https://doi.org/10.1038/s41587-021-01179-w

Check for updates

Using deep learning to annotate the protein universe

Maxwell L. Bileschi[1] ✉, David Belanger[1], Drew H. Bryant[1], Theo Sanderson[1,2], Brandon Carter[3], D. Sculley[1], Alex Bateman[4], Mark A. DePristo[1,5] and Lucy J. Colwell[1,6] ✉

Understanding the relationship between amino acid sequence and protein function is a long-standing challenge with far-reaching scientific and translational implications. State-of-the-art alignment-based techniques cannot predict function for one-third of microbial protein sequences, hampering our ability to exploit data from diverse organisms. Here, we train deep learning models to accurately predict functional annotations for unaligned amino acid sequences across rigorous benchmark assessments built from the 17,929 families of the protein families database Pfam. The models infer known patterns of evolutionary substitutions and learn representations that accurately cluster sequences from unseen families. Combining deep models with existing methods significantly improves remote homology detection, suggesting that the deep models learn complementary information. This approach extends the coverage of Pfam by >9.5%, exceeding additions made over the last decade, and predicts function for 360 human reference proteome proteins with no previous Pfam annotation. These results suggest that deep learning models will be a core component of future protein annotation tools.

As the cost of DNA sequencing drops and metagenomic sequencing projects flourish, efficient tools that annotate sequences with function are central to our ability to exploit these data and accelerate biotechnology[1,2]. State-of-the-art annotation uses profile hidden Markov models (pHMMs) built from hand-crafted sequence alignments and scoring functions[3–6] or algorithms such as BLASTp that use pairwise alignment across large sets of labeled sequences[7]. While these approaches are highly successful, the widely used protein families database Pfam has grown by less than 5% over the last 5 years, and at least one-third of microbial proteins cannot be annotated through alignment to functionally characterized sequences[8,9]. Pfam provides small, manually curated seed sets of protein domain sequences (Supplementary Fig. 1). These sets are used to build >17,000 HMM profiles that provide functional annotations for 77.2% of UniProtKB sequences in Pfam. Deep learning presents an opportunity to build on this expertise by learning a model that shares information across families and directly predicts annotation from unaligned sequence. Recent work reports deep models trained on large sets of protein sequences that predict Gene Ontology (GO) terms or Enzyme Commission (EC) numbers[10–19] and use the resulting learned representations to build exciting exploratory tools[17,20–24]. However, many Pfam family seeds contain relatively few sequences, presenting a challenge for data-hungry deep learning approaches. Moreover, it is not yet clear how deep learning models compare to existing state-of-the-art methods or whether they can extend our understanding of the relationship between protein sequence and function.

In this paper, we ask whether deep learning models can complement existing approaches and provide protein function prediction tools with broad coverage of the protein universe. We compare deep learning and existing approaches on the task of annotating unaligned protein domain sequences from Pfam-seed v.32.0, which includes 17,929 families, many of which have very few sequences[25]. For protein sequences, similarities between the test and train data mean that it is essential to stratify model performance as a function of the similarity between each held-out test sequence and the nearest sequence in the train set. We analyze both random and clustered splits in which sequences are assigned to the test or train split using similarity-based cluster membership[26]. We find that the deep models make fewer errors at annotating held-out test sequences than current approaches across both the random and clustered splits. To confirm that the model has captured the structure of unaligned protein sequences, we use the joint representation learned across protein families in one-shot learning to annotate sequences from small families that the model was not trained on. Classifying Pfam seed sequences presents a substantial challenge, and these findings demonstrate that deep learning models can accurately annotate protein domains with their functions.

Results

We used expertly curated seed sequences from the 17,929 families of Pfam v.32.0 to construct benchmark annotation tasks, where unaligned seed sequences from each family are split into train and test sets (1) randomly and (2) by clustering based on sequence identity (Methods). For the random split, 81.2% of sequences are used for training, 9.4% are used for validation and 9.4% are held out as unseen test sequences for all models (available for download at https://console.cloud.google.com/storage/browser/brain-genomics-public/research/proteins/pfam/random_split and https://console.cloud.google.com/storage/browser/brain-genomics-public/research/proteins/pfam/clustered_split, and the interactive notebook is located at https://www.kaggle.com/googleai/pfam-seed-random-split). We trained a neural network (ProtCNN) to classify held-out test sequences by Pfam family and show that it outperforms existing methods, displaying superior accuracy despite not using sequence alignment. On the random split, ProtCNN significantly outperforms the alignment-based methods Top Pick HMM (TPHMM), BLASTp and phmmer for sequences with 30–90% maximum identity to the

[1]Google Research, Cambridge, MA, USA. [2]The Francis Crick Institute, London, UK. [3]MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, USA. [4]European Molecular Biology Laboratory, European Bioinformatics Institute (EMBL-EBI), Hinxton, UK. [5]BigHat Biosciences, San Mateo, CA, USA. [6]Department of Chemistry, University of Cambridge, Cambridge, UK. ✉e-mail: mlbileschi@google.com; lcolwell@google.com

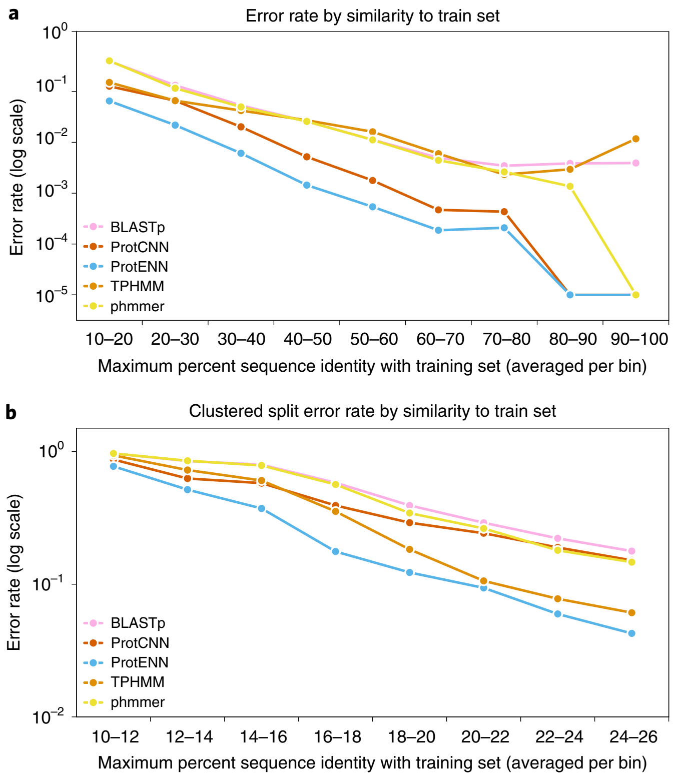


Fig. 1 | Model performance on Pfam-seed. Held-out test error rate as a function of max percent sequence identity of each test sequence with training sequences in the same Pfam family (Methods). Data are binned by percent sequence identity; the *x* axis labels describe the bin ranges. **a**, For the random split, ProtCNN makes significantly fewer errors than alignment-based methods for sequence identities in the range of 30–90% (two-sided McNemar test, 10–20%: 292 sequences, $P = 0.000070$; 20–30%: 3,628 sequences, $P < 10 \times 10^{-6}$; 30–40%: 9,537 sequences, $P < 10 \times 10^{-6}$; 40–50%: 16,798 sequences, $P < 10 \times 10^{-6}$; 50–60%: 22,662 sequences, $P < 10 \times 10^{-6}$; 60–70%: 28,277 sequences, $P < 10 \times 10^{-6}$; 70–80%: 40,221 sequences, $P < 10 \times 10^{-6}$; 80–90%: 4,429 sequences, $P = 0.000244$), while ProtENN is significantly better than alignment-based methods for sequence identities less than 90% (two-sided McNemar test, 10–20%: 292 sequences, $P = 0.000070$; 20–30%: 3,628 sequences, $P < 10 \times 10^{-6}$; 30–40%: 9,537 sequences, $P < 10 \times 10^{-6}$; 40–50%: 16,798 sequences, $P < 10 \times 10^{-6}$; 50–60%: 22,662 sequences, $P < 10 \times 10^{-6}$; 60–70%: 28,277 sequences, $P < 10 \times 10^{-6}$; 70–80%: 40,221 sequences, $P < 10 \times 10^{-6}$; 80–90%: 4,429 sequences, $P = 0.000244$). **b**, For the clustered split, where all sequence identities are ≤25%, ProtENN is significantly more accurate for all bins (two-sided McNemar test, 10–12%: 62 sequences, $P = 0.006348$; 12–14%: 426 sequences, $P < 10 \times 10^{-6}$; 14–16%: 1,058 sequences, $P < 10 \times 10^{-6}$; 16–18%: 2,516 sequences, $P < 10 \times 10^{-6}$; 18–20%: 4,419 sequences, $P < 10 \times 10^{-6}$; 20–22%: 6,013 sequences, $P = 0.011417$; 22–24%: 4,892 sequences, $P = 0.000120$; 24–25%: 1,902 sequences, $P = 0.005172$).

training set (Fig. 1a). ProtENN, a simple majority vote across an ensemble of 19 ProtCNN models, reduces the error rate to 0.16% or 205 misclassifications of 126,171 test sequences, with significantly better accuracy for sequences <90% identical to the training set (Fig. 1a, Supplementary Fig. 2 and Table 1) and notable improvement at annotating short sequences (Supplementary Fig. 3). We found that 11 sequences were classified incorrectly in the same way by every ProtENN ensemble element, and our analysis suggested that there may be some ambiguity over the correct family label in each case. For example, the sequence R7EGP4_9BACE/13-173 has

Table 1 | Model performance on the random split of Pfam-seed

Model	Error rate	Number of errors
TPHMM	1.414%	1,784
phmmer	1.531%	1,932
BLASTp	1.654%	2,087
k-mer	9.994%	12,610
ProtCNN	0.495%	625
ProtENN	**0.162%**	**205**

Performance is evaluated by the number of errors made when using each approach to classify the 126,171 protein domain sequences contained in the held-out random test set (available for download at https://console.cloud.google.com/storage/browser/brain-genomics-public/research/proteins/pfam/random_split). The model with the fewest errors is indicated in bold.

a 100% identity match to a UniProtKB sequence annotated with the YIP family in addition to a GO term GO:0016021 (integral component of membrane), which matches that of the YIP family. After further curation, Pfam family DUF1282 and the YIP1 family have now been merged in Pfam.

Expanding the set of annotated protein sequences requires remote homology detection, that is, accurate classification of sequences that have low similarity to the training data. To measure performance at this task, we used single-linkage clustering at 25% identity within each family to build a clustered split of the seed sequences, inspired by ref. [26]. The resulting benchmark has a distant held-out test set of 21,293 sequences (Methods, Supplementary Table 4 and Supplementary Fig. 4). Table 2 and Fig. 1b show that ProtENN is significantly more accurate for all bins including those with distant test sequences, a key requirement to expand coverage of the protein universe. Stratifying accuracy by the number of training sequences per family, Supplementary Fig. 5 shows that ProtENN outperforms alignment-based methods in all cases except the smallest families of the clustered split. To address this challenge of extrapolating from few examples, we used the sequence representation learned by the deep model (Fig. 2) to improve performance.

ProtCNN learns a length 1,100 real-valued vector representation of each sequence, regardless of its unaligned length[27]. For high accuracy, representations from each family must cluster tightly such that different families are well separated from each other. To test whether this learned representation could be used to accurately classify sequences from the smallest families, we built a new approach called ProtREP. For ProtREP, we compute the average learned representation for each family across its training sequences, yielding a sentinel family representation. We then classify each held-out test sequence by finding its nearest sentinel in the space of learned representations. For the same computational cost, ProtREP surpasses the accuracy of ProtCNN on the clustered split (Fig. 3a and Supplementary Fig. 6).

Over the last several years, increases in Pfam coverage have been driven by the identification of new families[6], which involves (1) determining that a domain does not belong to an existing family and (2) identifying additional family members[7,28]. To investigate whether ProtREP can build new families, we split the families, training the underlying neural network on only the 12,361 families with >9 random split train sequences. We used the resulting model to embed one training sequence from each of the unseen 5,568 small families (Methods). We then used ProtREP to classify all 126,171 held-out test sequences of the random split, achieving an overall accuracy of 99.21% and 85.1% on the 710 test sequences from the 5,568 unseen small families. This demonstrates that ProtREP can accurately grow a new family from a single example. Supplementary Table 9 shows that ProtREP accuracy on unseen families improves rapidly as additional founder sequences are added. This suggests that ProtREP will excel at an iterative approach to family discovery,

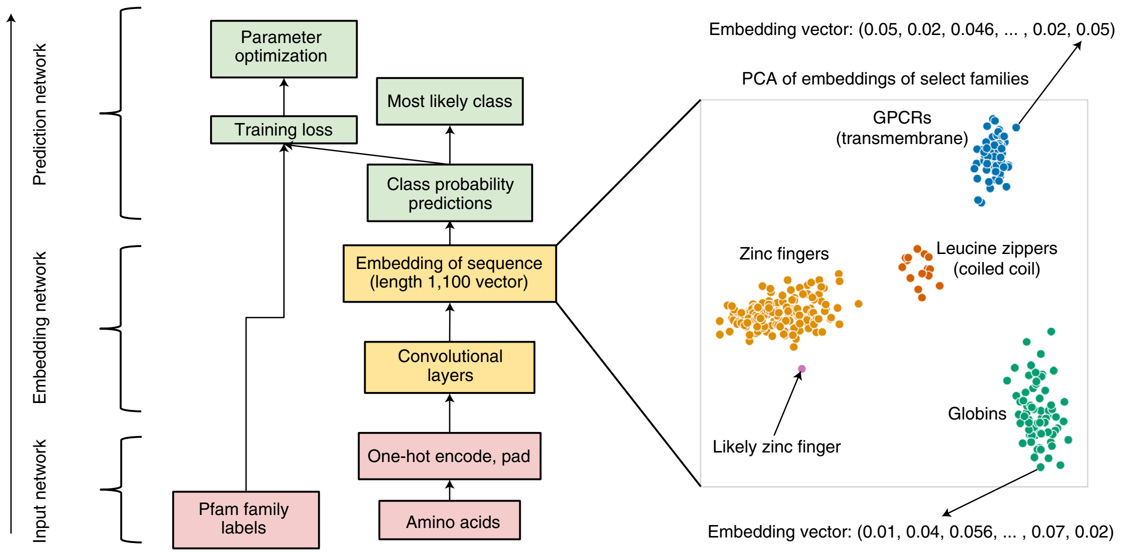

Prediction network
Embedding network
Input network
Parameter optimization
Training loss
Most likely class
Class probability predictions
Embedding of sequence (length 1,100 vector)
Convolutional layers
One-hot encode, pad
Amino acids
Pfam family labels
Embedding vector: (0.05, 0.02, 0.046, ... , 0.02, 0.05)
PCA of embeddings of select families
GPCRs (transmembrane)
Zinc fingers
Leucine zippers (coiled coil)
Likely zinc finger
Globins
Embedding vector: (0.01, 0.04, 0.056, ... , 0.07, 0.02)

Fig. 2 | ProtCNN architecture. The central graph illustrates the input (red), embedding (yellow) and prediction (green) networks together with the residual network (ResNet) architecture (left), while the right illustrates the representation of sequence space learned by ProtCNN and exploited by ProtREP through simple nearest neighbor approaches. In this representation, each sequence corresponds to a single point, and sequences from the same family are typically closer to each other than to sequences from other families; GPCR, G-protein-coupled receptor; PCA, principal-component analysis.

where a single founder sequence is gradually incremented by homologs identified by the model.

To probe what the deep models learn about protein sequence data, we trained ProtCNN on 80% of unaligned sequences from Pfam-full (Supplementary Fig. 7) and computed a similarity matrix of learned amino acid representations. Fig. 3b shows that this matrix closely mirrors the structure of the BLOSUM62 substitution matrix, which was derived by experts from carefully curated sequence alignments[29]. To measure the predicted impact of single mutations, we calculated the Kullback–Leibler divergence between the ProtCNN predictions for the original and mutated sequences (Supplementary Fig. 8). Mutations in disordered regions are predicted to have negligible effect, except for mutations to phenylalanine, tyrosine and tryptophan, which promote order due to aromatic ring stacking[30]. Similarly, ProtCNN amino acid preference within transmembrane helices agrees with prior knowledge[31].

These results demonstrate that ProtCNN learns a meaningful representation of protein sequence that (1) generalizes to unseen parts of sequence space and (2) can be used to predict and understand the properties of protein sequences. A further challenge is to detect a protein domain and its location within a protein sequence. This task is analogous to image segmentation, a task at which deep learning models excel. While ProtCNN was trained using domains, Supplementary Table 12 shows ProtCNN segmentation of full sequences into domains using a simple sliding window approach (Methods).

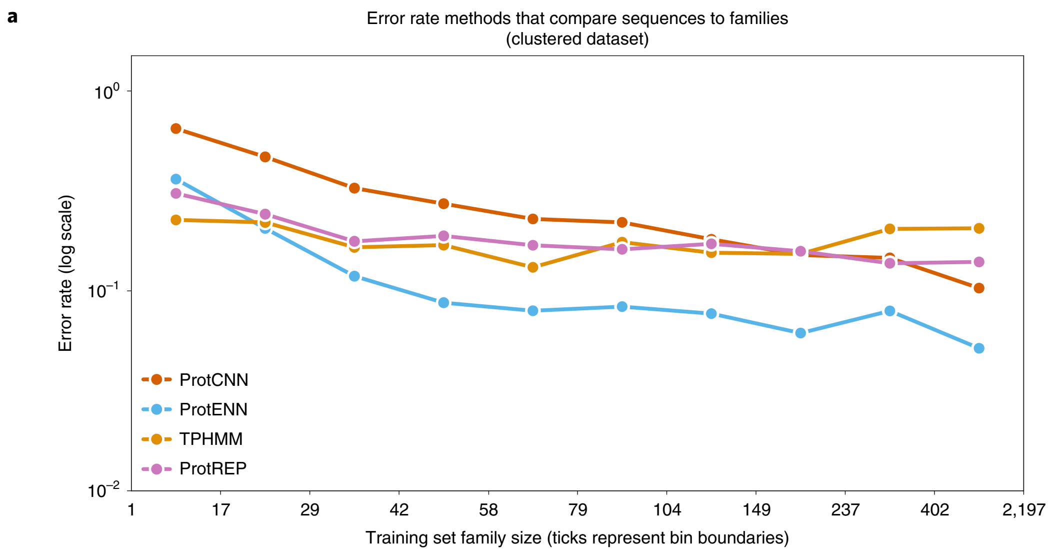
Deep models have the capacity to capture non-local sequence information and transfer information across protein families, meaning they can learn complementary information to the pHMMs used to build Pfam-full. This suggests that combining the two approaches could expand Pfam coverage of the protein universe. To test this idea on the clustered benchmark, we built a simple ensemble: for held-out test sequences where HMMER is confident, we retain its prediction, otherwise we consider the ProtENN prediction. Protein domain sequences can match multiple families within each expertly curated clan of related families[32–34]; so to be conservative, we work at the clan level (Supplementary Fig. 9). Figure 4a and Supplementary Figs. 10 and 11 show that the ensemble of ProtENN and HMMER is substantially more accurate than either model at clan-level remote sequence homology detection, reducing the clustered benchmark error rate by 38.6%. By contrast, combining BLAST and HMMER yields no performance gain (Supplementary Fig. 12). These results show that the deep models provide highly complementary information to alignment-based annotation methods.

Table 2 | Model performance on the clustered split of Pfam-seed

Model	Error rate	Number of errors
Top Pick HMM	18.1%	3,844
phmmer	32.6%	6,942
BLASTp	35.9%	7,639
ProtCNN	27.6%	5,882
ProtENN	**12.2%**	**2,590**

Performance is evaluated by the number of errors made when using each approach to classify the 21,293 protein domain sequences contained in the held-out clustered test set (available for download at https://console.cloud.google.com/storage/browser/brain-genomics-public/research/proteins/pfam/clustered_split). The model with the fewest errors is indicated in bold.

This orthogonality suggests that ProtENN can expand Pfam coverage of the protein universe. To test this hypothesis, we first used the clustered benchmark to identify criteria that yield reliable annotations. For held-out sequences from the clustered split, requiring ProtENN ensemble consensus >60% yields 99.9% accuracy, while adding the requirement that TPHMM agrees raises this to 100% (Fig. 4b and Supplementary Fig. 11). These criteria provide effective mechanisms for determining that a sequence domain does not belong to an existing family. To leverage these criteria to expand Pfam coverage, we trained ProtENN on Pfam-full and performed inference for 140 million regions where HMMER confidence was not sufficient for inclusion in Pfam v.34.0. Overall, we find that combining ProtENN + HMM increases Pfam coverage by 6.8 million sequence regions or ~9.5%, comparable to the number of regions added by the HMM-based pipeline over the last decade (Fig. 4c)[35]. These ProtENN-predicted regions provide annotations for ~1.8 million protein sequences with no annotation in Pfam-full, including 360 proteins from the human reference proteome[36]. These predictions are available as Pfam-N, part of the Pfam 34.0 release



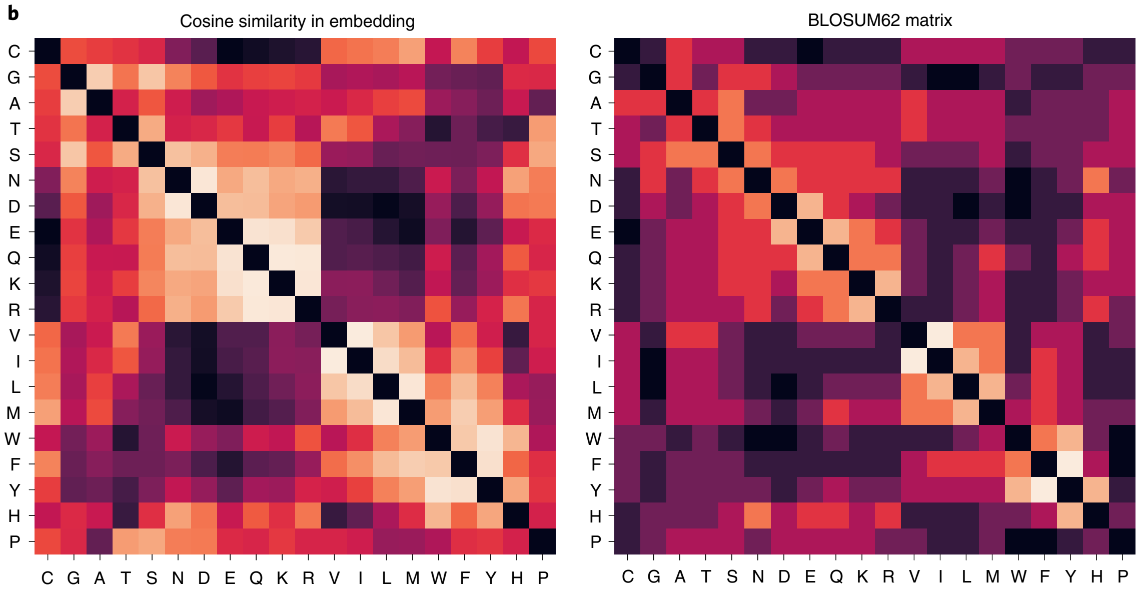


Fig. 3 | Clustered split performance of ProtCNN, ProtENN, TPHMM and ProtREP, which uses the learned representation of sequence space. a, Error rates at classifying held-out test sequences are stratified by the number of training sequences per family. ProtREP has the same computational complexity as ProtCNN but is more accurate for small families. This suggests that the speed–accuracy tradeoff between ProtCNN and ProtENN can be ameliorated to yield classifiers that are both faster than ensembles and as accurate. **b**, The amino acid embedding learned by ProtCNN from unaligned sequence data reflects the overall structure of the BLOSUM62 matrix[29].

(available for download at http://ftp.ebi.ac.uk/pub/databases/Pfam/releases/Pfam34.0/).

Discussion

The ~6.8 million sequence regions for which ProtENN adds Pfam annotations represent a noteworthy expansion of coverage without adding new protein families. We provide functional predictions for 360 human reference proteome proteins that had no previous Pfam annotation. Among these, we highlight our prediction that the protein Q96HJ9, recently renamed FMC1, contains the Pfam Complex1_LYR-like domain. Compellingly, Li et al.[37] showed experimentally that Q96HJ9 is required for mitochondrial ATP synthase function and binds a known assembly factor of the mitochondrial ATP synthase complex, entirely analogous with the role played by the yeast protein Fmc1p, which also contains a LYR protein domain, although homology between the yeast protein Fmc1p and Q96HJ9 cannot be detected by methods such as BLASTp[37].

Our benchmarking of the simple combination of ProtENN + HMMER shows that it is more accurate than either individual approach, suggesting that errors will be less frequent. For example, this method suggests that the protein domain I0I4T0/38-83 be added to the Pfam glutaredoxin family PF00462. The existing Pfam annotation in this region assigns it to the uncharacterized family DUF411, suggesting a potential functional similarity between DUF411 members and families within the thioredoxin clan. Indeed, this is borne out by the observation of the substantial number of overlapping InterProScan calls for DUF411 and thioredoxins (IPR036249); there are 2,865 TREMBL proteins that share these annotations, of which 731 are named 'Uncharacterized protein' in UniProt. Moreover, the uncharacterized CopG protein

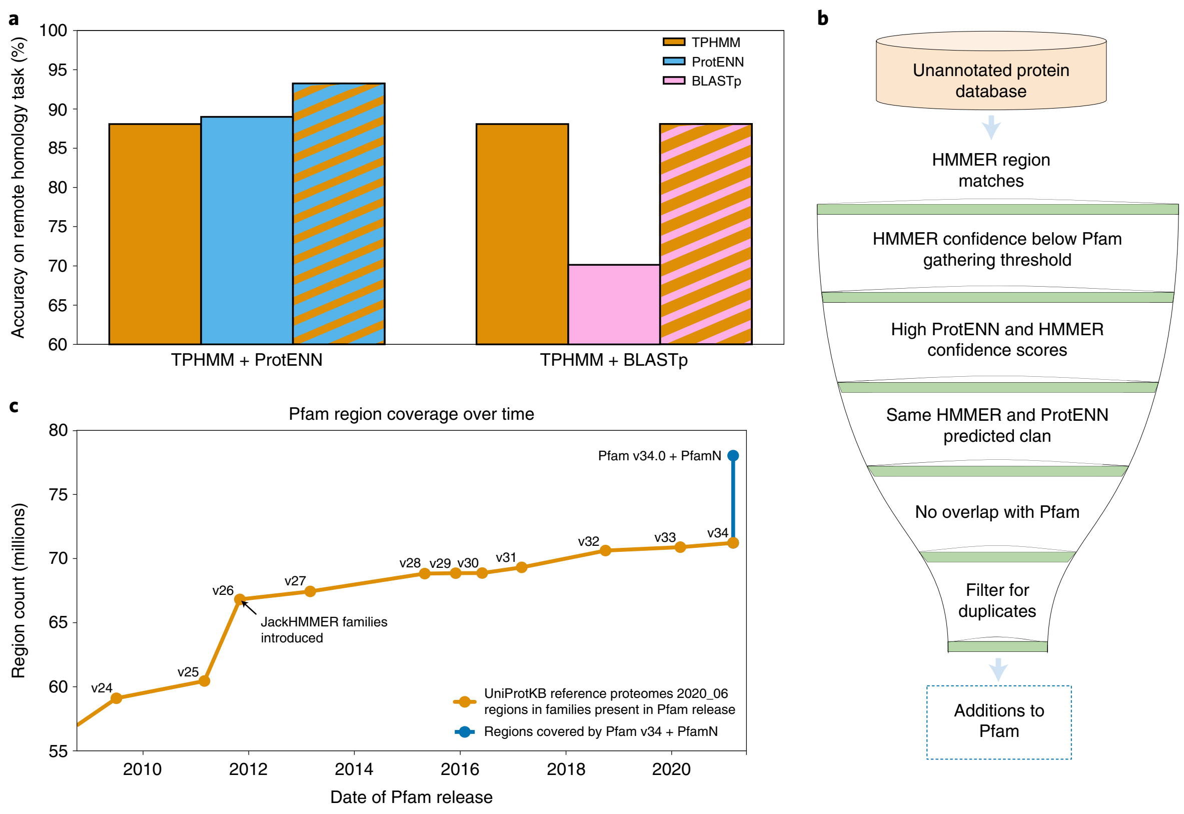


Fig. 4 | A combination of ProtENN and TPHMM improves performance on the remote homology task. a, Model performance comparison for held-out test sequences from the clustered split of Pfam-seed. A simple combination of the TPHMM and ProtENN models reduces the error rate by 38.6%, increasing accuracy from the ProtENN figure of 89.0% to 93.3%. By contrast, combining the BLAST and TPHMM models did not improve performance over the TPHMM performance of 88.1% accuracy. **b**, Illustration of the simple combination of the TPHMM and ProtENN models used to propose new additions to Pfam. Following these steps for the test sequences used in **a** that clear the confidence score thresholds results in an accuracy of 100%. **c**, The ProtENN-proposed increase in Pfam sequence regions in the context of recent Pfam releases.

from *Pseudomonas aeruginosa*, which contains a Pfam DUF411 annotation, was recently extensively characterized through structural and biochemical analysis[38]. The crystal structure reveals that CopG contains a thioredoxin domain modified by a C-terminal extension that contributes to metal binding. The sequence location of the thioredoxin domain is shifted by about 25 residues from the DUF411 Pfam annotation, but the large overlap supports the ProtENN-predicted link between the DUF411 family and the thioredoxin-like clan CL0172.

These examples, drawn from the ~6.8 million new sequence region annotations provided by this work, demonstrate the depth of insights provided by the deep learning models. The ability of ProtREP to improve accuracy without adding computational overhead suggests that model performance can be further improved using techniques from machine learning. For example, the accuracy of ProtREP could be extended by, for example, ensembling embeddings across multiple ProtCNN models or distilling the ensembled CNN models into a single model[39] before moving to embedding space. Using the ProtREP results presented here to discover new families is a natural next step, with the ability to further expand the annotated protein universe. These results present a substantial advance over previous efforts applying deep learning in terms of the number of families, the number of sequences per family and the rigorous comparison with existing methods. The model training protocol we describe can be applied to many sets of protein annotations to rapidly and efficiently annotate unlabeled sequences and unlock molecular diversity for both therapeutic and biotechnology applications.

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/s41587-021-01179-w.

Received: 6 May 2021; Accepted: 2 December 2021;
Published online: 21 February 2022

References

1. Steinegger, M. & Söding, J. Mmseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nat. Biotechnol.* **35**, 1026–1028 (2017).
2. Steinegger, M. & Söding, J. Clustering huge protein sequence sets in linear time. *Nat. Commun.* **9**, 2542 (2018).
3. Söding, J. Protein homology detection by HMM–HMM comparison. *Bioinformatics* **21**, 951–960 (2004).
4. Biegert, A. & Söding, J. Sequence context-specific profiles for homology searching. *Proc. Natl Acad. Sci. USA* **106**, 3770–3775 (2009).
5. Finn, R. D., Clements, J. & Eddy, S. R. HMMER web server: interactive sequence similarity searching. *Nucleic Acids Res.* **39**, W29–W37 (2011).
6. Mistry, J. et al. Pfam: the protein families database in 2021. *Nucleic Acids Res.* **49**, D412–D419 (2021).
7. 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).
8. Price, M. N. et al. Mutant phenotypes for thousands of bacterial genes of unknown function. *Nature* **557**, 503–509 (2018).
9. Chang, Y.-C. et al. COMBREX-DB: an experiment centered database of protein function: knowledge, predictions and knowledge gaps. *Nucleic Acids Res.* **44**, D330–D335 (2015).

10. UniProt Consortium. UniProt: the universal protein knowledgebase. *Nucleic Acids Res.* **45**, D158–D169 (2017).
11. Hou, J., Adhikari, B. & Cheng, J. DeepSF: deep convolutional neural network for mapping protein sequences to folds. *Bioinformatics* **34**, 1295–1303 (2017).
12. Kulmanov, M., Khan, M. A. & Hoehndorf, R. DeepGO: predicting protein functions from sequence and interactions using a deep ontology-aware classifier. *Bioinformatics* **34**, 660–668 (2017).
13. Cao, R. et al. ProLanGO: protein function prediction using neural machine translation based on a recurrent neural network. *Molecules* **22**, 1732 (2017).
14. Li, Y. et al. DEEPre: sequence-based enzyme ec number prediction by deep learning. *Bioinformatics* **34**, 760–769 (2017).
15. Szalkai, B. & Grolmusz, V. Near perfect protein multi-label classification with deep neural networks. *Methods* **132**, 50–56 (2018).
16. Zou, Z., Tian, S., Gao, X. & Li, Y. mlDEEPre: multi-functional enzyme function prediction with hierarchical multi-label deep learning. *Front. Genet.* **9**, 714 (2019).
17. Schwartz, A. S. et al. Deep semantic protein representation for annotation, discovery, and engineering. Preprint at *bioRxiv* https://doi.org/10.1101/365965 (2018).
18. Zhang, D. and Kabuka, M. R. Protein family classification with multi-layer graph convolutional networks. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)* 2390–2393 (IEEE, 2018).
19. Liu, X. Deep recurrent neural network for protein function prediction from sequence. Preprint at https://arxiv.org/abs/1701.08318 (2017).
20. Asgari, E. & Mofrad, M. R. K. Continuous distributed representation of biological sequences for deep proteomics and genomics. *PloS ONE* **10**, e0141287 (2015).
21. Sinai, S., Kelsic, E., Church, G. M. & Nowak, M. A. Variational auto-encoding of protein sequences. Preprint at https://arxiv.org/abs/1712.03346 (2017).
22. 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).
23. 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).
24. Littmann, M., Heinzinger, M., Dallago, C., Olenyi, T. & Rost, B. Embeddings from deep learning transfer GO annotations beyond homology. *Sci. Rep.* **11**, 1160 (2021).
25. El-Gebali, S. et al. The Pfam protein families database in 2019. *Nucleic Acids Res.* **47**, D427–D432 (2018).
26. Eddy, S. R. Accelerated profile HMM searches. *PLoS Comput. Biol.* **7**, e1002195 (2011).
27. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).
28. Johnson, L. S., Eddy, S. R. & Portugaly, E. Hidden Markov model speed heuristic and iterative hmm search procedure. *BMC Bioinformatics* **11**, 431 (2010).
29. Henikoff, S. & Henikoff, J. G. Amino acid substitution matrices from protein blocks. *Proc. Natl Acad. Sci. USA* **89**, 10915–10919 (1992).
30. Campen, A. et al. TOP-IDP-scale: a new amino acid scale measuring propensity for intrinsic disorder. *Protein Pept. Lett.* **15**, 956–963 (2008).
31. Pace, C. N. & Scholtz, J. M. A helix propensity scale based on experimental studies of peptides and proteins. *Biophysical J.* **75**, 422–427 (1998).
32. Finn, R. D. et al. Pfam: clans, web tools and services. *Nucleic Acids Res.* **34**, D247–D251 (2006).
33. Bateman, A. What are these new families with 2, 3, 4 endings? *Xfam Blog* https://xfam.wordpress.com/2012/01/19/what-are-these-new-families-with-_2-_3-_4-endings/ (2012).
34. Finn, R. D. et al. The Pfam protein families database: towards a more sustainable future. *Nucleic Acids Res.* **44**, D279–D285 (2015).
35. Bateman, A. Google research team bring deep learning to Pfam. *Xfam Blog* https://xfam.wordpress.com/2021/03/24/google-research-team-bring-deep-learning-to-pfam/ (2021).
36. UniProt Consortium. UniProt: a hub for protein information. *Nucleic Acids Res.* **43**, D204–D212 (2014).
37. Li, Y., Jourdain, A. A., Calvo, S. E., Liu, J. S. & Mootha, V. K. CLIC, a tool for expanding biological pathways based on co-expression across thousands of datasets. *PLoS Comput. Biol.* **13**, e1005653 (2017).
38. Hausrath, A. C., Ramirez, N. A., Ly, A. T. & McEvoy, M. M. The bacterial copper resistance protein CopG contains a cysteine-bridged tetranuclear copper cluster. *J. Biol. Chem.* **295**, 11364–11376 (2020).
39. Hinton, G., Vinyals, O. & Dean, J. Distilling the knowledge in a neural network. Preprint at https://arxiv.org/abs/1503.02531 (2015).

Methods

Benchmark data. To benchmark performance at unaligned protein domain sequence annotation, we use the highly curated Pfam database[25,40]. The 17,929 classes of Pfam 32.0 provide broad coverage of the known protein universe; 77.2% of the ~137 million sequences in UniProtKB have at least one Pfam family annotation, including 74.5% of proteins from reference proteomes[10,25]. Many domains have functional annotations, although at least 22% of Pfam domains have unknown function[25]. Pfam provides a profile HMM for each family constructed from the seed alignment. These manually curated seed alignments contain between 1 and 4,545 sequences (Supplementary Fig. 1), originally picked due to their trusted functional annotations[40].

Random split. We split each Pfam family with ≥10 seed sequences randomly into disjoint train, dev and test sets. A dev (development) set is a set used to tune hyperparameters that is separate from the test set to avoid overfitting on test data. This results in held-out test sequences for 13,071 families (Supplementary Table 1), where 2,819 families have exactly one sequence in each of the test and dev sets. The other 4,858 families have <10 seed sequences and are only present in the train set. This makes the task harder because there are more ways each test sequence can be misclassified. We provide the split for download (https://console.cloud.google.com/storage/browser/brain-genomics-public/research/proteins/pfam/random_split) and at Kaggle, together with an interactive Jupyter notebook (https://www.kaggle.com/googleai/pfam-seed-random-split).

Clustered split. For every family (no minimum family size), we split the sequences into train, dev and test sequence sets that are distant in sequence space from each other, following the protocol developed in ref. [26]. For each family, we split the sequences using the following steps (see Supplementary Methods for additional details):

1. Construct the matrix of pairwise distances, where distance is measured in terms of sequence identity according to the Pfam-seed family alignment (Supplementary Fig. 1c).
2. Run single-linkage hierarchical agglomerative clustering and process the resulting clustering tree to yield a set of clusters where each element of a given cluster is guaranteed to have at most $\alpha = 0.25\%$ sequence identity with the nearest element of any other cluster.
3. For each family, sort clusters by size, and add clusters to a set of training sequences until its size exceeds a fraction $\Delta = 0.66$ of the overall family size.
4. Split the non-training sequences into dev and test sets using the following steps:

   (a) Recluster each family at a threshold of $\alpha = 0.7$.
   (b) Place clusters into dev using a ratio of $\Delta = 0.5$.

The additional evaluation (dev) set allows model hyperparameters to be tuned without overfitting to the test data, because for each family, these two sets are also distant in sequence space. The number of sequences in the resulting split are shown in Supplementary Table 2.

ProtCNN. We use unaligned sequence data to train ProtCNN, a deep model that predicts which Pfam family each held-out Pfam seed sequence belongs to. ProtCNN processes an input sequence using two consecutive steps: (1) map the sequence to a 1,100-dimensional feature vector (known as an 'embedding') using multiple layers of non-linear transformations, and (2) apply a linear transformation to the embedding to predict family membership. Fig. 2 depicts the input, embedding and prediction networks. Sequences are represented using one-hots and presented to the model in batches, where each sequence is padded to the length of the longest sequence in the batch. All processing in the embedding network is invariant to padding (Supplementary Information). Details about neural network hyperparameters tuned using the dev set and inference speed are provided in Supplementary Tables 13–16.

For the embedding network, ProtCNN uses convolutional ResNets, a variant of convolutional neural networks that train quickly and are stable, even with many layers[41]. Fig. 2 depicts the ResNet architecture, which includes dilated convolutions[42]. The ProtCNN networks are translationally invariant, an advantage for working with unaligned protein sequence data. An n-dilated one-dimensional (1D)-convolution takes standard convolution operations over every nth element in a sequence, allowing local and global information to be combined without greatly increasing the number of model parameters. For our benchmark setup, we find that larger receptive fields (a function of how dilated the convolutions are) generally correspond to higher accuracy (Supplementary Fig. 2). The prediction network maps the output of the embedding network to a distribution over labels using a multiclass logistic regression model. The model prediction is the most likely label under this distribution. At train time, the weights and biases of the model are updated using standard forward and back propagation.

ProtENN. Replicate deep CNN models trained with different random parameter initializations make distinct errors, leading to ProtENN, an ensemble of ProtCNN models. Accuracy as a function of the number of ensemble elements is shown in Supplementary Fig. 2.

phmmer. We take the set of unaligned training sequences as a sequence database, and using 'phmmer' from HMMER 3.1b2 (ref. [26]), we query each test sequence against this database to find the closest match. Those test sequences that return hits above the default phmmer reporting threshold are then annotated with the label of the training sequence hit with the highest bit score. Of the 126,171 sequences in the test set, 42 did not return a hit using this approach. All training sequences that are not reported as hits by the phmmer function are counted as incorrect.

k-mer. An alignment-free approach is provided by a k-mer (or n-gram)-based model, where each sequence is represented by the set of k-mers that it contains. We train a multiclass logistic regression model on vectors of k-mer counts using the same stochastic gradient descent procedure as was used by our deep models (Supplementary Table 18).

BLASTp. BLASTp[43] is one of the most well-known algorithms for searching for similar sequences and is among the current state of the art. BLASTp uses an alignment to rank sequences according to their similarity to a target sequence, and from this, a user can impute functional annotation by ascribing known functions of similar sequences. We use BLASTp as a one-nearest neighbor algorithm by first using 'makeblastdb' (version 2.7.1+) with the training data. We then query sequences from that database using 'blastp -query', taking only the top hit by bit score. This implementation returns no hit for 259 (0.21%) of the 126,171 sequences in the Pfam-seed test set.

TPHMM. We used 'hmmbuild' from HMMER 3.1b2 to construct a pHMM from the aligned train sequences for each family in Pfam 32.0. We implement a simple top pick strategy to avoid any handicap from the filters built into HMMER 3.1b2. We first use 'hmmsearch' to search all 17,929 profiles against each unaligned test sequence and report at least one hit (using '--max' if necessary). We then call the profile with the highest score as the TPHMM prediction (Supplementary Methods). Removing the top pick strategy returns multiple family hits for 8.5% of test sequences from our random benchmark, resulting in a higher error rate. Note that we do not expect TPHMM to achieve 100% accuracy because the training data are a subset of the Pfam-seed alignment. For TPHMM only, we retain the alignment information from the whole Pfam-seed to avoid any artifacts introduced by realignment and enable optimal performance. During training, this provides information about the held-out test sequences used to measure performance, meaning that the reported TPHMM accuracy should be taken as an upper bound. By contrast, all alignment information is removed from the data for the deep learning models and other baselines.

ProtREP. We can use the learned representation to perform nearest neighbor classification for any set of protein sequences. Given a trained ProtCNN model, ProtREP computes an average embedding or learned representation per family over the training sequences and computes a linear whitening transformation to convert their covariance to the identity. This whitening transformation is then applied to the embedding of each test sequence, from which ProtREP classifies test sequences by computing the cosine similarity in embedding space to all train families in a single step. Thus ProtREP has the same computational efficiency as ProtCNN.

Annotation of sequences from families unseen during training. ProtCNN has 0% accuracy at classifying sequences into new families. To test ProtREP's ability at this task, we hold out the smallest families from model training. We use the training sequences from the 12,361 largest random split families to train ProtCNN and compute an average representation for each training family. We then embed one or more founder sequences from each of the 5,568 smallest families held out from training. ProtREP classifies each held-out test sequence into a new or existing family. We report results in Supplementary Table 9 for (1) all 126,171 held-out test sequences and (2) the 710 held-out test sequences belonging to the 5,568 families that were entirely held out during training (of which 4,858 families were too small for any members to be assigned to the test set).

Saturation mutagenesis experiment. We trained ProtCNN on Pfam-full and predicted the impact of single amino acid mutations by calculating the Kullback–Leibler divergence between the predicted distributions for the original and mutated sequences (Supplementary Table 19) of (1) an ATPase domain sequence (Supplementary Fig. 8) and (2) vasopressin (Supplementary Fig. 8). In the transmembrane regions, amino acid substitutions maintain function in the order FMLVI YACTS WGQHN KRPED for the ATPase and VLIAM FTWCY SNGQH PRDKE for vasopressin. The predicted disruptive effects of charged amino acids and proline agree with prior knowledge.

Domain calling procedure. We designed our experiment of domain calling (not just domain classification) as follows: the input is a full protein (not a precut domain), and the desired outputs are (1) which Pfam families are present and (2) where these domains are within the sequence. Note that an input protein can have 0 or more domains. This task is akin to boundary box annotation in computer vision. A simple approach taken in computer vision to convert an object classifier into a detector is to simply slide the classifier over all candidate bounding boxes and then find where the signal is strongest. We use this approach, first computing all possible (start, end) pairs for an input sequence and running ProtENN on each of them. We output the region where the signal is strongest for a particular family as a called domain. We do not consider very short (start, end) ostensible domain ranges because we find this leads to spurious calls, especially repeats. It is known that domain calls on repeats are a difficult issue, even for HMMs[44]. Although substantially more work is required, the fact that this procedure from the early computer vision literature worked without modification is encouraging.

Pfam clans. Around 45% of Pfam sequences belong to clans[25], groups of evolutionarily related families constructed through manual curation[32,34]. In some cases, a single HMM model cannot capture the full diversity of a large sequence family[32]. Measuring the accuracy of clan-level annotation takes into account the fact that the distinction between different families in the same clan may be less meaningful[33]. In Supplementary Fig. 9, we report accuracy at annotating the 55,604 held-out test sequences that belong to Pfam clans at both (1) the clan level and (2) the family level. Note that the deep models are not provided with clan-level annotations. Both ProtCNN and ProtENN accurately annotate protein domain sequences at both clan and family levels. At the clan level, the error rate of ProtENN is significantly lower for sequence identity in 30–70%; outside this range, neither ProtENN or TPHMM is significantly better ($P < 0.05$, McNemar test). All differences between ProtCNN and TPHMM are significant for sequence identity <70%. At the family level, the neural network models make significantly fewer errors for sequence identities <80%.

Combining ProtENN and HMMER. Combining ProtENN and HMMER yields a model that reduces the error on the clustered split dataset by 38.6%. Supplementary Fig. 10a shows that TPHMM is highly accurate on the held-out test sequences of the clustered split if the HMMER E value is $<10^{-4}$; however, many sequences have HMMER E values $>10^{-2}$, where ProtENN predictions are more accurate. Therefore, based on the HMMER E value, we choose the HMMER or ProtENN prediction. Supplementary Fig. 10b shows accuracy of the combined model as a function of the E value threshold used to determine which prediction is reported.

Similarly, we can build a confidence score for the ProtENN ensemble: to describe the extent to which the predictions of each model in ProtENN agree, we calculate the ensemble consensus as the ratio of votes for a particular label divided by the number of ensemble elements. Supplementary Fig. 11 reports analogous results if the ProtENN ensemble consensus is used in place of the HMMER E value to set the threshold for the combined model. For comparison, Supplementary Fig. 12a reports TPHMM and BLASTp accuracy as a function of HMMER E value. Across nearly all E values, TPHMM predictions are more accurate. As a result, Supplementary Fig. 12b shows that combining TPHMM and BLASTp does not reduce the error rate on the clustered split.

Using ProtENN to increase Pfam coverage. To build Pfam, the HMM models are used to search pfamseq, a large set of sequences drawn from UniProtKB reference proteomes[25,45]. Matches with statistical significance (bit score) below each family gathering threshold are significant. Here, we use all Pfam sequences to train ProtENN and use the trained model to predict family membership for each insignificant match. Supplementary Fig. 11 suggests that ProtENN accuracy saturates at ensemble consensus 60% on our clustered benchmark, while Supplementary Fig. 10 suggests that HMM accuracy increases rapidly for E values $<10^{2}$. We use these criteria to filter the large set of potential insignificant matches and select cases where ProtENN and HMMER agree at the clan or family level. To avoid overcounting, we only consider matches for which (1) the midpoint does not occur within a significant domain, (2) no significant domain sequence midpoint falls in the proposed new match and (3) the overlap is overall less than 20 residues (to account for overlaps between very short calls). This heuristic is conservative, as some regions are truly subsequences of others and are not included in our output set. As noted in the main text, the resulting additions to Pfam are available for download as Pfam-N, which is part of the Pfam v.34.0 release, available at http://ftp.ebi.ac.uk/pub/databases/Pfam/releases/Pfam34.0/.

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

Data availability

The data splits described in this manuscript are available for download at https://console.cloud.google.com/storage/browser/brain-genomics-public/research/proteins/pfam/random_split and https://console.cloud.google.com/storage/browser/brain-genomics-public/research/proteins/pfam/clustered_split, and an interactive notebook for data loading is available at https://www.kaggle.com/googleai/pfam-seed-random-split. Model predictions for Pfam-N are freely available to download as part of the Pfam v.34.0 release from http://ftp.ebi.ac.uk/pub/databases/Pfam/releases/Pfam34.0/.

Code availability

The TensorFlow API, specifically tensorflow-gpu v.1.15.4, was used to implement and train all deep models using the architectures described in the Methods. Code that documents model training using Python v.3.7 is available on GitHub at https://github.com/google-research/google-research/tree/master/using_dl_to_annotate_protein_universe. The training and validation datasets used for creating each model are available as described in the preceding section. Trained models are available in Google Cloud Storage at https://console.cloud.google.com/storage/browser/brain-genomics-public/research/proteins/pfam/models/single_domain_per_sequence_zipped_models, including the ensembles trained on the Pfam-seed random split, Pfam-seed clustered split, Pfam-full random split (all Pfam v.32.0) and the models used to generate Pfam-N v.34.0. ProtCNN inference was run using a custom Python script that (1) read in FASTA records and (2) ran inference of the ProtCNN as a TensorFlow SavedModel. An interactive notebook that demonstrates inference using ProtCNN is available at https://colab.research.google.com/github/google-research/google-research/blob/master/using_dl_to_annotate_protein_universe/neural_network/Neural_network_accuracy_on_random_seed_split.ipynb. An interactive notebook showing use of the trained models to produce Pfam class predictions as well as embeddings is available in GitHub at https://colab.sandbox.google.com/github/google-research/google-research/blob/master/using_dl_to_annotate_protein_universe/Using_Deep_Learning_to_Annotate_the_Protein_Universe.ipynb.

References

40. L.L. Sonnhammer, E., Eddy, S. R. & Durbin, R. Pfam: a comprehensive database of protein domain families based on seed alignments. *Proteins* **28**, 405–420 (1997).
41. 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 (IEEE, 2016).
42. Yu, F. and Koltun, V. Multi-scale context aggregation by dilated convolutions. Preprint at https://arxiv.org/abs/1511.07122 (2015).
43. Altschul, S. F., Gish, W., Miller, W., Myers, E. W. & Lipman, D. J. Basic local alignment search tool. *J. Mol. Biol.* **215**, 403–410 (1990).
44. El-Gebali, S., Richardson, L. & Finn, R. Repeats in Pfam. *EMBL-EBI Training* https://doi.org/10.6019/TOL.Pfam_repeats-t.2018.00001.1 (2018).
45. UniProt Consortium. UniProt: the universal protein knowledgebase. *Nucleic Acids Res.* **46**, 2699 (2018).

Acknowledgements

We thank J. Smith for countless conversations and guidance throughout this project; E. Bixby for an implementation of ragged tensor processing that sped up our ProtCNN implementation substantially on GPU; C. McClean, B. Alipanahi and S. Kearnes for extensive proofreading and feedback and Z. Nado for programming advice. L.J.C. gratefully acknowledges support from the Simons Foundation.

Author contributions

M.L.B., D.B., M.A.D. and L.J.C. conceived the study. All authors designed, implemented and used machine learning models to annotate protein domain sequences, analyzed the data and developed the approach used for Pfam-N. M.L.B., D.B. and L.J.C. wrote the paper, with input from all authors.

Competing interests

M.L.B., D.B., D.H.B., T.S., B.C., D.S., M.A.D. and L.J.C. performed research as part of their employment at Google LLC. Google is a technology company that sells machine learning services as part of its business. Portions of this work are covered by US patent WO2020210591A1, filed by Google.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41587-021-01179-w.

Correspondence and requests for materials should be addressed to Maxwell L. Bileschi or Lucy J. Colwell.

Peer review information *Nature Biotechnology* thanks Christian Dallago for their contribution to the peer review of this work.

Reprints and permissions information is available at www.nature.com/reprints.

nature research

Corresponding author(s):	Lucy Jane Colwell
Last updated by author(s):	Nov 22, 2021

Reporting Summary

Statistics

For all statistical analyses, confirm that the following items are present in the figure legend, table legend, main text, or Methods section.

n/a	Confirmed	
☐	☒	The exact sample size (*n*) for each experimental group/condition, given as a discrete number and unit of measurement
☐	☒	A statement on whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
☐	☒	The statistical test(s) used AND whether they are one- or two-sided *Only common tests should be described solely by name; describe more complex techniques in the Methods section.*
☐	☒	A description of all covariates tested
☐	☒	A description of any assumptions or corrections, such as tests of normality and adjustment for multiple comparisons
☐	☒	A full description of the statistical parameters including central tendency (e.g. means) or other basic estimates (e.g. regression coefficient) AND variation (e.g. standard deviation) or associated estimates of uncertainty (e.g. confidence intervals)
☐	☒	For null hypothesis testing, the test statistic (e.g. *F*, *t*, *r*) with confidence intervals, effect sizes, degrees of freedom and *P* value noted *Give P values as exact values whenever suitable.*
☒	☐	For Bayesian analysis, information on the choice of priors and Markov chain Monte Carlo settings
☒	☐	For hierarchical and complex designs, identification of the appropriate level for tests and full reporting of outcomes
☐	☒	Estimates of effect sizes (e.g. Cohen's *d*, Pearson's *r*), indicating how they were calculated

Our web collection on statistics for biologists contains articles on many of the points above.

Software and code

Policy information about availability of computer code

Data collection	No software was used to collect data for this manuscript, all data was obtained from the Pfam database v32.0 or v34.0. The data splits built in this manuscript have been made freely available from https://console.cloud.google.com/storage/browser/brain-genomics-public/research/proteins/pfam/random_split and https://console.cloud.google.com/storage/browser/brain-genomics-public/research/proteins/pfam/clustered_split.
Data analysis	The Top pick HMM model was implemented using HMMER 3.1b2. The TensorFlow API, specifically tensorflow-gpu v1.15.4, was used to implement and train all deep models using the architectures described in Methods. Code that documents model training using python v3.7 is available on github at \url{https://github.com/google-research/google-research/tree/master/using_dl_to_annotate_protein_universe}. The training and validation datasets used for creating each model are available as described in the preceding section. Trained models are available in Google Cloud Storage at \url{https://console.cloud.google.com/storage/browser/brain-genomics-public/research/proteins/pfam/models/single_domain_per_sequence_zipped_models}, including the ensembles trained on the Pfam seed random split, Pfam seed clustered split, Pfam full random split (all Pfam v32.0), and the models used to generate Pfam-N v34.0. ProtCNN inference was run using a custom python script that (a) read in FASTA records and (b) ran inference of the ProtCNN as a TensorFlow SavedModel. An interactive notebook that demonstrates inference using ProtCNN is available at \url{https://colab.research.google.com/github/google-research/google-research/blob/master/using_dl_to_annotate_protein_universe/neural_network/Neural_network_accuracy_on_random_seed_split.ipynb}. An interactive notebook showing use of the trained models to produce Pfam class predictions as well as embeddings is available in github at \url{https://colab.sandbox.google.com/github/google-research/google-research/blob/master/using_dl_to_annotate_protein_universe/Using_Deep_Learning_to_Annotate_the_Protein_Universe.ipynb}

Data

Policy information about availability of data

All manuscripts must include a data availability statement. This statement should provide the following information, where applicable:

- Accession codes, unique identifiers, or web links for publicly available datasets
- A list of figures that have associated raw data
- A description of any restrictions on data availability

The data splits described in this manuscript are available for download at \url{https://console.cloud.google.com/storage/browser/brain-genomics-public/research/proteins/pfam/random_split}, \url{https://console.cloud.google.com/storage/browser/brain-genomics-public/research/proteins/pfam/clustered_split} and an interactive notebook for data loading is available at \url{https://www.kaggle.com/googleai/pfam-seed-random-split}. Model predictions are freely available to download as part of the Pfam v34.0 release from \url{http://ftp.ebi.ac.uk/pub/databases/Pfam/releases/Pfam34.0/}.

Field-specific reporting

Please select the one below that is the best fit for your research. If you are not sure, read the appropriate sections before making your selection.

[✓ Life sciences] Behavioural & social sciences Ecological, evolutionary & environmental sciences

For a reference copy of the document with all sections, see nature.com/documents/nr-reporting-summary-flat.pdf

Life sciences study design

All studies must disclose on these points even when the disclosure is negative.

Sample size	The sample size was determined by the size of the Pfam data bases.
Data exclusions	We used the Pfam seed and full data sets. No data was excluded from the analysis.
Replication	59 replicate ProtCNN models were trained using different random initialization of parameters.
Randomization	Sequences were split (i) randomly and (ii) randomly by distance into train, tune and held-out test sets.
Blinding	All models were blinded to the labels of all sequences in all held-out test sets.

Reporting for specific materials, systems and methods

We require information from authors about some types of materials, experimental systems and methods used in many studies. Here, indicate whether each material, system or method listed is relevant to your study. If you are not sure if a list item applies to your research, read the appropriate section before selecting a response.

Materials & experimental systems

n/a	Involved in the study	
☒	☐	Antibodies
☒	☐	Eukaryotic cell lines
☒	☐	Palaeontology
☒	☐	Animals and other organisms
☒	☐	Human research participants
☒	☐	Clinical data

Methods

n/a	Involved in the study	
☒	☐	ChIP-seq
☒	☐	Flow cytometry
☒	☐	MRI-based neuroimaging