancy at the 62% sequence identity level, which we then divide by the number of residues in the target, and is an indication of the amount of covariation information in the MSA.

Distance potential. The distogram probabilities are estimated for discrete distance bins; therefore, to construct a differentiable potential, the distribution is interpolated with a cubic spline. Because the final bin accumulates probability mass from all distances beyond 22 Å, and as greater distances are harder to predict accurately, the potential was only fitted up to 18 Å (determined by cross-validation), with a constant extrapolation thereafter. Extended Data Figure 3c (bottom) shows the effect of varying the resolution of the distance histograms on structure accuracy.

To predict a reference distribution, a similar model is trained on the same dataset. The reference distribution is not conditioned on the sequence, but to account for the atoms between which we are predicting distances, we do provide a binary feature $\delta_{\alpha\beta}$ to indicate whether the residue is a glycine (C_α atom) or not (C_β) and the overall length of the protein.

A distance potential is created from the negative log likelihood of the distances, summed over all pairs of residues i, j (Supplementary equation (1)). With a reference state, this becomes the log-likelihood ratio of the distances under the full conditional model and under the background model (Supplementary equation (2)).

Torsions are modelled as a negative log likelihood under the predicted torsion distributions. As we have marginal distribution predictions, each of which can be multimodal, it can be difficult to jointly optimize the torsions. To unify all of the probability mass, at the cost of modelling fidelity of multimodal distributions, we fitted a unimodal von Mises distribution to the marginal predictions. This potential was summed over all residues i (Supplementary equation (3)).

Finally, to prevent steric clashes, a van der Waals term was introduced through the use of Rosetta's $V_{\mathrm{score2\_smooth}}$. Extended Data Figure 3c (top) shows the effect on the accuracy of the structure prediction of different terms in the potential.

Structure realization by gradient descent. To realize structures that minimize the constructed potential, we created a differentiable model of ideal protein backbone geometry, giving backbone atom coordinates as a function of the torsion angles (φ, ψ): $\mathbf{x} = G(\varphi, \psi)$. The complete potential to be minimized is then the sum of the distance, torsion and score2_smooth (Supplementary equation (4)). Although there is no guarantee that these potentials have equivalent scale, scaling parameters on the terms were introduced and chosen by cross-validation on CASP12 FM domains. In practice, equal weighting for all terms was found to lead to the best results.

As every term in V_{total} is differentiable with respect to the torsion angles, given an initial set of torsions φ, ψ, which can be sampled from the predicted torsion marginals, we can minimize V_{total} using a gradient descent algorithm, such as L-BFGS[31]. The optimized structure is dependent on the initial conditions, so we repeat the optimization multiple times with different initializations. A pool of the 20 lowest-potential structures is maintained and once full, we initialize 90% of trajectories from those with 30° noise added to the backbone torsions (the remaining 10% still being sampled from the predicted torsion distributions). In CASP13, we obtained 5,000 optimization runs for each chain. Figure 2c shows the change in TM score against the number of restarts per protein. As longer chains take longer to optimize, this work load was balanced across $(50 + L)/2$ parallel workers. Extended Data Figure 4 shows similar curves against computation time, always comparing sampling starting torsions from the predicted marginal distributions with restarting from the pool of previous structures.

Accuracy. We compare the final structures to the experimentally determined structures to measure their accuracy using metrics such as TM score, GDT_TS (global distance test, total score[44]) and r.m.s.d. All of these accuracy measures require geometric alignment between the candidate structure and the experimental structure. An alternative accuracy measure that requires no alignment is the lDDT[45], which measures the percentage of native pairwise distances D_{ij} under 15 Å, with sequence offsets $\geq r$ residues, that are realized in a candidate structure (as d_{ij}) within a tolerance of the true value, averaging across tolerances of 0.5, 1, 2 and 4 Å (without stereochemical checks), as shown in Supplementary equation (5)).

As the distogram predicts pairwise distances, we can introduce distogram lDDT (DLDDT), a measure similar to lDDT that is computed directly from the probabilities of the distograms, as shown in Supplementary equation (6)). As distances between residues nearby in the sequence are often short, easier to predict and are not critical in determining the overall fold topology, we set $r = 12$, considering only those distances for residues with a sequence separation ≥12. Because we predict C_β distances, for this study we computed both lDDT and DLDDT using the C_β distances. Extended Data Figure 3a shows that DLDDT_{12} has high correlation (Pearson's $r = 0.92$ for CASP13) with the lDDT_{12} of the realized structures.

Full chains without domain segmentation. Parameterizing proteins of length L by two torsion angles per residue, the dimension of the space of structures grows as $2L$; thus, searching for structures of large proteins becomes much more difficult. Traditionally this problem was addressed by splitting longer protein chains into pieces—termed domains—that fold independently. However, domain segmentation from the sequence alone is itself difficult and error-prone. For this study, we avoided domain segmentation and folded entire chains. Typically, MSAs are based on a given domain segmentation; however, we used a sliding window approach, computing a full-chain MSA to predict a baseline full-sequence distogram. We then computed MSAs for subsequences of the chain, trying windows of size 64, 128, 256 with offsets at multiples of 64. Each of these MSAs gave rise to an individual distogram that corresponded to an on-diagonal square of the full-chain distogram. We averaged all of these distograms together, weighted by the number of sequences in the MSA to produce an average full-chain distogram that is more accurate in regions in which many alignments can be found. For the CASP13 assessment, full chains were relaxed with Rosetta relax with a potential of $V_{\text{Talaris2014}} + 0.2\, V_{\text{distance}}$ (weighting determined by cross-validation) and submissions from all of the systems were ranked based on this potential.

CASP13 results. For CASP13, the five AlphaFold submissions were from three different systems, all of which used potentials based on the neural network distance predictions. The systems that are not described here are described in a separate paper[8]. Before T0975, two systems based on simulated annealing and fragment assembly (and using 40-bin distance distributions) were used. From T0975 onward, newly trained 64-bin distogram predictions were used and structures were generated by the gradient descent system described here (three independent runs) as well as one of the fragment assembly systems (five independent runs). The five submissions were chosen from these eight structures (the lowest potential structure generated by each independent run) with the first submission (top-one) being the lowest-potential structure generated by gradient descent. The remaining four submissions were the four best other structures, with the fifth being a gradient descent structure if none had been chosen for position 2, 3 or 4. All submissions for T0999 were generated by gradient descent. Extended Data Figure 5a shows the methods used for each submission, comparing with 'back-fill' structures generated by a single run of gradient descent for targets before T0975. Extended Data Figure 5b shows that the gradient descent method that was used later in CASP performed better than the fragment assembly method, in each category. Extended Data Figure 5c compares the accuracy of the AlphaFold submissions for FM and FM/TBM domains with the next best group 322. The assessors of CASP13 FM used expert visual inspection[46] to choose the best submissions for each target and found that AlphaFold had nearly twice as many best models as the next best group.

Biological relevance of AlphaFold predictions. There is a wide range of uses of predicted structures, all with different accuracy requirements, from generally understanding the fold shape to understanding detailed side-chain configurations in binding regions. Contact predictions alone can guide biological insights[47], for instance, to target mutations to destabilize the protein. Figure 1c and Extended Data Fig. 2a show that the accuracy of the contact predictions from AlphaFold exceeds that of the state-of-the-art predictions. In Extended Data Figs. 6–8, we present further results that show that the accuracy improvements of AlphaFold lead to more accurate interpretations of function (Extended Data Fig. 6); better interface prediction for protein–protein interactions (Extended Data Fig. 7); better binding pocket prediction (Extended Data Fig. 8) and improved molecular replacement in crystallography.

Thus far only template-based predictions have been able to deliver the most accurate predictions. Although AlphaFold is able to match TBM without using templates, and in some cases outperform other methods (for example, T0981-D5, 72.8 GDT_TS, and T0957s1-D2, 88.0 GDT_TS, two TBM-hard domains for which the top-one model of AlphaFold is 12 GDT_TS better than any other top-one submission), the accuracy for FM targets still lags behind that for TBM targets and can still not be relied on for the detailed understanding of hard structures. In an analysis of the performance of CASP13 TBM predictions for molecular replacement, another study[48] reported that the AlphaFold predictions (raw coordinates, without B-factors) led to a marginally greater log-likelihood gain than those of any other group, indicating that these improved structures can assist in phasing for X-ray crystallography.

Interpretation of distogram neural network. We have shown that the deep distance prediction neural network achieves high accuracy, but we would like to understand how the network arrives at its distance predictions and—in particular—to understand how the inputs to the model affect the final prediction. This might improve our understanding of the folding mechanisms or suggest improvements to the model. However, deep neural networks are complex nonlinear functions of their inputs, and so this attribution problem is difficult, under-specified and an on-going topic of research. Even so, there are a number of methods for such analysis: here we apply Integrated Gradients[49] to our trained distogram network to indicate the location of input features that affect the network's predictions of a particular distance.

In Extended Data Fig. 9, plots of summed absolute Integrated Gradient, $\sum_c |S^{I,J}_{i,j,c}|$, (defined in Supplementary equations (7)–(9)) are shown for selected I,J output pairs in T0986s2; and in Extended Data Fig. 10, the top-10 highest attribution input pairs for each output pair are shown on top of the top-one predicted structure of AlphaFold. The attribution maps are sparse and highly structured, closely reflecting the predicted geometry of the protein. For the four in-contact pairs presented (1, 2, 3, 5), all of the highest attribution pairs are pairs within or between the secondary structure that one or both of the output pair(s) are members of. In 1, the helix residues are important as well as connections between the strands that follow either end of the helix, which might indicate strain on the helix. In 2, all of the most important residue pairs connect the same two strands, whereas in 3, a mixture of inter-strand pairs and strand residues is most salient. In 5, the most important pairs involve the packing of nearby secondary structure elements to the strand and helix. For the non-contacting pair, 4, the most important input pairs are the residues that are geometrically between I and J in the predicted protein structure. Furthermore, most of the high-attribution input pairs are themselves in contact.

As the network is tasked with predicting the spatial geometry, with no structure available at the input, these patterns of interaction indicate that the network is using intermediate predictions to discover important interactions and channelling information from related residues to refine the final prediction.

Reporting summary

Further information on research design is available in the Nature Research Reporting Summary linked to this paper.

Data availability

Our training, validation and test data splits (CATH domain codes) are available from https://github.com/deepmind/deepmind-research/tree/master/alphafold_casp13. The following versions of public datasets were used in this study: PDB 2018-03-15; CATH 2018-03-16; Uniclust30 2017-10; and PSI-BLAST nr dataset (as of 15 December 2017).

Code availability

Source code for the distogram, reference distogram and torsion prediction neural networks, together with the neural network weights and input data for the CASP13 targets are available for research and non-commercial use at https://github.com/deepmind/deepmind-research/tree/master/alphafold_casp13. We make use of several open-source libraries to conduct our experiments, particularly HHblits[36], PSI-BLAST[37] and the machine-learning framework TensorFlow (https://github.com/tensorflow/tensorflow) along with the TensorFlow library Sonnet (https://github.com/deepmind/sonnet), which provides implementations of individual model components[50]. We also used Rosetta[9] under license.

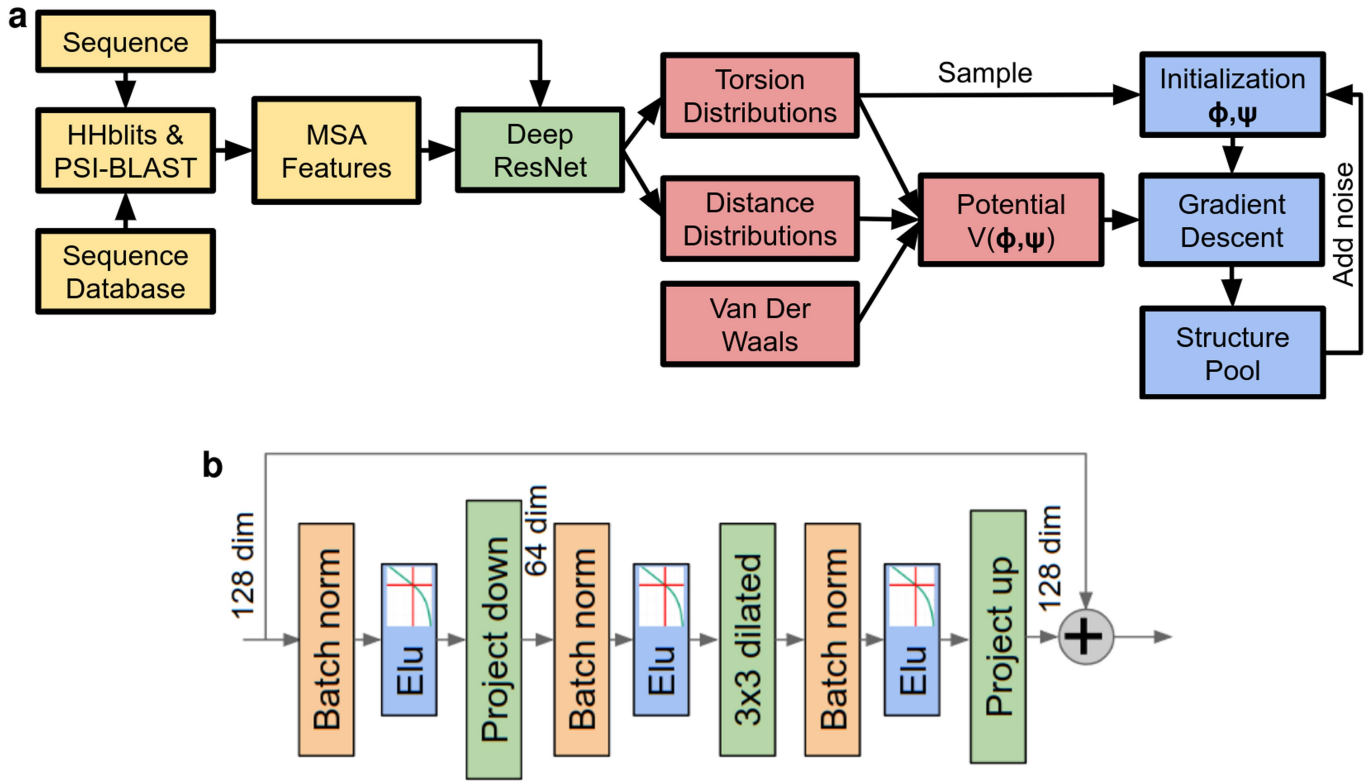
34. Dawson, N. L. et al. CATH: an expanded resource to predict protein function through structure and sequence. *Nucleic Acids Res.* **45**, D289–D295 (2017).
35. Mirdita, M. et al. Uniclust databases of clustered and deeply annotated protein sequences and alignments. *Nucleic Acids Res.* **45**, D170–D176 (2017).
36. Remmert, M., Biegert, A., Hauser, A. & Söding, J. HHblits: lightning-fast iterative protein sequence searching by HMM–HMM alignment. *Nat. Methods* **9**, 173–175 (2012).
37. Altschul, S. F. et al. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.* **25**, 3389–3402 (1997).
38. Yu, F. & Koltun, V. Multi-scale context aggregation by dilated convolutions. Preprint at *arXiv* https://arxiv.org/abs/1511.07122 (2015).
39. Oord, A. d. et al. Wavenet: a generative model for raw audio. Preprint at *arXiv* https://arxiv.org/abs/1609.03499 (2016).
40. Clevert, D.-A., Unterthiner, T. & Hochreiter, S. Fast and accurate deep network learning by exponential linear units (ELUs). Preprint at *arXiv* https://arxiv.org/abs/1511.07289 (2015).
41. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **15**, 1929–1958 (2014).
42. Kabsch, W. & Sander, C. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers* **22**, 2577–2637 (1983).
43. Yang, Y. et al. Sixty-five years of the long march in protein secondary structure prediction: the final stretch? *Briefings Bioinf.* **19**, 482–494 (2018).
44. Zemla, A., Venclovas, C., Moult, J. & Fidelis, K. Processing and analysis of CASP3 protein structure predictions. *Proteins* **37**, 22–29 (1999).
45. Mariani, V., Biasini, M., Barbato, A. & Schwede, T. lDDT: a local superposition-free score for comparing protein structures and models using distance difference tests. *Bioinformatics* **29**, 2722–2728 (2013).
46. Abriata, L. A., Tamo, G. E. & Dal Peraro, M. A further leap of improvement in tertiary structure prediction in CASP13 prompts new routes for future assessments. *Proteins* **87**, 1100–1112 (2019).
47. Kayikci, M. et al. Visualization and analysis of non-covalent contacts using the Protein Contacts Atlas. *Nat. Struct. Mol. Biol.* **25**, 185–194 (2018).
48. Croll, T. I. et al. Evaluation of template-based modeling in CASP13. *Proteins* **87**, 1113–1127 (2019).
49. Sundararajan, M., Taly, A. & Yan, Q. Axiomatic attribution for deep networks. In *Proc. 34th International Conference on Machine Learning* **Vol. 70**, 3319–3328 (2017).
50. Abadi, M. et al. Tensorflow: a system for large-scale machine learning. In *Proc. 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)* 265–283 (2016).
51. Söding, J., Biegert, A. & Lupas, A. N. The HHpred interactive server for protein homology detection and structure prediction. *Nucleic Acids Res.* **33**, W244–W248 (2005).
52. Cong, Q. et al. An automatic method for CASP9 free modeling structure prediction assessment. *Bioinformatics* **27**, 3371–3378 (2011).
53. Zhang, Y. & Skolnick, J. TM-align: a protein structure alignment algorithm based on the TM-score. *Nucleic Acids Res.* **33**, 2302–2309 (2005).
54. Tovchigrechko, A., Wells, C. A. & Vakser, I. A. Docking of protein models. *Protein Sci.* **11**, 1888–1896 (2002).
55. Audet, M. et al. Crystal structure of misoprostol bound to the labor inducer prostaglandin E_2 receptor. *Nat. Chem. Biol.* **15**, 11–17 (2019).

Acknowledgements We thank C. Meyer for assistance in preparing the paper; B. Coppin, O. Vinyals, M. Barwinski, R. Sun, C. Elkin, P. Dolan, M. Lai and Y. Li for their contributions and support; O. Ronneberger for reading the paper; the rest of the DeepMind team for their support; the CASP13 organisers and the experimentalists whose structures enabled the assessment.

Author contributions R.E., J.J., J.K., L.S., A.W.S., C.Q., T.G., A.Ž., A.B., H.P. and K.S. designed and built the AlphaFold system with advice from D.S., K.K. and D.H. D.T.J. provided advice and guidance on protein structure prediction methodology. S.P. contributed to software engineering. S.C., A.W.R.N., K.K. and D.H. managed the project. J.K., A.W.S., T.G., A.Ž., A.B., R.E., P.K. and J.J. analysed the CASP results for the paper. A.W.S. and J.K. wrote the paper with contributions from J.J., R.E., L.S., T.G., A.B., A.Ž., D.T.J., P.K., K.K. and D.H. A.W.S. led the team.

Competing interests A.W.S., J.K., T.G., J.J., L.S., R.E., H.P., C.Q., K.S., A.Ž. and A.B. have filed provisional patent applications relating to machine learning for predicting protein structures. The remaining authors declare no competing interests.

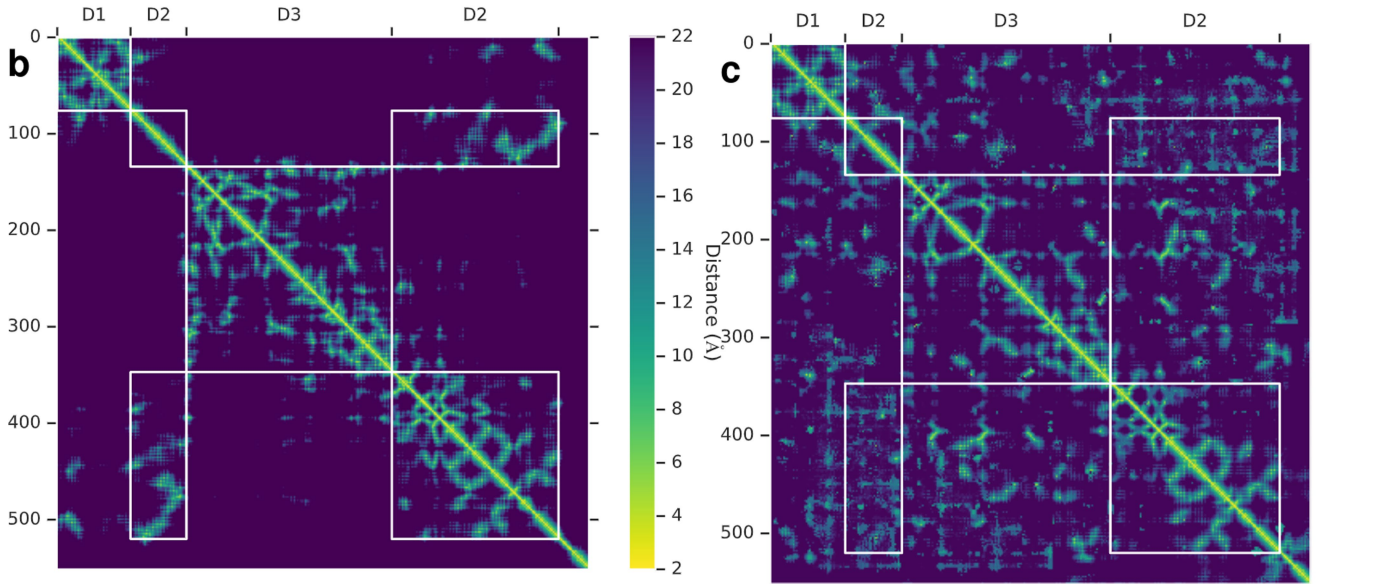
Additional information
Supplementary information is available for this paper at https://doi.org/10.1038/s41586-019-1923-7.
Correspondence and requests for materials should be addressed to A.W.S.
Peer review information *Nature* thanks Mohammed AlQuraishi and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.



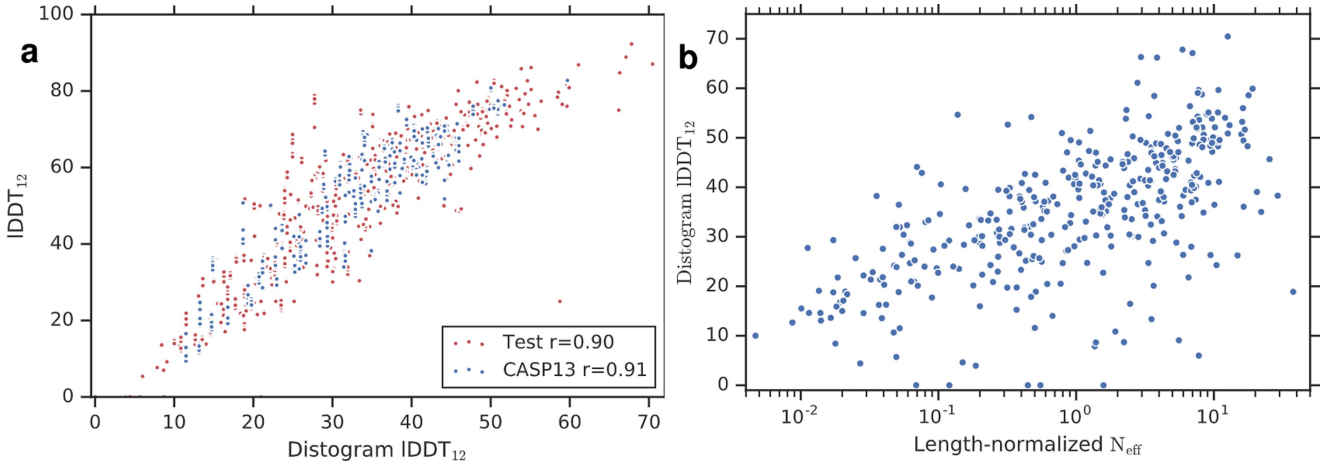
Extended Data Fig. 1 | Schematics of the folding system and neural network. **a**, The overall folding system. Feature extraction stages (constructing the MSA using sequence database search and computing MSA-based features) are shown in yellow; the structure-prediction neural network in green; potential construction in red; and structure realization in blue. **b**, The layers used in one block of the deep residual convolutional network. The dilated convolution is applied to activations of reduced dimension. The output of the block is added to the representation from the previous layer. The bypass connections of the residual network enable gradients to pass back through the network undiminished, permitting the training of very deep networks.

a

Contact precisions		L long			L/2 long			L/5 long		
Set	N	AF	498	032	AF	498	032	AF	498	032
FM	31	**46.1**	43.1	40.1	**58.5**	54.9	51.6	**69.9**	67.3	61.9
FM/TBM	12	**59.1**	53.0	48.9	**74.2**	64.5	64.2	**85.3**	81.0	79.6
TBM	61	**68.3**	65.5	61.9	**82.4**	80.3	76.4	**90.6**	90.5	87.1



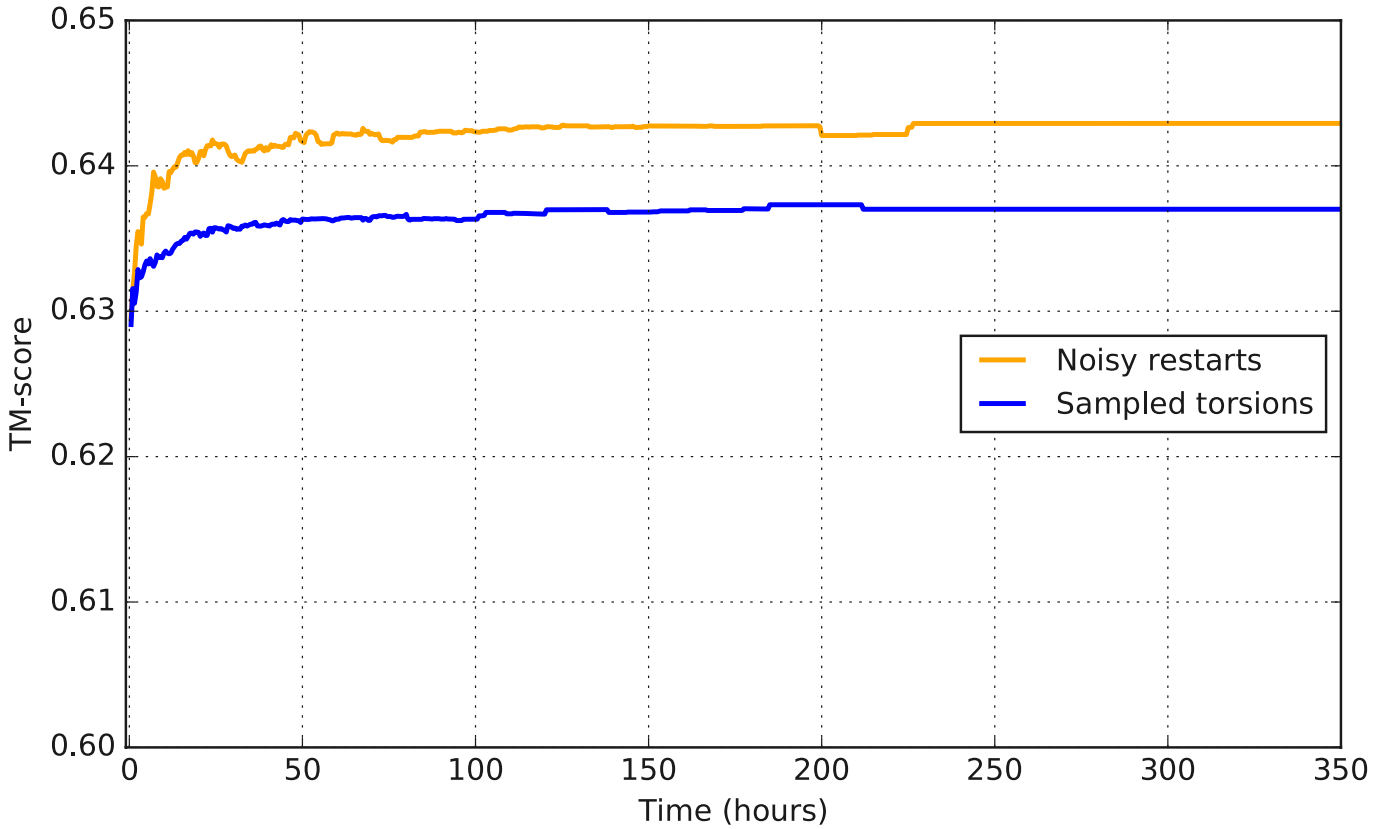
Extended Data Fig. 2 | CASP13 contact precisions. a, Precisions (as shown in Fig. 1c) for long-range contact prediction in CASP13 for the most probable L, $L/2$ or $L/5$ contacts, where L is the length of the domain. The distance distributions used by AlphaFold (AF) in CASP13, thresholded to contact predictions, are compared with submissions by the two best-ranked contact prediction methods in CASP13: 498 (RaptorX-Contact[26]) and 032 (TripletRes[32]), on 'all groups' targets, with updated domain definitions for T0953s2. **b**, **c**, True distances (**b**) and modes of the predicted distogram (**c**) for CASP13 target T0990. CASP divides this chain into three domains as shown (D3 is inserted in D2) for which there are 39, 36 and 42 HHblits alignments, respectively (from the CASP website).



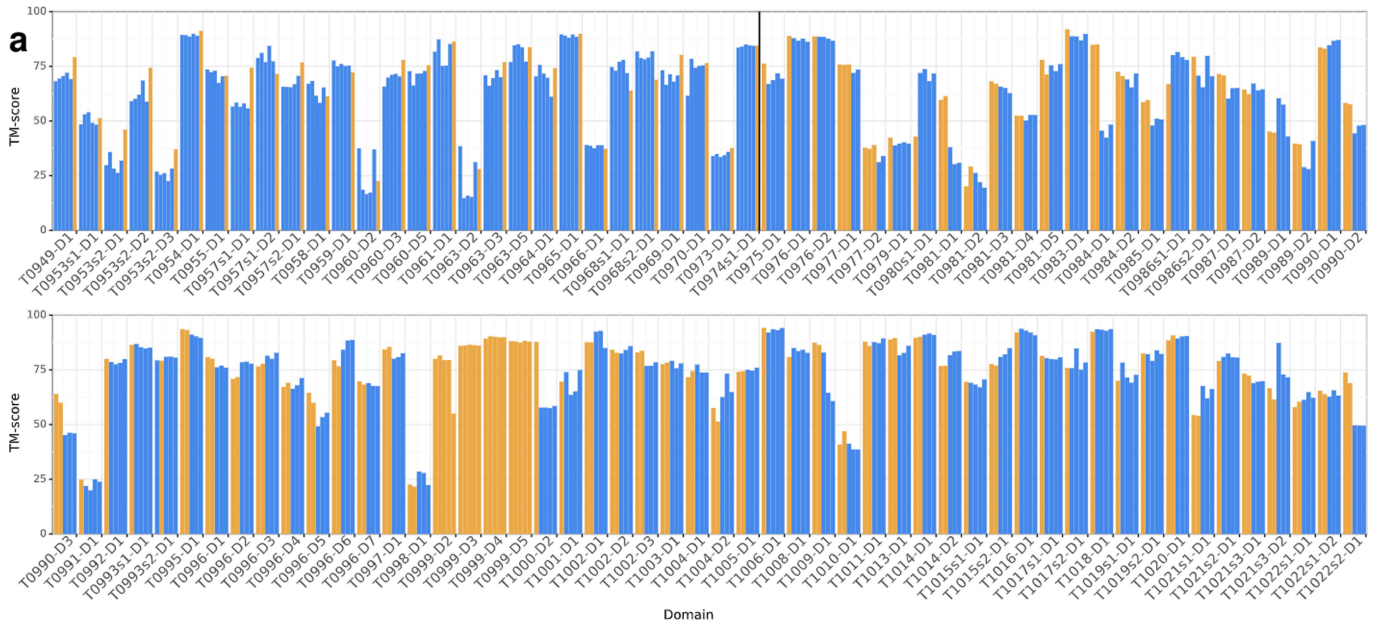
c

Potential	Bins	TM-score	GDT_TS	lDDT	RMSD (Å)	$-\log_{10} P$
Full + relax	51/64	0.649	65.8	54.2	5.94	7.3
Full	51/64	0.642	65.0	53.9	5.91	–
W/o reference	51/64	0.632	64.3	50.0	6.64	4.0
W/o `score2_smooth`	51/64	0.641	64.8	53.7	5.93	1.2
W/o torsions	51/64	0.637	64.3	53.6	6.04	8.2
W/o distogram	51/64	0.266	29.1	19.1	14.88	130
Full	48/64	0.643	65.0	54.1	5.90	
Full	24/32	0.643	65.0	53.8	5.89	
Full	12/16	0.644	65.1	53.9	5.85	
Full	6/8	0.641	64.6	53.7	5.94	
Full	3/4	0.620	62.4	52.8	6.22	
Full	2/3	0.576	58.2	49.3	8.38	

Extended Data Fig. 3 | Analysis of structure accuracies. a, $lDDT_{12}$ versus distogram $lDDT_{12}$ (see Methods, 'Accuracy'). The distogram accuracy predicts the lDDT of the realized structure well (particularly for medium- and long-range residue pairs, as well as the TM score as shown in Fig. 4a) for both CASP13 ($n = 500$: 5 decoys for domains excluding T0999) and test ($n = 377$) datasets. Data are shown with Pearson's correlation coefficients. **b**, $DLDDT_{12}$ against the effective number of sequences in the MSA (N_{eff}) normalized by sequence length ($n = 377$). The number of effective sequences correlates with this measure of distogram accuracy ($r = 0.634$). **c**, Structure accuracy measures, computed on the test set ($n = 377$), for gradient descent optimization of different forms of the potential. Top, removing terms in the potential, and showing the effect of following optimization with Rosetta relax. 'P' shows the significance of the potential giving different results from 'Full', for a two-tailed paired data *t*-test. 'Bins' shows the number of bins fitted by the spline before extrapolation and the number in the full distribution. In CASP13, splines were fitted to the first 51 of 64 bins. Bottom, reducing the resolution of the distogram distributions. The original 64-bin distogram predictions are repeatedly downsampled by a factor of 2 by summing adjacent bins, in each case with constant extrapolation beyond 18 Å (the last quarter of the bins). The two-level potential in the final row, which was designed to compare with contact predictions, is constructed by summing the probability mass below 8 Å and between 8 and 14 Å, with constant extrapolation beyond 14 Å. The TM scores in this table are plotted in Fig. 4b.

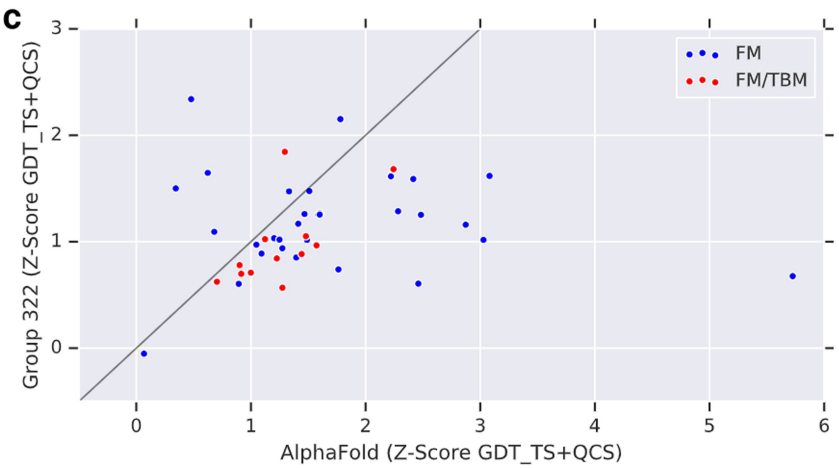


Extended Data Fig. 4 | TM score versus per-target computation time computed as an average over the test set. Structure realization requires a modest computation budget, which can be parallelized over multiple machines. Full optimization with noisy restarts (orange) is compared with initialization from sampled torsions (blue). Computation is measured as the product of the number of (CPU-based) machines and time elapsed and can be largely parallelized. Longer targets take longer to optimize. Figure 2e shows how the TM score increases with the number of repeats of gradient descent. $n = 377$.

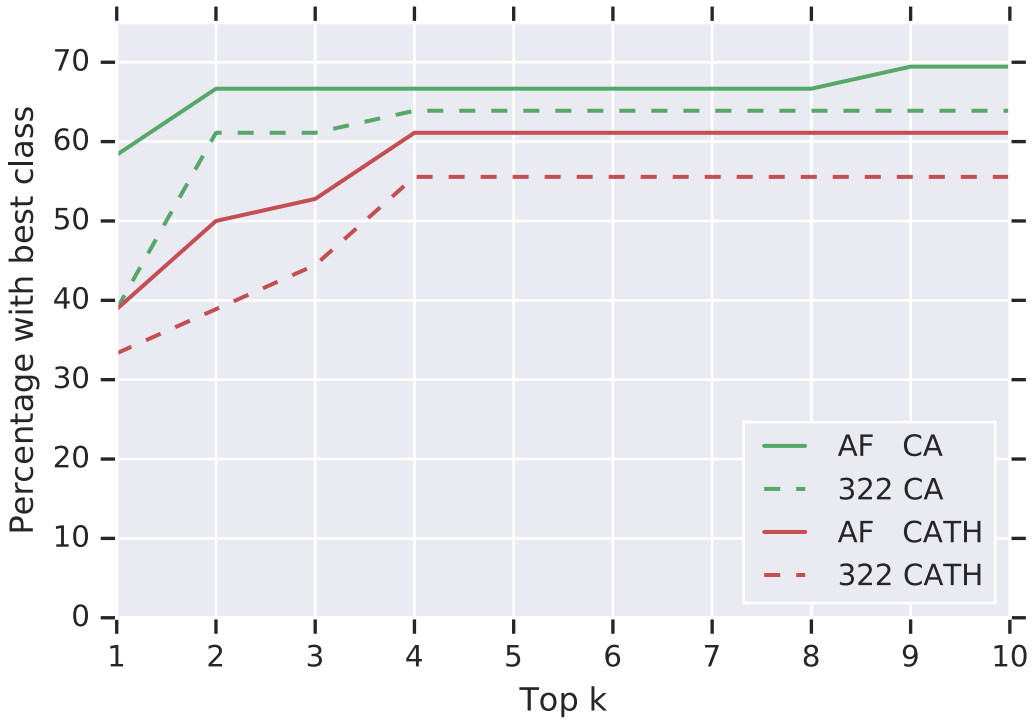


b

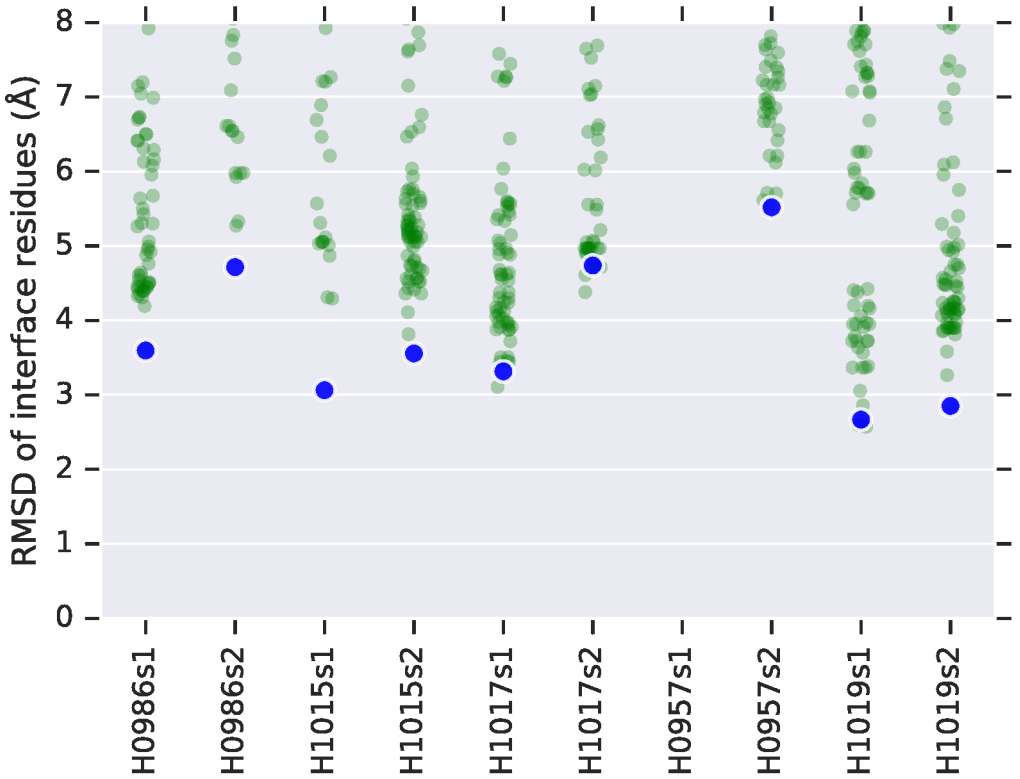
Method	FM	TBM/FM	TBM	All
Top-1	58.0	68.1	76.2	69.9
Best-of-5	62.6	73.6	78.6	73.2
1× gradient descent	58.4	71.6	76.3	70.4
1× fragment assembly	54.3	69.9	74.5	68.0



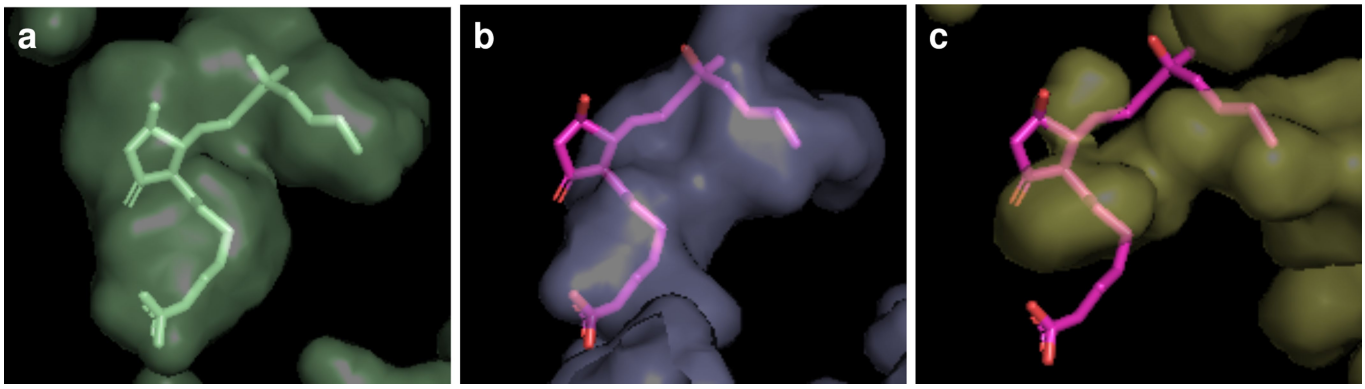
Extended Data Fig. 5 | AlphaFold CASP13 results. a, The TM score for each of the five AlphaFold CASP13 submissions are shown. Simulated annealing with fragment assembly entries are shown in blue. Gradient-descent entries are shown in yellow. Gradient descent was only used for targets T0975 and later, so to the left of the black line we also show the results for a single 'back-fill' run of gradient descent for each earlier target using the deployed system. T0999 (1,589 residues) was manually segmented based on HHpred[51] homology matching. **b**, Average TM scores of the AlphaFold CASP13 submissions (n = 104 domains), comparing the first model submitted, the best-of-five model (submission with highest GDT_TS), a single run of full-chain gradient descent (a CASP13 run for T0975 and later, back-fill for earlier targets) and a single CASP13 run of fragment assembly with domain segmentation (using a gradient descent submission for T0999). **c**, The formula-standardized (z) scores of the assessors for GDT TS + QCS[52], best-of-five for CASP FM (n = 31) and FM/TBM (n = 12) domains comparing AlphaFold with the closest competitor (group 322), coloured by domain category. AlphaFold performs better (P = 0.0032, one-tailed paired statistic t-test).



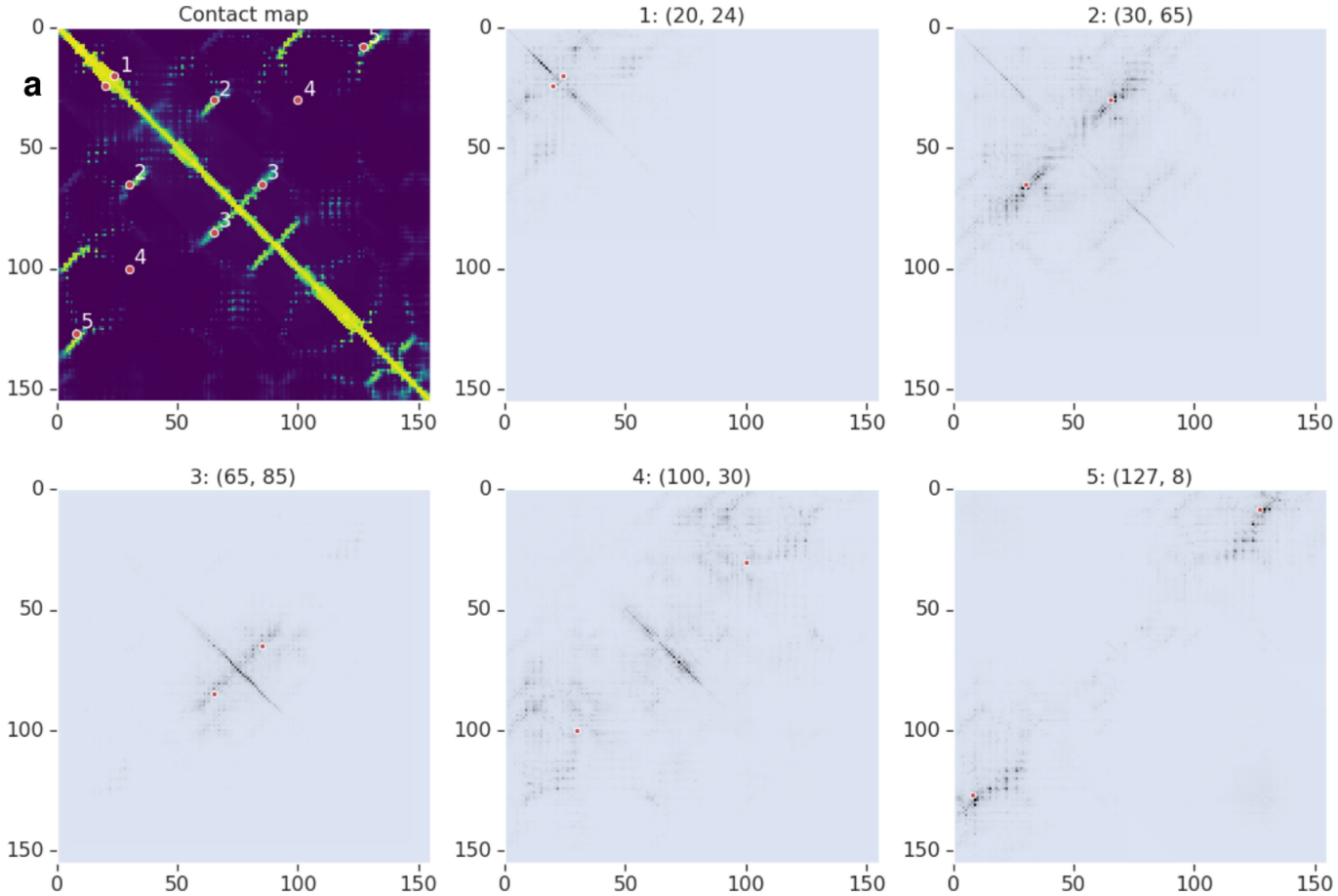
Extended Data Fig. 6 | Correct fold identification by structural search in CATH. Often protein function can be inferred by finding homologous proteins of known function. Here we show that the FM predictions of AlphaFold give greater accuracy in a structure-based search for homologous domains in the CATH database. For each of the FM or TBM/FM domains, the top-one submission and ground truth are compared to all 30,744 CATH S40 non-redundant domains with TM-align[53]. For the 36 domains for which there is a good ground-truth match (score > 0.5), we show the percentage of decoys for which a domain with the same CATH code (CATH in red, CA in green; CAT results are close to CATH results) as the top ground-truth match is in the top-*k* matches with score > 0.5. Curves are shown for AlphaFold and the next-best group (322). AlphaFold predictions determine the matching fold more accurately. Determination of the matching CATH domain can provide insights into the function of a new protein.



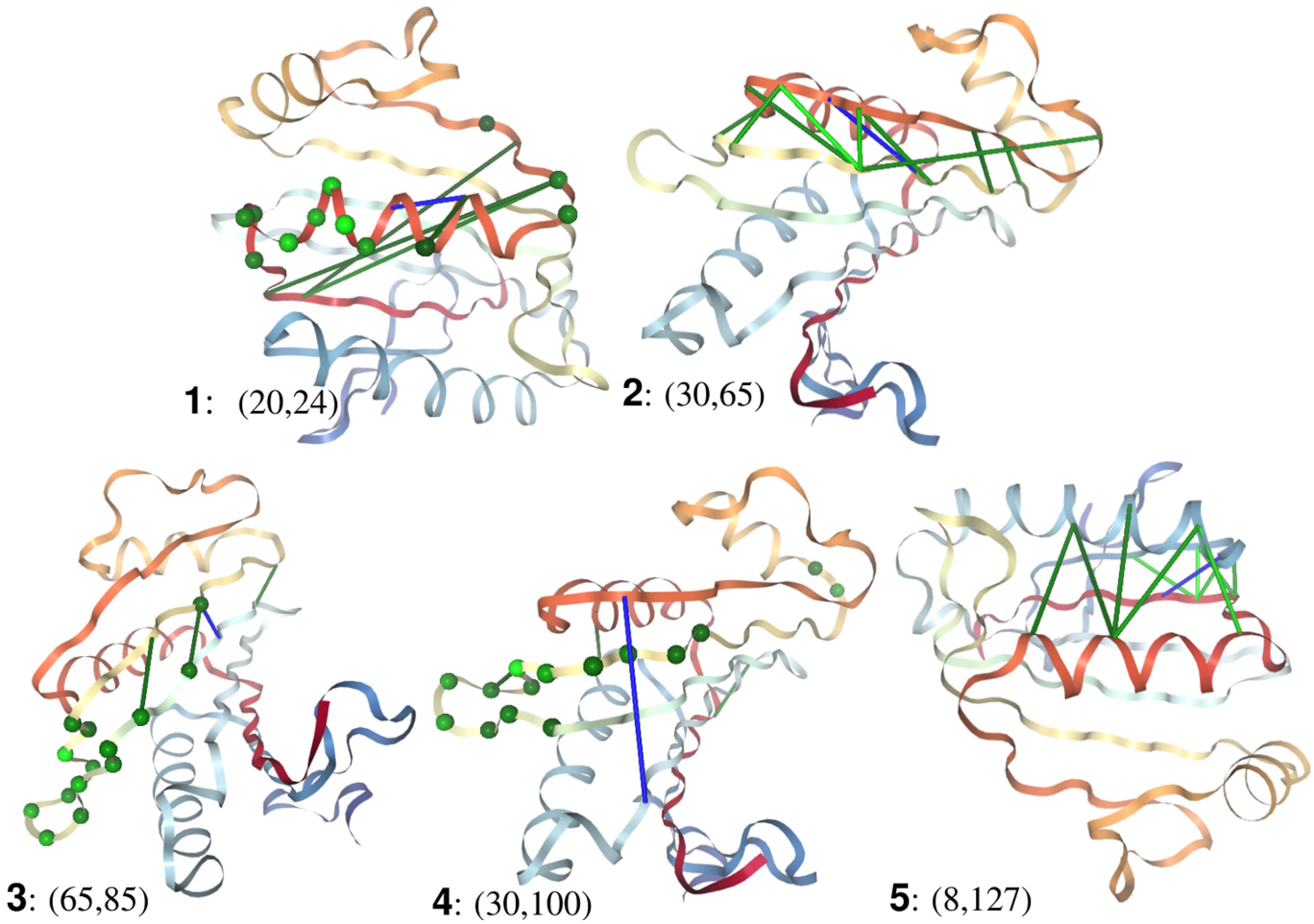
Extended Data Fig. 7 | Accuracy of predictions for interfaces. Protein–protein interaction is an important domain for understanding protein function that has hitherto largely been limited to template-based models because of the need for high-accuracy predictions, although there has been moderate success[54] in docking with predicted structures up to 6 Å r.m.s.d. This figure shows that the predictions by AlphaFold improve accuracy in the interface regions of chains in hetero-dimer structures and are probably better candidates for docking, although docking did not form part of the AlphaFold system and all submissions were for isolated chains rather than complexes. For the five all-groups heterodimer CASP13 targets, the full-atom r.m.s.d. values of the interface residues (residues with a ground-truth inter-chain heavy-atom distance <10 Å) are computed for the chain submissions of all groups (green), relative to the target complex. Results >8 Å are not shown. AlphaFold (blue) achieves consistently high accuracy interface regions and, for 4 out of 5 targets, predicts interfaces below <5 Å for both chains.



Extended Data Fig. 8 | Ligand pocket visualizations for T1011. T1011 (PDB 6M9T) is the EP3 receptor bound to misoprostol-FA[55]. **a**, The native structure showing the ligand in a pocket. **b**, **c**, Submission 5 (78.0 GDT TS) by AlphaFold (**b**), made without knowledge of the ligand, shows a pocket more similar to the true pocket than that of the best other submission (322, model 3, 68.7 GDT TS) (**c**). Both submissions are aligned to the native protein using the same subset of residues from the helices close to the ligand pocket and visualized with the interior pocket together with the native ligand position.



Extended Data Fig. 9 | Attribution map of distogram network. The contact probability map of T0986s2, and the summed absolute value of the Integrated Gradient, $\sum_c |S^{IJ}_{i,j,c}|$, of the input two-dimensional features with respect to the expected distance between five different pairs of residues (I,J): (1) a helix self-contact, (2) a long-range strand–strand contact, (3) a medium-range strand–strand contact, (4) a non-contact and (5) a very long-range strand–strand contact. Each pair is shown as two red dots on the diagrams. Darker colours indicate a higher attribution weight.



Extended Data Fig. 10 | Attribution shown on predicted structure. For T0986s2 (TM score 0.8), the top 10 input pairs, including self-pairs, with the highest attribution weight for each of the five output pairs shown in Extended Data Fig. 9, are shown as lines (or spheres for self-pairs) coloured by sensitivity, lighter green colours indicate more sensitive, and the output pair is shown as a blue line.

Article

Highly accurate protein structure prediction for the human proteome

https://doi.org/10.1038/s41586-021-03828-1

Received: 11 May 2021

Accepted: 16 July 2021

Published online: 22 July 2021

Open access

Check for updates

Kathryn Tunyasuvunakool[1]✉, Jonas Adler[1], Zachary Wu[1], Tim Green[1], Michal Zielinski[1], Augustin Žídek[1], Alex Bridgland[1], Andrew Cowie[1], Clemens Meyer[1], Agata Laydon[1], Sameer Velankar[2], Gerard J. Kleywegt[2], Alex Bateman[2], Richard Evans[1], Alexander Pritzel[1], Michael Figurnov[1], Olaf Ronneberger[1], Russ Bates[1], Simon A. A. Kohl[1], Anna Potapenko[1], Andrew J. Ballard[1], Bernardino Romera-Paredes[1], Stanislav Nikolov[1], Rishub Jain[1], Ellen Clancy[1], David Reiman[1], Stig Petersen[1], Andrew W. Senior[1], Koray Kavukcuoglu[1], Ewan Birney[2], Pushmeet Kohli[1], John Jumper[1,3]✉ & Demis Hassabis[1,3]✉

Protein structures can provide invaluable information, both for reasoning about biological processes and for enabling interventions such as structure-based drug development or targeted mutagenesis. After decades of effort, 17% of the total residues in human protein sequences are covered by an experimentally determined structure[1]. Here we markedly expand the structural coverage of the proteome by applying the state-of-the-art machine learning method, AlphaFold[2], at a scale that covers almost the entire human proteome (98.5% of human proteins). The resulting dataset covers 58% of residues with a confident prediction, of which a subset (36% of all residues) have very high confidence. We introduce several metrics developed by building on the AlphaFold model and use them to interpret the dataset, identifying strong multi-domain predictions as well as regions that are likely to be disordered. Finally, we provide some case studies to illustrate how high-quality predictions could be used to generate biological hypotheses. We are making our predictions freely available to the community and anticipate that routine large-scale and high-accuracy structure prediction will become an important tool that will allow new questions to be addressed from a structural perspective.

The monumental success of the human genome project revealed new worlds of protein-coding genes, and many researchers set out to map these proteins to their structures[3,4]. Thanks to the efforts of individual laboratories and dedicated structural genomics initiatives, more than 50,000 human protein structures have now been deposited, making *Homo sapiens* by far the best represented species in the Protein Data Bank (PDB)[5]. Despite this intensive study, only 35% of human proteins map to a PDB entry, and in many cases the structure covers only a fragment of the sequence[6]. Experimental structure determination requires overcoming many time-consuming hurdles: the protein must be produced in sufficient quantities and purified, appropriate sample preparation conditions chosen and high-quality datasets collected. A target may prove intractable at any stage, and depending on the chosen method, properties such as protein size, the presence of transmembrane regions, presence of disorder or susceptibility to conformational change can be a hindrance[7,8]. As such, full structural coverage of the proteome remains an outstanding challenge.

Protein structure prediction contributes to closing this gap by providing actionable structural hypotheses quickly and at scale. Previous large-scale structure prediction studies have addressed protein families[9–12], specific functional classes[13,14], domains identified within whole proteomes[15] and, in some cases, full chains or complexes[16,17]. In particular, projects such as the SWISS-MODEL Repository, Genome3D and ModBase have made valuable contributions by providing access to large numbers of structures and encouraging their free use by the community[17–19]. Related protein bioinformatics fields have developed alongside structure prediction, including protein design[20,21], function annotation[22–24], disorder prediction[25], and domain identification and classification[26–28]. Although some of our analyses are inspired by these previous studies, here we focus mainly on structural investigations for which scale and accuracy are particularly beneficial.

Structure prediction has seen substantial progress in recent years, as evidenced by the results of the biennial Critical Assessment of protein Structure Prediction (CASP)[29,30]. In particular, the latest version of AlphaFold was entered in CASP14 under the team name 'AlphaFold2'. This system used a completely different model from our CASP13 entry[31], and demonstrated a considerable improvement over previous methods in terms of providing routinely high accuracy[29,30]. Backbone predictions with sub-Ångström root mean square deviation (Cα r.m.s.d.) are now common, and side chains are increasingly accurate[2]. Good results can often be achieved even for challenging proteins without a template structure in the PDB, or with relatively few related sequences to build a multiple sequence alignment (MSA)[2]. These improvements are important, because more accurate models permit a wider range of

[1]DeepMind, London, UK. [2]European Molecular Biology Laboratory, European Bioinformatics Institute, Hinxton, UK. [3]These authors contributed equally: John Jumper, Demis Hassabis.
✉e-mail: ktkool@deepmind.com; jumper@deepmind.com; dhcontact@deepmind.com

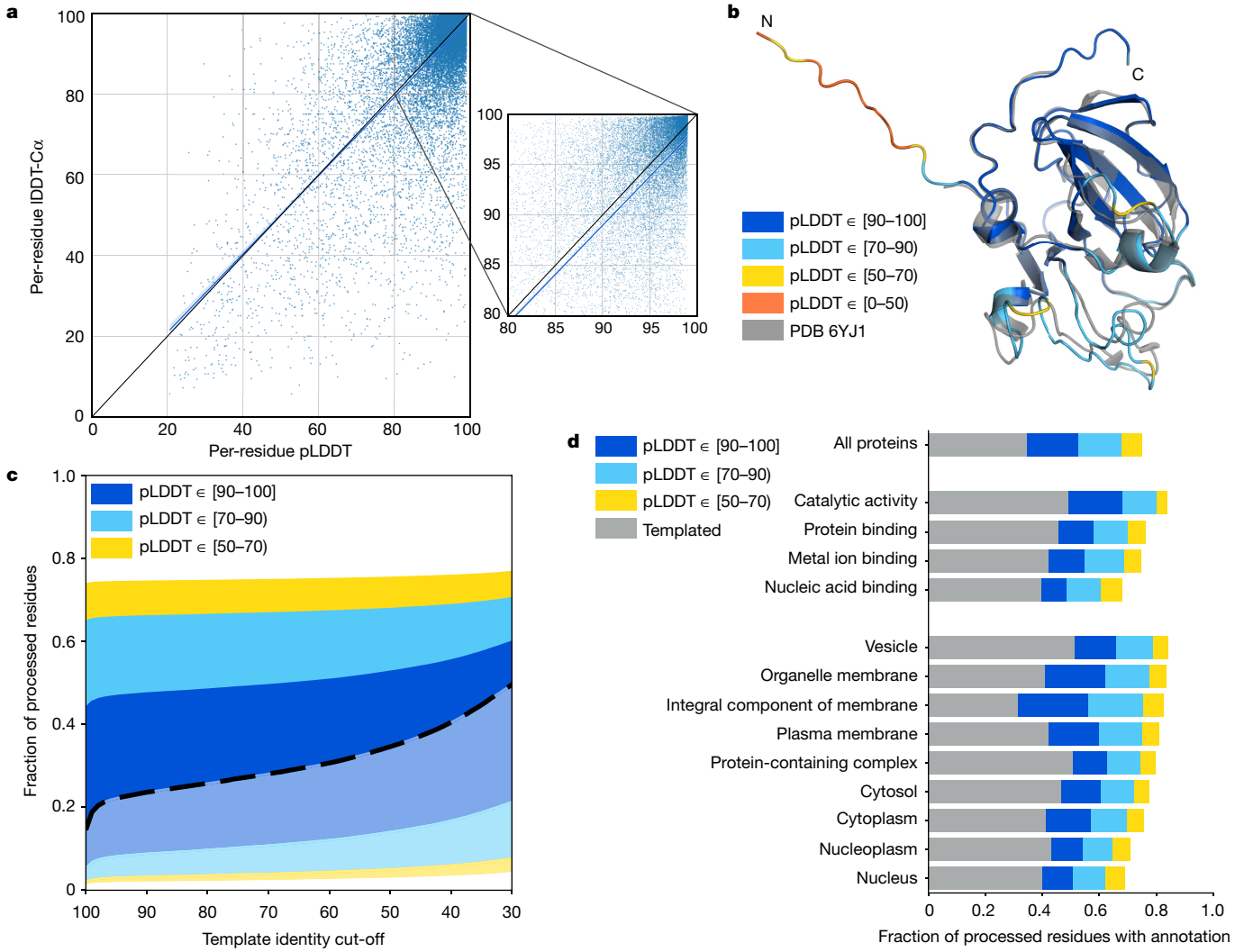


Fig. 1 | Model confidence and added coverage. a, Correlation between per-residue pLDDT and lDDT-Cα. Data are based on a held-out set of recent PDB chains (Methods) filtered to those with a reported resolution of <3.5 Å (n = 10,215 chains and 2,756,569 residues). The scatterplot shows a subsample (1% of residues), with the blue line showing a least-squares linear fit and the shaded region a 95% confidence interval estimated with 1,000 bootstrap samples. The black line shows $x = y$, for comparison. The smaller plot is a magnified region of the larger one. On the full dataset, the Pearson's $r = 0.73$ and the least-squares linear fit is $y = (0.967 \pm 0.001) \times x + (1.9 \pm 0.1)$. **b**, AlphaFold prediction and experimental structure for a CASP14 target (PDB: 6YJ1)[64]. The prediction is coloured by model confidence band, and the N terminus is an expression tag included in CASP but unresolved in the PDB structure. **c**, AlphaFold model confidence on all residues for which a prediction was produced (n = 10,537,122 residues). Residues covered by a template at the specified identity level are shown in a lighter colour and a heavy dashed line separates these from residues without a template. **d**, Added residue-level coverage of the proteome for high-level GO terms, on top of residues covered by a template with sequence identity of more than 50%. Based on the same human proteome dataset as in **c** (n = 10,537,122 residues).

applications: not only homology search and putative function assignment, but also molecular replacement and druggable pocket detection, for instance[32–34]. In light of this, we applied the current state-of-the-art method—AlphaFold—to the human proteome. All of our predictions can be accessed freely at https://alphafold.ebi.ac.uk/, hosted by the European Bioinformatics Institute.

Model confidence and added coverage

We predicted structures for the UniProt human reference proteome (one representative sequence per gene), with an upper length limit of 2,700 residues[6]. The final dataset covers 98.5% of human proteins with a full chain prediction.

For the resulting predictions to be practically useful, they must come with a well-calibrated and sequence-resolved confidence measure. The latter point is particularly important when predicting full chains, as we expect to see high confidence on domains but low confidence on linkers and unstructured regions (Extended Data Fig. 1). To this end, AlphaFold produces a per-residue confidence metric called predicted local distance difference test (pLDDT) on a scale from 0 to 100. pLDDT estimates how well the prediction would agree with an experimental structure based on the local distance difference test Cα (lDDT-Cα)[35]. It has been shown to be well-calibrated (Fig. 1a, Extended Data Fig. 2 and Extended Data Table 1) and full details on how the pLDDT is produced are given in the supplementary information of the companion AlphaFold paper[2].

We consider a prediction highly accurate when—in addition to a good backbone prediction—the side chains are frequently correctly oriented. On this basis, pLDDT > 90 is taken as the high accuracy cut-off, above which AlphaFold χ_1 rotamers are 80% correct for a recent PDB

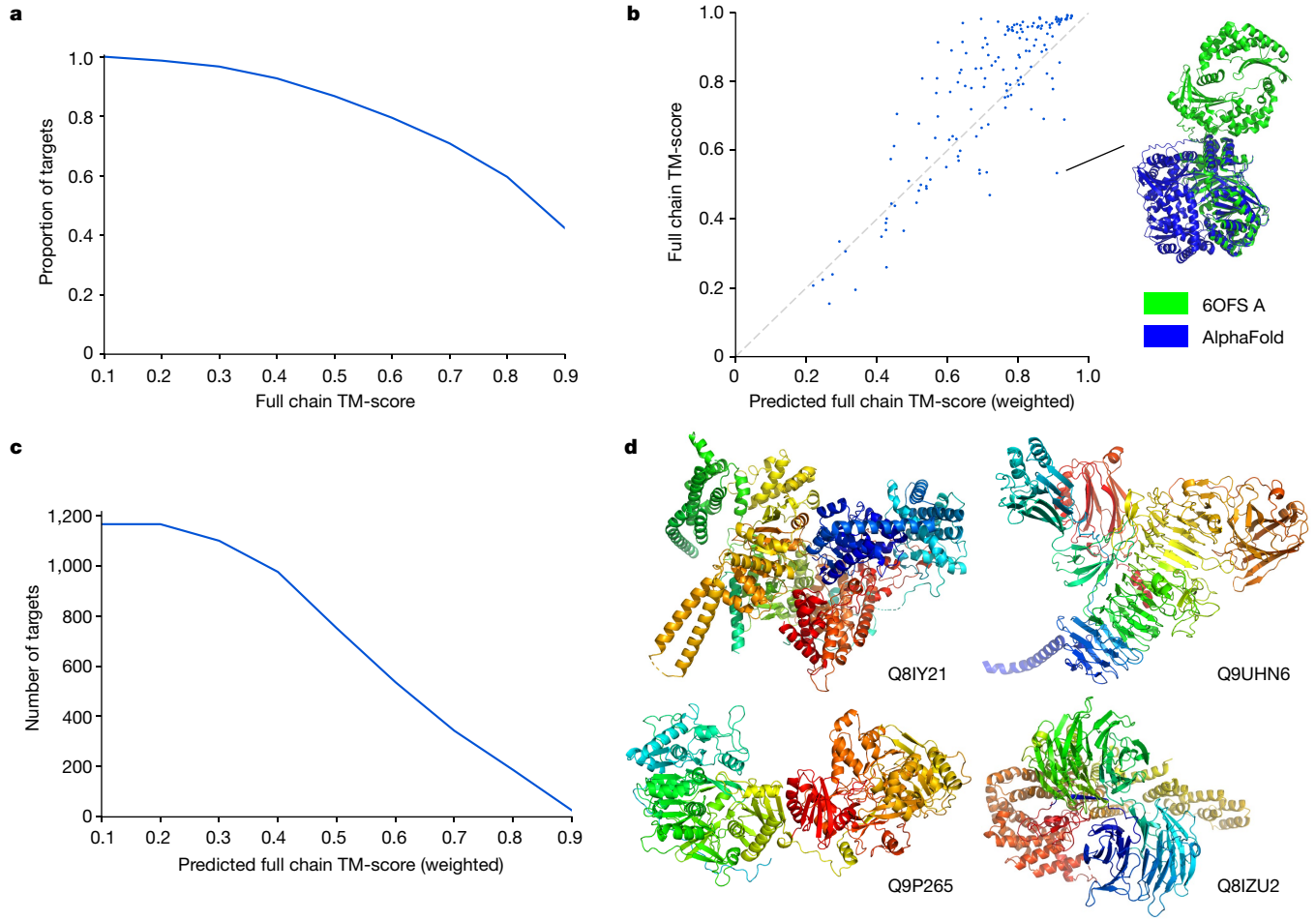


Fig. 2 | Full chain structure prediction. a, TM-score distribution for AlphaFold evaluated on a held-out set of template-filtered, long PDB chains ($n = 151$ chains). Includes recent PDB proteins with more than 800 resolved residues and best 50% coverage template below 30% identity. **b**, Correlation between full chain TM-score and pTM on the same set ($n = 151$ chains), Pearson's $r = 0.84$. The ground truth and predicted structure are shown for the most over-optimistic outlier (PDB: 6OFS, chain A). **c**, pTM distribution on a subset of the human proteome that we expect to be enriched for structurally novel multidomain proteins ($n = 1{,}165$ chains). Human proteome predictions comprise more than 600 confident residues (more than 50% coverage) and no proteins with 50% coverage templates. **d**, Four of the top hits from the set shown in **c**, filtering by pTM > 0.8 and sorting by number of confident residues. Proteins are labelled by their UniProt accession. For clarity, regions with pLDDT < 50 are hidden, as are isolated smaller regions that were left after this cropping.

test dataset (Extended Data Fig. 3). A lower cut-off of pLDDT > 70 corresponds to a generally correct backbone prediction (Extended Data Table 2). The accuracy of AlphaFold within a number of pLDDT bands is illustrated for an example protein in Fig. 1b.

Of the human proteome, 35.7% of total residues fall within the highest accuracy band (corresponding to 38.6% of residues for which a prediction was produced) (Fig. 1c). This is double the number of residues covered by an experimental structure. In total, 58.0% of residues were predicted confidently (pLDDT > 70), indicating that we also add substantial coverage for sequences without a good template in PDB (with a sequence identity below 30%). At the per-protein level, 43.8% of proteins have a confident prediction on at least three quarters of their sequence, while 1,290 proteins contain a substantial region (more than 200 residues) with pLDDT ≥ 70 and no good template.

The dataset adds high-quality structural models across a broad range of Gene Ontology (GO) terms[36,37], including pharmaceutically relevant classes such as enzymes and membrane proteins[38] (Fig. 1d). Membrane proteins, in particular, are generally underrepresented in the PDB because they have historically been challenging experimental targets. This shows that AlphaFold is able to produce confident predictions even for protein classes that are not abundant within its training set.

We note that the accuracy of AlphaFold was validated in CASP14[2], which focuses on challenging proteins that are dissimilar to structures already available in the PDB. By contrast, many human proteins have templates with high sequence identity. To evaluate the applicability of AlphaFold to this collection, we predicted structures for 1 year of targets from the Continuous Automated Model Evaluation (CAMEO) benchmark[39,40]—a structure-prediction assessment that measures a wider range of difficulties. We find that AlphaFold adds substantial accuracy over the BestSingleStructuralTemplate baseline of CAMEO across a wide range of levels of template identity (Extended Data Fig. 4).

Prediction of full-length protein chains

Many previous large-scale structure prediction efforts have focused on domains—regions of the sequence that fold independently[9–11,15]. Here we process full-length protein chains. There are several motivations for this. Restricting the prediction to pre-identified domains risks missing structured regions that have yet to be annotated. It also discards contextual information from the rest of the sequence, which might be useful in cases in which two or more domains interact substantially.

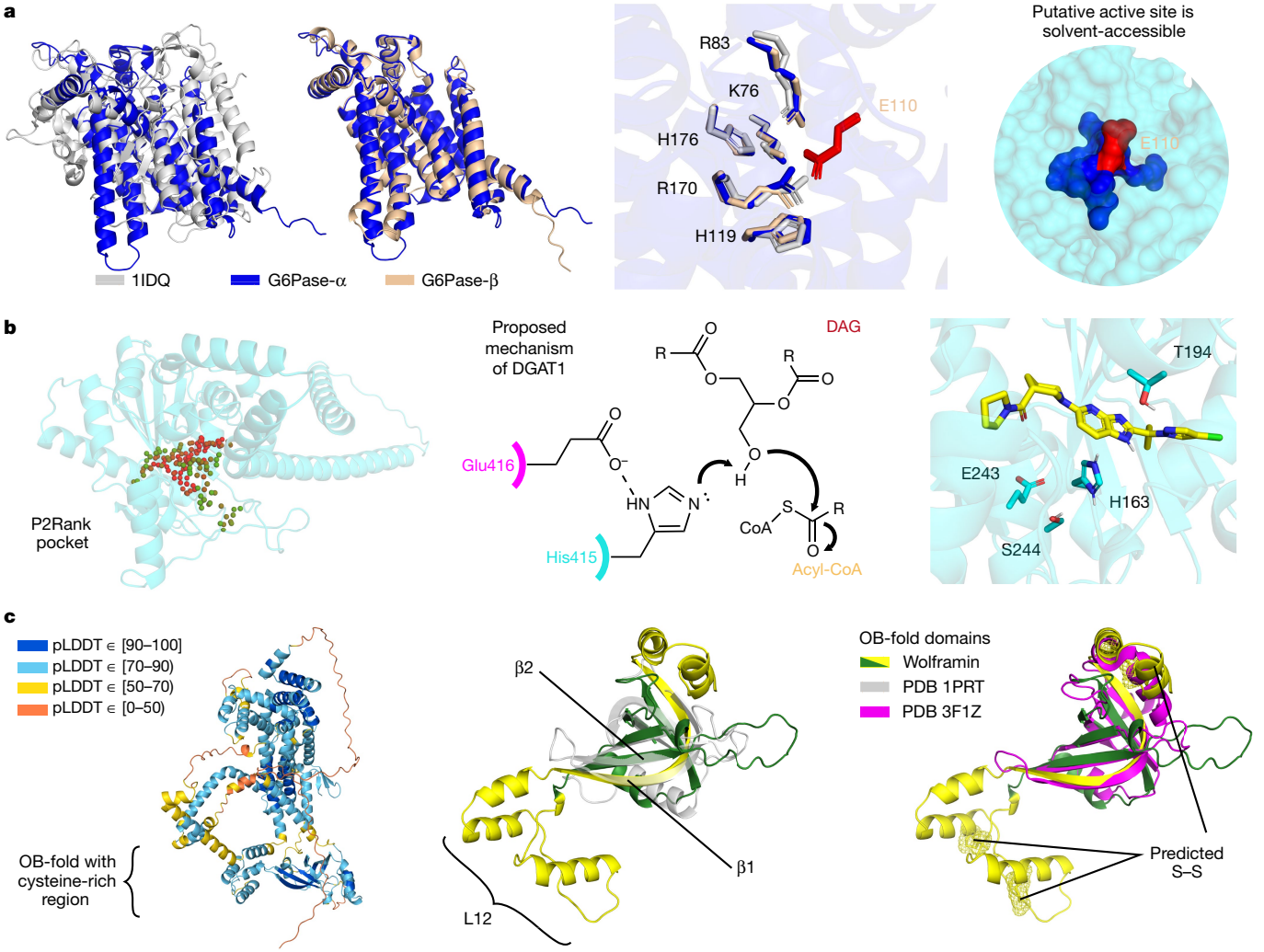


Fig. 3 | Highlighted structure predictions. a, Left, comparison of the active sites of two G6Pases (G6Pase-α and G6Pase-β) and a chloroperoxidase (PDB 1IDQ). The G6Pases are glucose-forming enzymes that contain a conserved, solvent-accessible glutamate (red; right) opposite the shared active-site residues (middle). **b**, Left, pocket prediction (P2Rank[65]) identifies a putative binding pocket for DGAT2, which is involved in body-fat synthesis. Red and green spheres represent the ligandability scores by P2Rank of 1 and 0, respectively. Middle, a proposed mechanism for DGAT1[51] activates the substrate with Glu416 and His415, which have analogous residues in the DGAT2 pocket. The docked inhibitor is well placed for polar interactions with His163 and Thr194 (right). The chemical structure (middle) is adapted from ref. [51]. **c**, Predicted structure of wolframin, mutations in which cause Wolfram syndrome. Although there are regions in wolframin with low pLDDT (left), we could identify an OB-fold region (green/yellow), with a comparable core to a prototypical OB-fold (grey; middle). However, the most similar PDB chain (magenta; right) lacks the conserved cysteine-rich region (yellow) of our prediction. This region forms the characteristic β1 strand and an extended L12 loop, and is predicted to contain three disulfide bridges (yellow mesh).

Finally, the full chain approach lets the model attempt an inter-domain packing prediction.

Inter-domain accuracy was assessed at CASP14, and AlphaFold outperformed other methods[41]. However, the assessment was based on a small target set. To further evaluate AlphaFold on long multi-domain proteins, we compiled a test dataset of recent PDB chains that were not in the training set of the model. Only chains with more than 800 resolved residues were included, and a template filter was applied (Methods). Performance on this set was evaluated using the template modelling score (TM-score[42]), which should better reflect global, as opposed to per-domain, accuracy. The results were encouraging, with 70% of predictions having a TM-score > 0.7 (Fig. 2a).

The supplementary information of the companion AlphaFold paper[2] describes how a variety of useful predictors can be built on top of the main model. In particular, we can predict the residues that are likely to be experimentally resolved, and use them to produce a predicted TM-score (pTM), in which the contribution of each residue is weighted by the probability of it being resolved (Supplementary Methods 1). The motivation for the weighting is to downweight unstructured parts of the prediction, producing a metric that better reflects the confidence of the model about the packing of the structured domains that are present. On the same recent PDB test dataset, pTM correlates well with the actual TM-score (Pearson's $r = 0.84$) (Fig. 2b). Notably, although some outliers in this plot are genuine failure cases, others appear to be plausible alternate conformations (for example, 6OFS chain A[43] in Fig. 2b).

We computed pTM scores for the human proteome, in an effort to identify multi-domain predictions that could feature novel domain packings. The criteria applied were a pLDDT > 70 on at least 600 residues constituting over half the sequence, with no template hit covering more than half the sequence. The distribution of pTM scores after applying the above filters is shown in Fig. 2c. Note that we would not

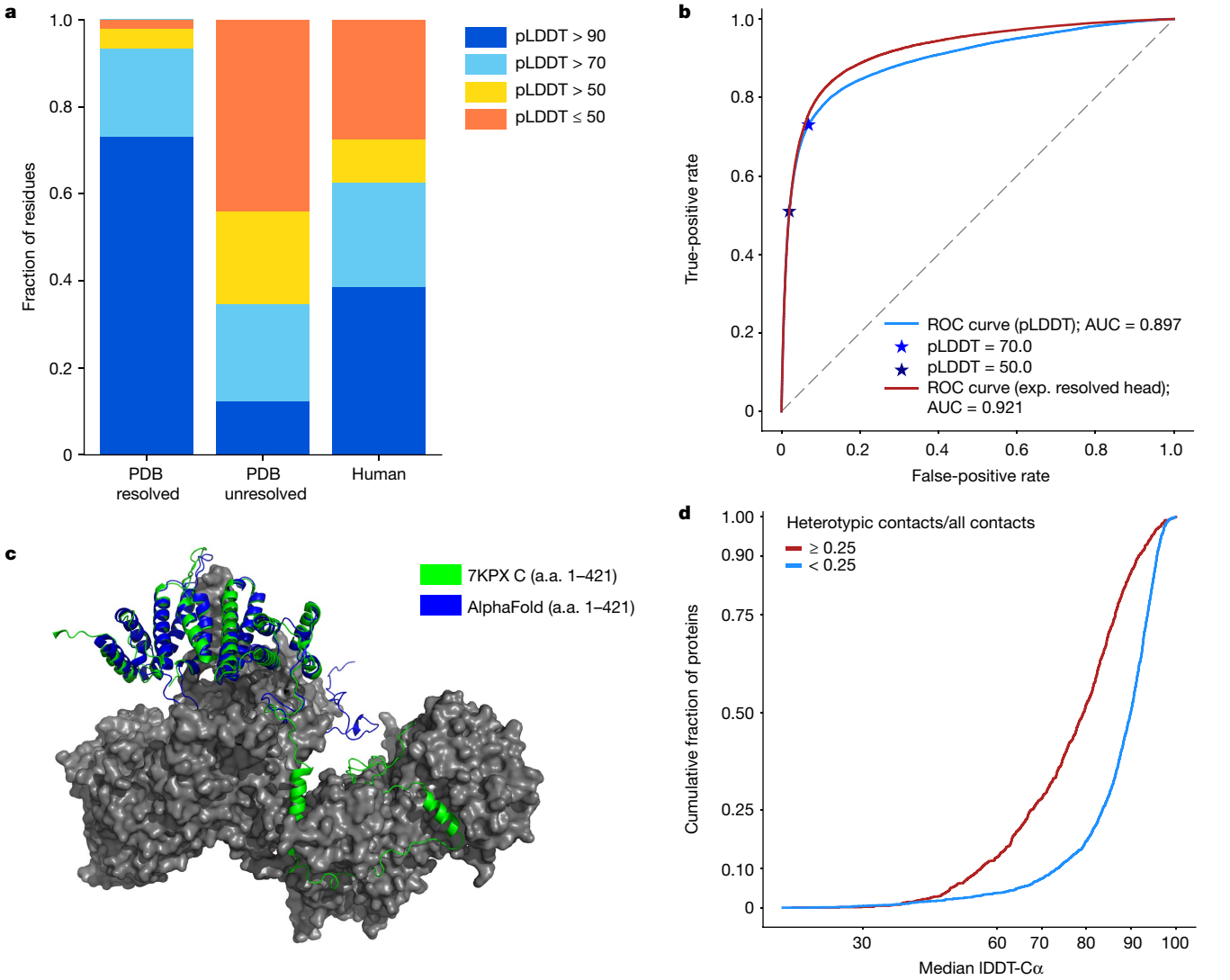


Fig. 4 | Low-confidence regions. a, pLDDT distribution of the resolved parts of PDB sequences (n = 3,440,359 residues), the unresolved parts of PDB sequences (n = 589,079 residues) and the human proteome (n = 10,537,122 residues). **b**, Performance of pLDDT and the experimentally resolved head of AlphaFold as disorder predictors on the CAID Disprot-PDB benchmark dataset (n = 178,124 residues). **c**, An example low-confidence prediction aligned to the corresponding PDB submission (7KPX chain C)[66]. The globular domain is well-predicted but the extended interface exhibits low pLDDT and is incorrect apart from some of the secondary structure. a.a., amino acid. **d**, A high ratio of heterotypic contacts is associated with a lower AlphaFold accuracy on the recent PDB dataset, restricted to proteins with fewer than 40% of residues with template identity above 30% (n = 3,007 chains) (Methods). The ratio of heterotypic contacts is defined as: heterotypic/(intra-chain + homomeric + heterotypic).

expect uniformly high TM-scores to be achievable for this set, as some proteins will contain domains that are mobile relative to each other, with no fixed packing. Of the set, 187 proteins have pTM > 0.8 and 343 have pTM > 0.7. Although we expect the inter-domain accuracy of AlphaFold to be lower than its within-domain accuracy, this set should nonetheless be enriched for interesting multi-domain predictions, suggesting that the dataset provides on the order of hundreds of these. Four examples—the predictions with the highest number of confident residues subject to pTM > 0.8—are shown in Fig. 2d.

Highlighted predictions

We next discuss some case study predictions and the insights that they may provide. All predictions presented are de novo, lacking any template with 25% sequence identity or more covering 20% of the sequence. Our discussion concerns biological hypotheses, which would ultimately need to be confirmed by experimental studies.

Glucose-6-phosphatase

G6Pase-α (UniProt P35575) is a membrane-bound enzyme that catalyses the final step in glucose synthesis; it is therefore of critical importance to maintaining blood sugar levels. To our knowledge, no experimental structure exists, but previous studies have attempted to characterize the transmembrane topology[44] and active site[45]. Our prediction has very high confidence (median pLDDT of 95.5) and gives a nine-helix topology with the putative active site accessible via an entry tunnel that is roughly in line with the surface of the endoplasmic reticulum (Fig. 3a and Supplementary Video 1). Positively charged residues in our prediction (median pLDDT of 96.6) align closely with the previously identified active site homologue in a fungal vanadium chloroperoxidase (PDB 1IDQ; r.m.s.d. of 0.56 Å; 49 out of 51 aligned atoms)[46]. As these enzymes have distinct functions, we investigated our prediction for clues about substrate specificity. In the G6Pase-α binding pocket face, opposite the residues shared with the chloroperoxidase, we predict a conserved glutamate (Glu110) that is also present in our G6Pase-β

prediction (Glu105) but not in the chloroperoxidase (Fig. 3a). The glutamate could stabilize the binding pocket in a closed conformation, forming salt bridges with positively charged residues there. It is also the most solvent-exposed residue of the putative active site, suggesting a possible gating function. To our knowledge, this residue has not been discussed previously and illustrates the novel mechanistic hypotheses that can be obtained from high-quality structure predictions.

Diacylglycerol *O*-acyltransferase 2

Triacylglycerol synthesis is responsible for storing excess metabolic energy as fat in adipose tissue. DGAT2 (UniProt Q96PD7) is one of two essential acyltransferases catalysing the final acyl addition in this pathway, and inhibiting DGAT2 has been shown to improve liver function in mouse models of liver disease[47]. With our highly confident predicted structure (median pLDDT of 95.9), we set out to identify the binding pocket for a known inhibitor, PF-06424439 (ref. [48]). We identified a pocket (median pLDDT of 93.7) in which we were able to dock the inhibitor and observe specific interactions (Fig. 3b) that were not recapitulated in a negative example[49] (Extended Data Fig. 5 and Supplementary Methods 2). DGAT2 has an evolutionarily divergent but biochemically similar analogue, diacylglycerol *O*-acyltransferase 1 (DGAT1)[50]. Within the binding pocket of DGAT2, we identified residues (Glu243 and His163) (Fig. 3b) that are analogous to the proposed catalytic residues in DGAT1 (His415 and Glu416)[51], although we note that the nearby Ser244 in DGAT2 may present an alternative mechanism through an acyl-enzyme intermediate. Previous experimental research with DGAT2 has shown that mutating His163 has a stronger deleterious effect than mutating a histidine that is two residues away[52]. Additionally, Glu243 and His163 are conserved across species[50], supporting this hypothesized catalytic geometry.

Wolframin

Wolframin (UniProt O76024) is a transmembrane protein localized to the endoplasmic reticulum. Mutations in the *WFS1* gene are associated with Wolfram syndrome 1, a neurodegenerative disease characterized by early onset diabetes, gradual visual and hearing loss, and early death[53,54]. Given the lower confidence in our full prediction (median pLDDT of 81.7) (Fig. 3c), we proposed identifying regions that are unique to this structure. A recent evolutionary analysis suggested domains for wolframin, which our prediction largely supports[55]. An interesting distinction is the incorporation of a cysteine-rich domain (Fig. 3c, yellow) to the oligonucleotide binding (OB) fold (Fig. 3c, green and yellow) as the characteristic β1 strand[56]. The cysteine-rich region then forms an extended L12 loop with two predicted disulfide bridges, before looping back to the prototypical β2 strand. Comparing our prediction for this region (median pLDDT of 86.0) to existing PDB chains using TM-align[42,57] identified 3F1Z[58] as the most similar known chain (TM-score of 0.472) (Fig. 3c, magenta). Despite being the most similar chain, 3F1Z lacks the cysteines that are present in wolframin, which could form disulfide cross-links in the endoplasmic reticulum[59]. As this region is hypothesized to recruit other proteins[55], these structural insights are probably important to understanding its partners.

Regions without a confident prediction

As we are applying AlphaFold to the entire human proteome, we would expect a considerable percentage of residues to be contained in regions that are always or sometimes disordered in solution. Disorder is common in the proteomes of eukaryotes[60,61], and one previous study[62] estimated that the percentage of disordered residues in the human proteome is between 37% and 50%. Thus disorder will have a large role when we consider a comprehensive set of predictions that covers an entire proteome.

Furthermore, we observed a large difference in the pLDDT distribution between resolved and unresolved residues in PDB sequences (Fig. 4a). To investigate this connection, we evaluated pLDDT as a disorder predictor on the Critical Assessment of protein Intrinsic Disorder prediction (CAID) benchmark dataset[25]. The results showed pLDDT to be a competitive disorder predictor compared with the current state of the art (SPOT-Disorder2[63]), with an area under the curve (AUC) of 0.897 (Fig. 4b). Moreover, the supplementary information of the companion AlphaFold paper[2] describes an 'experimentally resolved head', which is specifically trained for the task of predicting whether a residue will be resolved in an experimental structure. The experimentally resolved head performed even better on the CAID benchmark, with an AUC of 0.921.

These disorder prediction results suggest that a considerable percentage of low-confidence residues may be explained by some form of disorder, but we caution that this could encompass both regions that are intrinsically disordered and regions that are structured only in complex. A potential example of the latter scenario drawn from a recent PDB structure is shown in Fig. 4c; chain C interacts extensively with the rest of the complex, such that the interface region would be unlikely to adopt the same structure outside of this context. In a systematic analysis of recent PDB chains, we observed that AlphaFold has much lower accuracy for regions in which the chain has a high percentage of heterotypic, cross-chain contacts (Fig. 4d).

In summary, our current interpretation of regions in which AlphaFold exhibits low pLDDT is that they have high likelihood of being unstructured in isolation. In the current dataset, long regions with pLDDT < 50 adopt a readily identifiable ribbon-like appearance, and should not be interpreted as structures but rather as a prediction of disorder.

Discussion

In this study, we generated comprehensive, state-of-the-art structure predictions for the human proteome. The resulting dataset makes a large contribution to the structural coverage of the proteome; particularly for tasks in which high accuracy is advantageous, such as molecular replacement or the characterization of binding sites. We also applied several metrics produced by building on the AlphaFold architecture—pLDDT, pTM and the experimentally resolved head—to demonstrate how they can be used to interpret our predictions.

Although we present several case studies to illustrate the type of insights that may be gained from these data, we recognize that there is still much more to uncover. By making our predictions available to the community via https://alphafold.ebi.ac.uk/, we hope to enable exploration of new directions in structural bioinformatics.

The parts of the human proteome that are still without a confident prediction represent directions for future research. Some proportion of these will be genuine failures, in which a fixed structure exists but the current version of AlphaFold does not predict it. In many other cases, in which the sequence is unstructured in isolation, the problem arguably falls outside the scope of single-chain structure prediction. It will be crucial to develop new methods that can address the biology of these regions—for example, by predicting the structure in complex or by predicting a distribution over possible states in the cellular milieu.

Finally, we note that the importance of the human proteome for health and medicine has led to it being intensively studied from a structural perspective. Other organisms are much less well represented in the PDB, including biologically important, medically relevant or economically important species. Structure prediction may have a more profound effect on the study of these organisms, for which fewer experimental structures are available. Looking beyond the proteome scale, the UniProt database contains hundreds of millions of proteins that have so far been addressed mainly by sequence-based methods, and for which the easy availability of structures could open up entirely new avenues of investigation. By providing scalable structure prediction with very high accuracy, AlphaFold could enable an exciting shift towards structural bioinformatics, further illuminating protein space.

Online content

Any methods, additional references, Nature Research reporting summaries, source data, extended data, supplementary information, acknowledgements, peer review information; details of author contributions and competing interests; and statements of data and code availability are available at https://doi.org/10.1038/s41586-021-03828-1.

1. SWISS-MODEL. *Homo sapiens* (human). https://swissmodel.expasy.org/repository/species/9606 (2021).
2. Jumper, J. et al. Highly accurate protein structure prediction with AlphaFold. *Nature* https://doi.org/10.1038/s41586-021-03819-2 (2021).
3. International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome. *Nature* **409**, 860–921 (2001).
4. Venter, J. C. et al. The sequence of the human genome. *Science* **291**, 1304–1351 (2001).
5. wwPDB Consortium. Protein Data Bank: the single global archive for 3D macromolecular structure data. *Nucleic Acids Res.* **47**, D520–D528 (2018).
6. The UniProt Consortium. UniProt: the universal protein knowledgebase in 2021. *Nucleic Acids Res.* **49**, D480–D489 (2021).
7. Slabinski, L. et al. The challenge of protein structure determination—lessons from structural genomics. *Protein Sci.* **16**, 2472–2482 (2007).
8. Elmlund, D., Le, S. N. & Elmlund, H. High-resolution cryo-EM: the nuts and bolts. *Curr. Opin. Struct. Biol.* **46**, 1–6 (2017).
9. Yang, J. et al. Improved protein structure prediction using predicted interresidue orientations. *Proc. Natl Acad. Sci. USA* **117**, 1496–1503 (2020).
10. Greener, J. G., Kandathil, S. M. & Jones, D. T. Deep learning extends de novo protein modelling coverage of genomes using iteratively predicted structural constraints. *Nat. Commun.* **10**, 3977 (2019).
11. Michel, M., Menéndez Hurtado, D., Uziela, K. & Elofsson, A. Large-scale structure prediction by improved contact predictions and model quality assessment. *Bioinformatics* **33**, i23–i29 (2017).
12. Ovchinnikov, S. et al. Large-scale determination of previously unsolved protein structures using evolutionary information. *eLife* **4**, e09248 (2015).
13. Zhang, J., Yang, J., Jang, R. & Zhang, Y. GPCR-I-TASSER: a hybrid approach to G protein-coupled receptor structure modeling and the application to the human genome. *Structure* **23**, 1538–1549 (2015).
14. Bender, B. J., Marlow, B. & Meiler, J. Improving homology modeling from low-sequence identity templates in Rosetta: a case study in GPCRs. *PLOS Comput. Biol.* **16**, e1007597 (2020).
15. Drew, K. et al. The Proteome Folding Project: proteome-scale prediction of structure and function. *Genome Res.* **21**, 1981–1994 (2011).
16. Xu, D. & Zhang, Y. Ab initio structure prediction for *Escherichia coli*: towards genome-wide protein structure modeling and fold assignment. *Sci. Rep.* **3**, 1895 (2013).
17. Waterhouse, A. et al. SWISS-MODEL: homology modelling of protein structures and complexes. *Nucleic Acids Res.* **46**, W296–W303 (2018).
18. Sillitoe, I. et al. Genome3D: integrating a collaborative data pipeline to expand the depth and breadth of consensus protein structure annotation. *Nucleic Acids Res.* **48**, D314–D319 (2020).
19. Pieper, U. et al. ModBase, a database of annotated comparative protein structure models and associated resources. *Nucleic Acids Res.* **42**, D336–D346 (2014).
20. Huang, P.-S., Boyken, S. E. & Baker, D. The coming of age of de novo protein design. *Nature* **537**, 320–327 (2016).
21. Kuhlman, B. & Bradley, P. Advances in protein structure prediction and design. *Nat. Rev. Mol. Cell Biol.* **20**, 681–697 (2019).
22. The Gene Ontology Consortium. The Gene Ontology Resource: 20 years and still GOing strong. *Nucleic Acids Res.* **47**, D330–D338 (2019).
23. Zhou, N. et al. The CAFA challenge reports improved protein function prediction and new functional annotations for hundreds of genes through experimental screens. *Genome Biol.* **20**, 244 (2019).
24. Gligorijević, V. et al. Structure-based protein function prediction using graph convolutional networks. *Nat. Commun.* **12**, 3168 (2021).
25. Necci, M., Piovesan, D. CAID Predictors, DisProt Curators & Tosatto, S. C. E. Critical assessment of protein intrinsic disorder prediction. *Nat. Methods* **18**, 472–481 (2021).
26. Sillitoe, I. et al. CATH: expanding the horizons of structure-based functional annotations for genome sequences. *Nucleic Acids Res.* **47**, D280–D284 (2019).
27. Andreeva, A., Kulesha, E., Gough, J. & Murzin, A. G. The SCOP database in 2020: expanded classification of representative family and superfamily domains of known protein structures. *Nucleic Acids Res.* **48**, D376–D382 (2020).
28. Mistry, J. et al. Pfam: the protein families database in 2021. *Nucleic Acids Res.* **49**, D412–D419 (2021).
29. Kryshtafovych, A., Schwede, T., Topf, M., Fidelis, K. & Moult, J. Critical assessment of methods of protein structure prediction (CASP)-round XIII. *Proteins* **87**, 1011–1020 (2019).
30. Pereira, J. et al. High-accuracy protein structure prediction in CASP14. *Proteins* https://doi.org/10.1002/prot.26171 (2021).
31. Senior, A. W. et al. Improved protein structure prediction using potentials from deep learning. *Nature* **577**, 706–710 (2020).
32. Zhang, Y. Protein structure prediction: when is it useful? *Curr. Opin. Struct. Biol.* **19**, 145–155 (2009).
33. Flower, T. G. & Hurley, J. H. Crystallographic molecular replacement using an in silico-generated search model of SARS-CoV-2 ORF8. *Protein Sci.* **30**, 728–734 (2021).
34. Egbert, M. et al. Functional assessment. https://predictioncenter.org/casp14/doc/presentations/2020_12_03_Function_Assessment_VajdaLab_KozakovLab.pdf (2020).
35. Mariani, V., Biasini, M., Barbato, A. & Schwede, T. lDDT: a local superposition-free score for comparing protein structures and models using distance difference tests. *Bioinformatics* **29**, 2722–2728 (2013).
36. The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nat. Genet.* **25**, 25–29 (2000).
37. The Gene Ontology Consortium. The Gene Ontology resource: enriching a GOld mine. *Nucleic Acids Res.* **49**, D325–D334 (2021).
38. Hopkins, A. L. & Groom, C. R. The druggable genome. *Nat. Rev. Drug Discov.* **1**, 727–730 (2002).
39. Haas, J. et al. Introducing "best single template" models as reference baseline for the Continuous Automated Model Evaluation (CAMEO). *Proteins* **87**, 1378–1387 (2019).
40. Haas, J. et al. Continuous Automated Model Evaluation (CAMEO) complementing the critical assessment of structure prediction in CASP12. *Proteins* **86**, 387–398 (2018).
41. Schaeffer, R. D., Kinch, L. & Grishin, N. CASP14: InterDomain Performance. https://predictioncenter.org/casp14/doc/presentations/2020_12_02_Interdomain_assessment1_Schaeffer.pdf (2020).
42. Zhang, Y. & Skolnick, J. Scoring function for automated assessment of protein structure template quality. *Proteins* **57**, 702–710 (2004).
43. Grinter, R. et al. Protease-associated import systems are widespread in Gram-negative bacteria. *PLoS Genet.* **15**, e1008435 (2019).
44. Pan, C.-J., Lei, K.-J., Annabi, B., Hemrika, W. & Chou, J. Y. Transmembrane topology of glucose-6-phosphatase. *J. Biol. Chem.* **273**, 6144–6148 (1998).
45. van Schaftingen, E. & Gerin, I. The glucose-6-phosphatase system. *Biochem. J.* **362**, 513–532 (2002).
46. Messerschmidt, A., Prade, L. & Wever, R. Implications for the catalytic mechanism of the vanadium-containing enzyme chloroperoxidase from the fungus *Curvularia inaequalis* by X-ray structures of the native and peroxide form. *Biol. Chem.* **378**, 309–315 (1997).
47. Amin, N. B. et al. Targeting diacylglycerol acyltransferase 2 for the treatment of nonalcoholic steatohepatitis. *Sci. Transl. Med.* **11**, eaav9701 (2019).
48. Futatsugi, K. et al. Discovery and optimization of imidazopyridine-based inhibitors of diacylglycerol acyltransferase 2 (DGAT2). *J. Med. Chem.* **58**, 7173–7185 (2015).
49. Birch, A. M. et al. Discovery of a potent, selective, and orally efficacious pyrimidinooxazinyl bicyclooctaneacetic acid diacylglycerol acyltransferase-1 inhibitor. *J. Med. Chem.* **52**, 1558–1568 (2009).
50. Cao, H. Structure-function analysis of diacylglycerol acyltransferase sequences from 70 organisms. *BMC Res. Notes* **4**, 249 (2011).
51. Wang, L. et al. Structure and mechanism of human diacylglycerol *O*-acyltransferase 1. *Nature* **581**, 329–332 (2020).
52. Stone, S. J., Levin, M. C. & Farese, R. V. Jr. Membrane topology and identification of key functional amino acid residues of murine acyl-CoA:diacylglycerol acyltransferase-2. *J. Biol. Chem.* **281**, 40273–40282 (2006).
53. Rigoli, L., Lombardo, F. & Di Bella, C. Wolfram syndrome and *WFS1* gene. *Clin. Genet.* **79**, 103–117 (2011).
54. Urano, F. Wolfram syndrome: diagnosis, management, and treatment. *Curr. Diab. Rep.* **16**, 6 (2016).
55. Schäffer, D. E., Iyer, L. M., Burroughs, A. M. & Aravind, L. Functional innovation in the evolution of the calcium-dependent system of the eukaryotic endoplasmic reticulum. *Front. Genet.* **11**, 34 (2020).
56. Guardino, K. M., Sheftic, S. R., Slattery, R. E. & Alexandrescu, A. T. Relative stabilities of conserved and non-conserved structures in the OB-fold superfamily. *Int. J. Mol. Sci.* **10**, 2412–2430 (2009).
57. Zhang, Y. & Skolnick, J. TM-align: a protein structure alignment algorithm based on the TM-score. *Nucleic Acids Res.* **33**, 2302–2309 (2005).
58. Das, D. et al. The structure of KPN03535 (gi|152972051), a novel putative lipoprotein from *Klebsiella pneumoniae*, reveals an OB-fold. *Acta Crystallogr. F* **66**, 1254–1260 (2010).
59. Fass, D. & Thorpe, C. Chemistry and enzymology of disulfide cross-linking in proteins. *Chem. Rev.* **118**, 1169–1198 (2018).
60. Basile, W., Salvatore, M., Bassot, C. & Elofsson, A. Why do eukaryotic proteins contain more intrinsically disordered regions? *PLOS Comput. Biol.* **15**, e1007186 (2019).
61. Bhowmick, A. et al. Finding our way in the dark proteome. *J. Am. Chem. Soc.* **138**, 9730–9742 (2016).
62. Oates, M. E. et al. D^2P^2: database of disordered protein predictions. *Nucleic Acids Res.* **41**, D508–D516 (2013).
63. Hanson, J., Paliwal, K. K., Litfin, T. & Zhou, Y. SPOT-Disorder2: improved protein intrinsic disorder prediction by ensembled deep learning. *Genomics Proteomics Bioinformatics* **17**, 645–656 (2019).
64. Dunne, M., Ernst, P., Sobieraj, A., Pluckthun, A. & Loessner, M. J. The M23 peptidase domain of the *Staphylococcal* phage 2638A endolysin. https://doi.org/10.2210/pdb6YJ1/pdb (2020).
65. Krivák, R. & Hoksza, D. P2Rank: machine learning based tool for rapid and accurate prediction of ligand binding sites from protein structure. *J. Cheminform.* **10**, 39 (2018).
66. Li, Y.-C. et al. Structure and noncanonical Cdk8 activation mechanism within an Argonaute-containing Mediator kinase module. *Sci. Adv.* **7**, eabd4484 (2021).

Methods

Structure prediction (human proteome)

Sequences for the human reference proteome were obtained from UniProt release 2021_02[6]. Structure prediction was attempted for all sequences with 16–2,700 amino acids; sequences with residue codes B, J, O, U, Z or X were excluded. The length ceiling of 2,700 residues does not represent an absolute limit for the method, but was chosen to keep run times manageable. The structure prediction process was largely as described in the AlphaFold paper[2], consisting of five steps: MSA construction, template search, inference with five models, model ranking based on mean pLDDT and constrained relaxation of the predicted structures. The following differences were introduced for the proteome-scale pipeline. First, the search against the metagenomics database Big Fantastic Database (BFD) was replaced with a search against 'Reduced BFD' using Jackhmmer from HMMER3[67,68]. Reduced BFD consists of a multiline FASTA file containing the first non-consensus sequence from each BFD a3m alignment. Second, the amount of ensembling was reduced by a factor of eight. At least four relaxed full chain models were successfully produced for 20,296 sequences out of 20,614 FASTA entries, covering 98.5% of proteins. Sequences with more than 2,700 residues account for the majority of exclusions. This amounts to 10,537,122 residues (92.5% of residues).

Structure prediction (recent PDB dataset)

For structure predictions of recent PDB sequences, we used a copy of the PDB downloaded on 15 February 2021. Structures were filtered to those with a release date after 30 April 2018 (the date limit for inclusion in the training set). Chains were then further filtered to remove sequences that consisted of a single amino acid, sequences with an ambiguous chemical component at any residue position and sequences without a PDB 40% sequence clustering. Exact duplicates were removed by choosing the chain with the most resolved Cα atoms as the representative sequence. Then, structures with fewer than 16 resolved residues, with unknown residues and structures solved by NMR methods were filtered out. Structure prediction then followed the same procedure as for the human proteome with the same length and residue limits, except that templates with a release date after 30 April 2018 were disallowed. Finally, the dataset was redundancy reduced, by taking the chain with the best non-zero resolution from each cluster in the PDB 40% sequence clustering, producing a dataset of 12,494 chains. This is referred to as the recent PDB dataset.

Computational resources

Inference was run on V100 graphics processing units (GPUs), with each sequence inferenced five times to produce five inputs to model selection. To prevent out-of-memory errors, long sequences were assigned to multi-GPU workers. Specifically, sequences of length 1,401–2,000 residues were processed by workers with two GPUs, and those of length 2,001–2,700 residues by workers with four GPUs (further details of unified memory on longer proteins are provided in the companion paper[2]; it is possible higher memory workers could be used without additional GPUs).

The total resources used for inference were logged and amounted to 930 GPU days. This accounts for generating five models per protein; around 190 GPU days would be sufficient to inference each protein once. Long sequences had a disproportionate effect owing to the multi-GPU workers described above. Approximately 250 GPU days would have been sufficient to produce five models for all proteins shorter than 1,400 residues. For reference, Extended Data Fig. 6 shows the relationship between sequence length and inference time.

All other stages of the pipeline (MSA search, template search and constrained relaxation) ran on the central processing unit (CPU) and used standard tools. Our human proteome run made use of some cached intermediates (for example, stored MSA search results). However, we estimate the total cost of running these stages from scratch at 510 core days. This estimate is based on taking a sample of 240 human proteins stratified by length, timing each stage when run with empty caches, fitting a quadratic relationship between sequence length and run time, then applying that relationship to the sequences in the human proteome. Extended Data Figure 7 shows the data used to make this estimate.

Template coverage

Except where otherwise noted, template coverage was estimated on a per-residue basis as follows. Hmmsearch was run against a copy of the PDB SEQRES (downloaded on 15 February 2021) using default flags[67]. The prior template coverage at residue i is the maximum percentage sequence identity of all hits covering residue i, regardless of whether the hit residue is experimentally resolved. For the recent PDB analysis, only template hits corresponding to a structure released before 30 April 2018 were accepted.

In the section on full chain prediction, template filtering is based on the highest sequence identity of any single Hmmsearch hit with more than 50% coverage. This is because high-coverage templates are particularly relevant when considering whether a predicted domain packing is novel.

GO term breakdown

GO annotations were taken from the XML metadata for the UniProt human reference proteome and were matched to the Gene Ontology in obo format[36,37]. One erroneous is_a relationship was manually removed (GO:0071702 is_a GO:0006820, see change log https://www.ebi.ac.uk/QuickGO/term/GO:0071702). The ontology file was used to propagate the GO annotations using is_a and part_of relations to assign parent–child relationships, and accounting for alternative IDs.

GO terms were then filtered to a manageable number for display, first by filtering for terms with more than 3,000 annotations, and from those selecting only moderately specific terms (a term cannot have a child with more than 3,000 annotations). The remaining terms in the 'molecular function' and 'cellular component' ontologies are shown in Fig. 1d.

Structure analysis

Structure images were created in PyMOL[69], and PyMOL align was used to compute r.m.s.d.s (outlier rejection is described in the text where applicable).

For docking against DGAT2, P2Rank[65] was used to identify ligand-binding pockets in the AlphaFold structure. AutoDockTools[70] was used to convert the AlphaFold prediction to PDBQT format. For the ligands, DGAT2-specific inhibitor (CAS number 1469284-79-4) and DGAT1-specific inhibitor (CAS number 942999-61-3) were also prepared in PDBQT format using AutoDockTools. AutoDock Vina[71] was run with an exhaustiveness parameter of 32, a seed of 0 and a docking search space of $25 \times 25 \times 25$ Å^3 centred at the point identified by P2Rank.

For identifying the most similar structure to wolframin, TM-align[42] was used to compare against all PDB chains (downloaded 15 February 2021) with our prediction as the reference. This returned 3F1Z with a TM-score of 0.472.

Additional metrics

The implementation of pTM is described in supplementary information section 1.9.7 of the companion AlphaFold paper[2] and the implementation of the experimentally resolved head is described in supplementary information section 1.9.10 of the companion AlphaFold paper[2]. The weighted version of pTM is described in Supplementary Methods 1.

Analysis of low-confidence regions

For evaluation on CAID, the target sequences and ground-truth labels for the Disprot-PDB dataset were downloaded from https://idpcentral.

org/. Structure prediction was performed as described above for the recent PDB dataset, with a template cut-off of 30 April 2018. To enable complete coverage, two sequences containing non-standard residues (X, U) had these remapped to G (glycine). Sequences longer than 2,000 residues were split into two segments: 1–2,000 and 2,000–end, and the pLDDT and experimentally resolved head arrays were concatenated for evaluation. The two evaluated disorder predictors were taken to be 1 −0.01 × pLDDT and 1 − predicted resolvability for Cα atoms.

To obtain the ratio of heterotypic contacts to all contacts (Fig. 4d), two residues are considered in contact if their Cβ atoms (or Cα for glycine) are within 8 Å and if they are separated in primary sequence by at least three other residues (to exclude contacts within an α-helix). Heteromers are identified as protein entities with a different entity_id in the structure mmCIF file.

Comparison with BestSingleStructuralTemplate

CAMEO data for the period 21 March 2020 to 13 March 2021 were downloaded from the CAMEO website. AlphaFold predictions were produced for all sequences in the target.fasta files, using the same procedure detailed above but with a maximum template date of 1 March 2020. Predictions were scored against the CAMEO ground truth using lDDT-Cα. For BestSingleStructuralTemplate, lDDT-Cα scores were taken from the CAMEO JavaScript Object Notation (JSON) files provided. Structures solved by solution NMR and solid-state NMR were filtered out at the analysis stage. To determine the template identity, templates were drawn from a copy of the PDB downloaded on 15 February 2021 with a template search performed using Hmmsearch. Templates were filtered to those with at least 70% coverage of the sequence and a release date before the query. The template with the highest *e*-value after filtering was used to compute the template identity. Targets were binned according to template identity, with width 10 bins ranging from 30 to 90. Extended Data Figure 4 shows the distribution of lDDT-Cα for each model within each bin as a box plot (horizontal line at the median, box spanning from the lower to the upper quartile, whiskers extending to the minimum and maximum. In total 428 targets were included in the analysis.

Reporting summary

Further information on research design is available in the Nature Research Reporting Summary linked to this paper.

Data availability

Structure predictions by AlphaFold for the human proteome are available under a CC-BY-4.0 license at https://alphafold.ebi.ac.uk/. All input data are freely available from public sources. The human reference proteome together with its XML annotations was obtained from UniProt v.2021_02 (https://ftp.ebi.ac.uk/pub/databases/uniprot/previous_releases/release-2021_02/knowledgebase/). At prediction time, MSA search was performed against UniRef90 v.2020_03 (https://ftp.ebi.ac.uk/pub/databases/uniprot/previous_releases/release-2020_03/uniref/), MGnify clusters v.2018_12 (https://ftp.ebi.ac.uk/pub/databases/metagenomics/peptide_database/2018_12/) and a reduced version of BFD (produced as outlined in the Methods using the BFD (https://bfd.mmseqs.com/)). Template structures, the SEQRES fasta file and the 40% sequence clustering were taken from a copy of the PDB downloaded on 15 February 2021 (https://www.wwpdb.org/ftp/pdb-ftp-sites; see also https://ftp.wwpdb.org/pub/pdb/derived_data/ and https://cdn.rcsb.org/resources/sequence/clusters/bc-40.out for sequence data). Experimental structures were drawn from the same copy of the PDB; we show structures with accessions 6YJ1[64], 6OFS[43], 1IDQ[46], 1PRT[72], 3F1Z[58], 7KPX[66] and 6VP0[51]. The template search used PDB70, downloaded on 10 February 2021 (http://wwwuser.gwdg.de/~compbiol/data/hhsuite/databases/hhsuite_dbs/). The CAID dataset was downloaded from https://idpcentral.org/caid/data/1/reference/disprot-disorder-pdb-atleast.txt. CAMEO data was accessed on 17 March 2021 at https://www.cameo3d.org/static/downloads/modeling/1-year/raw_targets-1-year.public.tar.gz. A copy of the current Gene Ontology database was downloaded on 29 April 2021 from http://current.geneontology.org/ontology/go.obo. Source data are provided with this paper.

Code availability

Source code for the AlphaFold model, trained weights and an inference script are available under an open-source license at https://github.com/deepmind/alphafold. Neural networks were developed with TensorFlow v.1 (https://github.com/tensorflow/tensorflow), Sonnet v.1 (https://github.com/deepmind/sonnet), JAX v.0.1.69 (https://github.com/google/jax/) and Haiku v.0.0.4 (https://github.com/deepmind/dm-haiku).

For MSA search on UniRef90, MGnify clusters and the reduced BFD, we used jackhmmer and for the template search on the PDB SEQRES we used hmmsearch, both from HMMER v.3.3 (http://eddylab.org/software/hmmer/). For the template search against PDB70, we used HHsearch from HH-suite v.3.0-beta.3 14/07/2017 (https://github.com/soedinglab/hh-suite). For constrained relaxation of structures, we used OpenMM v.7.3.1 (https://github.com/openmm/openmm) with the Amber99sb force field.

Docking analysis on DGAT used P2Rank v.2.1 (https://github.com/rdk/p2rank), MGLTools v.1.5.6 (https://ccsb.scripps.edu/mgltools/) and AutoDockVina v.1.1.2 (http://vina.scripps.edu/download/) on a workstation running Debian GNU/Linux rodete 5.10.40-1rodete1-amd64 x86_64.

Data analysis used Python v.3.6 (https://www.python.org/), NumPy v.1.16.4 (https://github.com/numpy/numpy), SciPy v.1.2.1 (https://www.scipy.org/), seaborn v.0.11.1 (https://github.com/mwaskom/seaborn), scikit-learn v.0.24.0 (https://github.com/scikit-learn/), Matplotlib v.3.3.4 (https://github.com/matplotlib/matplotlib), pandas v.1.1.5 (https://github.com/pandas-dev/pandas) and Colab (https://research.google.com/colaboratory). TM-align v.20190822 (https://zhanglab.dcmb.med.umich.edu/TM-align) was used for computing TM-scores. Structure analysis used Pymol v.2.3.0 (https://github.com/schrodinger/pymol-open-source).

67. Eddy, S. R. A new generation of homology search tools based on probabilistic inference. *Genome Inform.* **23**, 205–211 (2009).
68. Steinegger, M., Mirdita, M. & Söding, J. Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold. *Nat. Methods* **16**, 603–606 (2019).
69. Schrödinger. The PyMOL Molecular Graphics System v.1.8 (2015).
70. Morris, G. M. et al. AutoDock4 and AutoDockTools4: automated docking with selective receptor flexibility. *J. Comput. Chem.* **30**, 2785–2791 (2009).
71. Trott, O. & Olson, A. J. AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *J. Comput. Chem.* **31**, 455–461 (2010).
72. Stein, P. E. et al. The crystal structure of pertussis toxin. *Structure* **2**, 45–57 (1994).
73. Necci, M., Piovesan, D., Clementel, D., Dosztányi, Z. & Tosatto, S. C. E. MobiDB-lite 3.0: fast consensus annotation of intrinsic disorder flavours in proteins. *Bioinformatics* **36**, 5533–5534 (2020).
74. Dyson, H. J. Roles of intrinsic disorder in protein–nucleic acid interactions. *Mol. Biosyst.* **8**, 97–104 (2012).
75. Dunbrack, R. L. Jr & Karplus, M. Backbone-dependent rotamer library for proteins. Application to side-chain prediction. *J. Mol. Biol.* **230**, 543–574 (1993).

Acknowledgements We thank A. Paterson, C. Low, C. Donner, D. Evans, F. Yang, J. Stanway, J. Stanton, L. Deason, N. Latysheva, N. Hobbs, R. Hadsell, R. Green, S. Brown, V. Bolina, Ž. Avsec and the Research Platform Team for their contributions; R. Kemp for help in managing the project and our colleagues at DeepMind, Google and Alphabet for their encouragement and support; E. van Schaftingen, M. Zhou and F. Urano for reading and commenting on our discussion of glucose-6-phosphatase, diacylglycerol *O*-acyltransferase 2 and wolframin, respectively; the team at EMBL-EBI for their work on making AlphaFold structure predictions available, in particular M. Varadi, M. Deshpande, S. Sasidharan Nair, S. Anyango, G. Yordanova, C. Natassia, D. Yuan and E. Heard.

Author contributions K.T., J.J. and D.H. led the research. D.H., K.K., P.K., C.M. and E.C. managed the research. T.G. developed the proteome-scale inference system. K.T., J.A., Z.W., M.Z., R.E., M.F., A. Bridgland and A.C. generated and analysed the structure predictions. J.J., M.F., S.A.A.K.

and O.R. developed the metrics used to interpret predictions. A.Ž., S.P., T.G., A.C. and K.T. developed the data-processing pipelines to produce the AlphaFold protein structure database. S.V., A.L., A. Bateman, G.J.K., D.H. and E.B. managed the work to make AlphaFold predictions available via EMBL-EBI-hosted resources. S.V., G.J.K. and A. Bateman provided scientific advice on how predictions should be displayed. J.J., R.E., A. Pritzel, M.F., O.R., R.B., A. Potapenko, S.A.A.K., B.R.-P., J.A., A.W.S., T.G., A.Ž., K.T., A. Bridgland, A.J.B., A.C., S.N., R.J., D.R. and M.Z. developed the network and associated infrastructure used in inferencing the proteome. K.T., J.A., Z.W., J.J., M.F., M.Z., C.M. and D.H. wrote the paper.

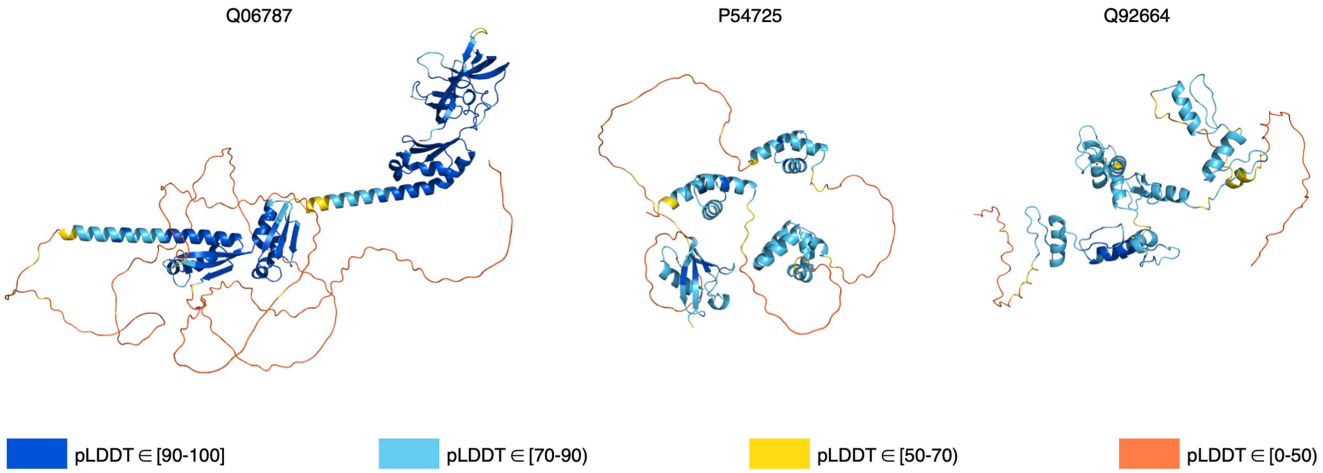
Competing interests J.J., R.E., A. Pritzel, T.G., M.F., O.R., R.B., A. Bridgland, S.A.A.K., D.R. and A.W.S. have filed non-provisional patent applications 16/701,070, PCT/EP2020/084238, and provisional patent applications 63/107,362, 63/118,917, 63/118,918, 63/118,921 and 63/118,919, each in the name of DeepMind Technologies Limited, each pending, relating to machine learning for predicting protein structures. E.B. is a paid consultant to Oxford Nanopore and Dovetail Inc, which are genomics companies. The other authors declare no competing interests.

Additional information

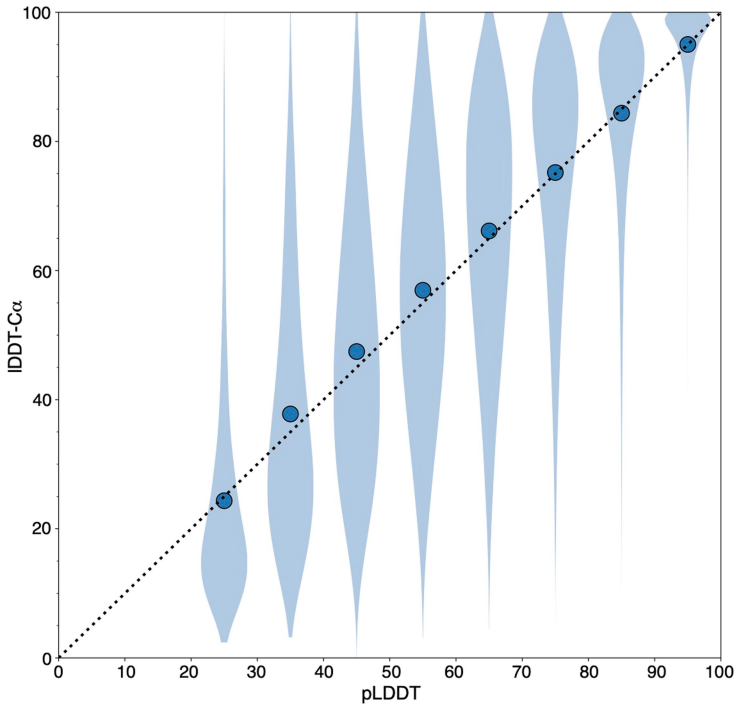
Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41586-021-03828-1.

Correspondence and requests for materials should be addressed to K.T., J.J. or D.H.

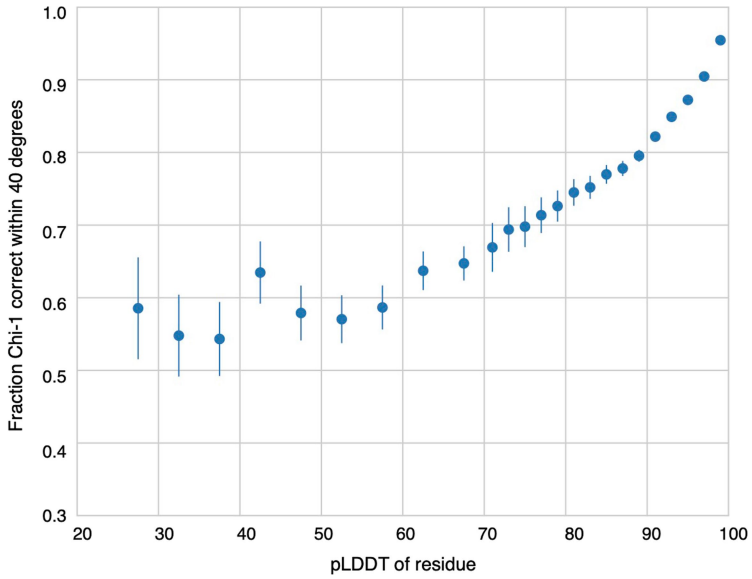
Peer review information *Nature* thanks Mohammed AlQuraishi, Yang Zhang and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.



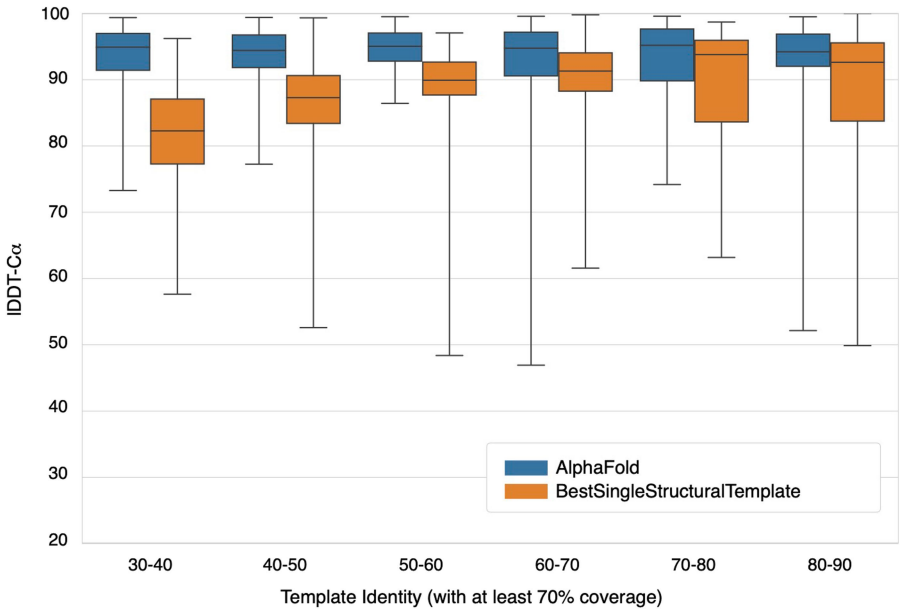
Extended Data Fig. 1 | Example full chain outputs containing both high- and low-confidence regions. Q06787 (synaptic functional regulator FMR1) and P54725 (UV excision repair protein RAD23 homologue A) are predicted to be disordered outside the experimentally determined regions by MobiDB[73]. Q92664 (transcription factor IIIA) has been described as 'beads on a string', consisting of zinc-finger domains joined by flexible linkers[74].



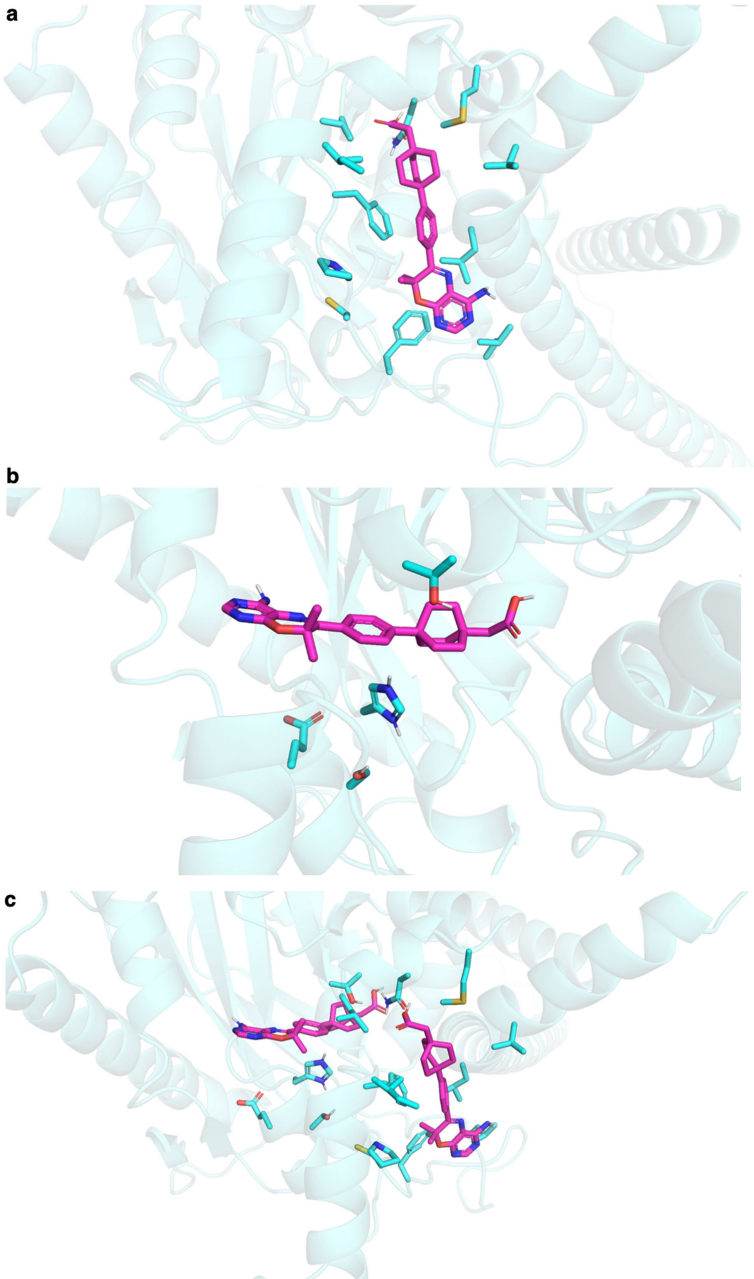
Extended Data Fig. 2 | Distribution of per-residue lDDT-Cα within eight pLDDT bins. This represents an alternative visualization to Fig. 1a that does not sample the data. It uses the recent PDB dataset (Methods), which is restricted to structures with a reported resolution of <3.5 Å (n = 2,756,569 residues). Residues were assigned to bins of width 10 based on their pLDDT (minimum, 20; maximum, 100). Markers show the mean lDDT-Cα within each bin, while the lDDT-Cα distribution is visualized as a Matplotlib violin plot (kernel density estimate bandwidth, 0.2). The smallest sample size for the corresponding violin is 5,655 residues for the left-most bin.



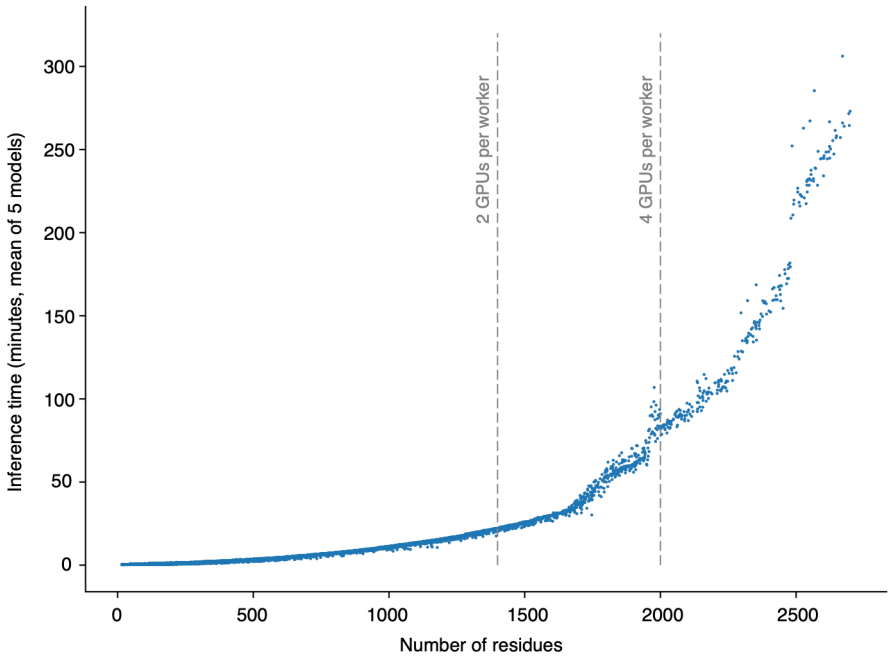
Extended Data Fig. 3 | Relationship between pLDDT and side-chain χ_1 correctness. Evaluated on the recent PDB dataset (Methods), which is restricted to structures with a reported resolution of <2.5 Å (n = 5,983 chains) and residues with a B-factor of <30 Å^2 (n = 609,623 residues). Residues are binned by pLDDT, with bin width 5 between 20 and 70 pLDDT and bin width 2 above 70 pLDDT. A χ_1 angle is considered correct if it is within 40° of its value in the PDB structure[75]. Markers show the proportion of correct χ_1 angles within each bin; error bars indicate the 95% confidence interval (two-sided Student's t-test). The smallest sample size for the error bars is 193 residues for the left-most bin.



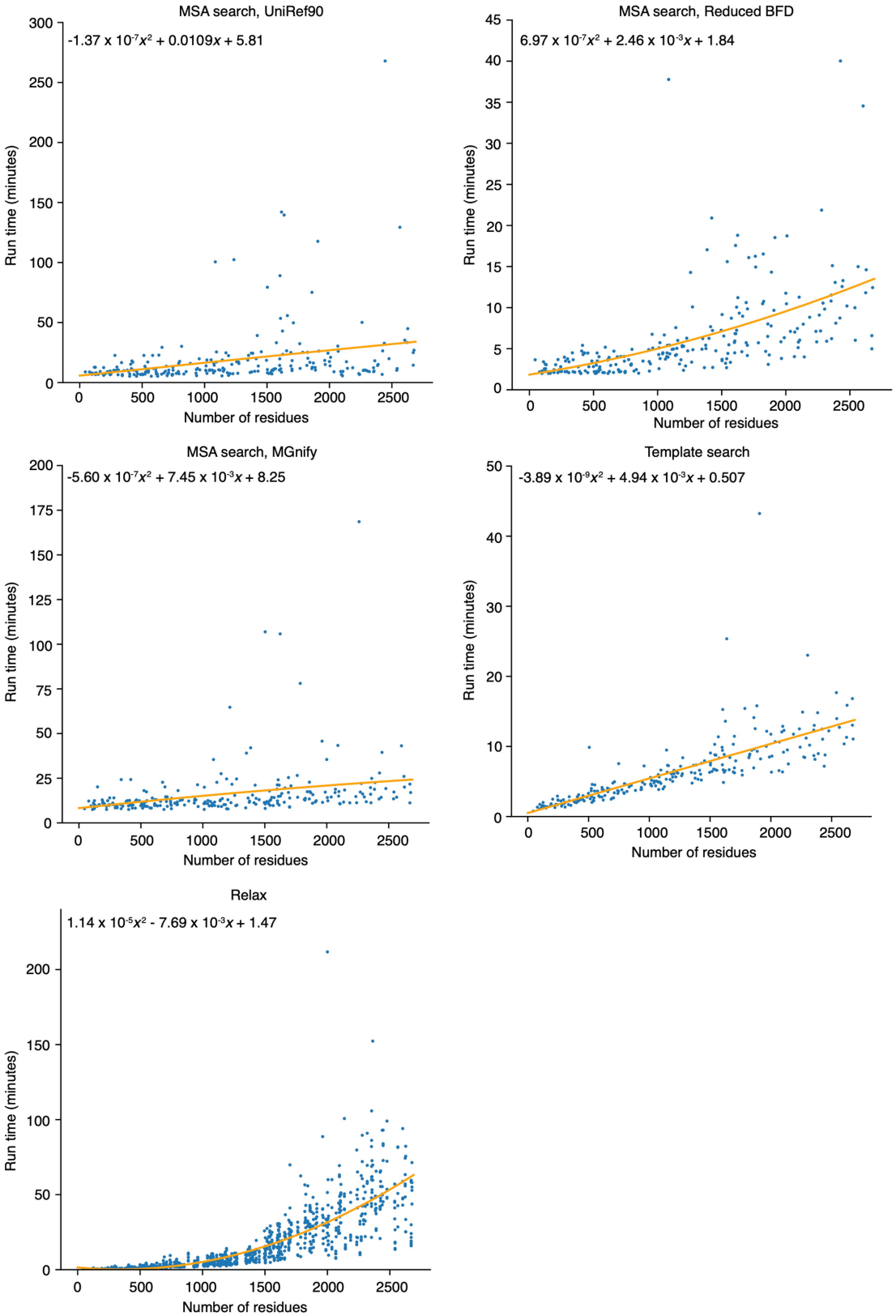
Extended Data Fig. 4 | AlphaFold performance at a range of template sequence identities. lDDT-Cα for AlphaFold and BestSingleStructuralTemplate on 1 year of CAMEO targets[39]. Targets are binned according to the sequence identity of the best template covering at least 70% of the target, and a box plot is shown for each bin. The horizontal line indicates the median, boxes range from the lower to the upper quartile, and the whiskers extend from the minimum to the maximum. In total, 428 targets are included (see Source Data); the smallest number of targets in any bin is 18.



Extended Data Fig. 5 | Docking poses for a DGAT1-specific inhibitor in DGAT2. a, Top binding pose from Autodock Vina for a DGAT1-specific inhibitor in DGAT2, which does not match the predicted binding pocket for a DGAT2-specific inhibitor. **b**, Next best binding pose, which matches the binding pocket for the DGAT2-specific inhibitor, but does not contain components that satisfy the polar side chains His163 and Thr194. **c**, Relative positions of both binding poses.



Extended Data Fig. 6 | Relationship between sequence length and inference time. On the basis of logs from our human proteome set. All of the processed proteins are shown ($n = 20,296$). Each point indicates the mean inference time for the protein over the models produced. Vertical lines show the length cut-offs above which sequences were processed by multi-GPU workers.



Extended Data Fig. 7 | Relationship between sequence length and run time for the non-inference stages of the pipeline. On the basis of 240 human protein sequences, chosen by stratified sampling from the length buckets: [16, 500), [500, 1,000), [1,000, 1,500), [1,500, 2,000), [2,000, 2,500) and [2,500, 2,700]. The relax plot shows five times more points, since five relaxed models are generated per protein. Coefficients for the quadratic lines of best fit were computed with Numpy polyfit.

Extended Data Table 1 | lDDT-Cα distribution in various pLDDT bins

	lDDT-Cα					
	Mean	Median	Q1	Q3	IQR	< lower bin edge - 5
pLDDT in (50-70]	62.5	63.8	48.6	77.8	29.2	20%
pLDDT in (70-90]	82.3	86.5	76.1	92.9	16.7	12%
pLDDT in (90-100]	95.0	97.4	93.8	99.3	5.6	7%

Data are based on the per-residue lDDT-Cα and per-residue pLDDT of resolved regions. This table uses the recent PDB dataset (Methods), which is restricted to structures with a reported resolution of <3.5 Å. The total number of chains included is 10,215.

Extended Data Table 2 | Relationship between pLDDT and TM-score

	TM-score					
	Mean	Median	Q1	Q3	IQR	≥ 0.5
pLDDT in (50-70]	0.44	0.43	0.29	0.58	0.29	37%
pLDDT in (70-90]	0.75	0.83	0.63	0.92	0.29	86%
pLDDT in (90-100]	0.93	0.97	0.92	0.98	0.06	99%

Binning is based on the mean pLDDT over each chain, weighted by the output of the experimentally resolved head. This table uses the recent PDB dataset (Methods), which restricted to structures with a reported resolution of <3.5 Å. The total number of chains included is 10,215.