nature methods

Article

https://doi.org/10.1038/s41592-025-02776-2

Biophysics-based protein language models for protein engineering

Received: 19 March 2024

Accepted: 2 July 2025

Published online: 11 September 2025

Check for updates

Sam Gelman[1,2], Bryce Johnson[1,2], Chase R. Freschlin[3], Arnav Sharma[1,2], Sameer D'Costa[3], John Peters[2,4], Anthony Gitter[1,2,4,6] ✉ & Philip A. Romero[3,5,6] ✉

Protein language models trained on evolutionary data have emerged as powerful tools for predictive problems involving protein sequence, structure and function. However, these models overlook decades of research into biophysical factors governing protein function. We propose mutational effect transfer learning (METL), a protein language model framework that unites advanced machine learning and biophysical modeling. Using the METL framework, we pretrain transformer-based neural networks on biophysical simulation data to capture fundamental relationships between protein sequence, structure and energetics. We fine-tune METL on experimental sequence–function data to harness these biophysical signals and apply them when predicting protein properties like thermostability, catalytic activity and fluorescence. METL excels in challenging protein engineering tasks like generalizing from small training sets and position extrapolation, although existing methods that train on evolutionary signals remain powerful for many types of experimental assays. We demonstrate METL's ability to design functional green fluorescent protein variants when trained on only 64 examples, showcasing the potential of biophysics-based protein language models for protein engineering.

Just as words combine to form sentences that convey meaning in human languages, the specific arrangement of amino acids in proteins can be viewed as an information-rich language describing molecular structure and behavior. Protein language models (PLMs) harness advances in natural language processing to decode intricate patterns and relationships within protein sequences[1]. These models learn meaningful, low-dimensional representations that capture the semantic organization of protein space and have broad utility in protein engineering[2]. PLMs can be adapted to specific protein properties like enzyme activity or stability with limited training examples[3,4], and they can be used in predictive or generative settings to design custom-made proteins with desired characteristics[5,6].

PLMs such as UniRep[7] and evolutionary scale modeling (ESM)[8] are trained on vast repositories of natural protein sequences distributed across the evolutionary tree. The training process typically involves self-supervised autoregressive next-token prediction or masked token prediction[1,9]. Through this process, PLMs learn context-aware representations of amino acids within proteins. Training on examples of natural proteins produces PLMs that implicitly capture protein structure, biological function and other evolutionary pressures. While these models are powerful, they do not take advantage of the extensive knowledge of protein biophysics and molecular mechanisms acquired over the last century; thus, they are largely unaware of the underlying physical principles governing protein function.

[1]Department of Computer Sciences, University of Wisconsin–Madison, Madison, WI, USA. [2]Morgridge Institute for Research, Madison, WI, USA. [3]Department of Biochemistry, University of Wisconsin–Madison, Madison, WI, USA. [4]Department of Biostatistics and Medical Informatics, University of Wisconsin–Madison, Madison, WI, USA. [5]Department of Biomedical Engineering, Duke University, Durham, NC, USA. [6]These authors contributed equally: Anthony Gitter, Philip A. Romero. ✉e-mail: gitter@biostat.wisc.edu; philip.romero@duke.edu

We introduce METL, a PLM that integrates biophysical knowledge during pretraining before being fine-tuned with experimental data for protein engineering applications. Unlike evolutionary-based PLMs, METL is pretrained on biophysical data generated through molecular simulations across diverse protein sequences and structural folds. METL captures biophysical relationships inherent in these molecular simulations and learns a biophysically grounded protein representation. This biophysics-informed approach allows METL to understand and predict protein function based on underlying biophysical mechanisms, offering insights that can complement traditional evolutionary-based models.

Following pretraining, we fine-tune METL using experimental sequence–function data, producing biophysics-aware models that can predict specific protein properties. Experimental data play a critical role in protein engineering by providing direct, empirical relationships between sequence variations and observed functional outcomes. In contrast to zero-shot models that rely solely on pretrained knowledge or de novo models that build completely new proteins from scratch, METL uses experimental data to explicitly predict how sequence changes influence protein function. METL excels in protein engineering tasks like generalizing from small experimental training sets and extrapolating to mutations not observed in the training data. We demonstrate METL's ability to design functional green fluorescent protein (GFP) variants when trained on only 64 sequence–function examples. METL establishes a general framework for incorporating biophysical knowledge into PLMs and will become increasingly powerful with advances in molecular modeling and simulation methods.

Results

Pretraining PLMs with synthetic data

Deep neural networks and language models are revolutionizing protein modeling and design, but these models struggle in low data settings and when generalizing beyond their training data. Although neural networks have proven capable in learning complex sequence–structure–function relationships, they largely ignore the vast accumulated knowledge of protein biophysics. This limits their ability to perform the strong generalization needed for protein engineering, which is the process of modifying a protein to improve its properties[10]. We introduce a framework that incorporates synthetic data from molecular simulations as a means to augment experimental data with biophysical information (Fig. 1). Molecular modeling can generate large datasets revealing mappings from amino acid sequences to protein structure and energetic attributes. Pretraining on this data imparts fundamental biophysical knowledge that can be connected with experimental observations.

We introduce the METL framework for learning protein sequence–function relationships. METL operates in three steps: synthetic data generation, synthetic data pretraining and experimental data fine-tuning. First, we generate synthetic pretraining data via molecular modeling with Rosetta[11] to model the structures of millions of protein sequence variants. For each modeled structure, we extract 55 biophysical attributes including molecular surface areas, solvation energies, van der Waals interactions and hydrogen bonding (Supplementary Table 1). Second, we pretrain a transformer encoder[12] to learn relationships between amino acid sequences and these biophysical attributes and to form an internal representation of protein sequences based on their underlying biophysics. The transformer uses a protein structure-based relative positional embedding[13] that considers the three-dimensional (3D) distances between residues. Finally, we fine-tune the pretrained transformer encoder on experimental sequence–function data to produce a model that integrates prior biophysical knowledge with experimental data. The fine-tuned models input new sequences and predict the particular property learned from the sequence–function data.

We implement two pretraining strategies, METL-Local and METL-Global, that specialize across different scales of protein sequence space (Fig. 1d). METL-Local learns a protein representation targeted to a specific protein of interest. We start with the protein of interest, generate 20 million sequence variants with up to five random amino acid substitutions, model the variants' structures using Rosetta, compute the biophysical attributes and train a transformer encoder to predict the biophysical attributes from the sequence. METL-Local demonstrates strong predictive performance on these attributes (Supplementary Fig. 1a), achieving a mean Spearman correlation of 0.91 for Rosetta's total score energy term across the eight METL-Local source models we trained. Although METL-Local accurately recapitulates the biophysical attributes, the primary purpose of pretraining is to learn an information-rich protein representation that can be fine-tuned on experimental data.

METL-Global extends the pretraining to encapsulate a broader protein sequence space, learning a general protein representation applicable to any protein of interest. We select 148 diverse base proteins[14] (Supplementary Table 2) and generate 200,000 sequence variants with up to five random amino acid substitutions for each. We then model the approximately 30 million resulting structures with Rosetta, extract biophysical attributes, and train a transformer encoder, following a similar methodology to METL-Local. With METL-Global, we observed a substantial difference in predictive ability for in-distribution structures (those included in the METL-Global pretraining data, mean Rosetta total score Spearman correlation of 0.85) and out-of-distribution structures (those not included, mean Rosetta total score Spearman correlation of 0.16; Supplementary Fig. 1b), indicating METL-Global overfits to the 148 base proteins present in the pretraining data. However, we find it still captures biologically relevant amino acid embeddings (Supplementary Fig. 2) that are informative for protein engineering tasks even on the out-of-distribution proteins.

Generalization of biophysics-based PLMs

Generalizing to new data is challenging for neural networks trained with small or biased datasets. This issue is crucial in protein engineering because experimental datasets often have few training examples and/or skewed mutation distributions. These factors impact the accuracy and utility of learned models when using them to design new protein variants.

We rigorously evaluated the predictive generalization performance of METL on 11 experimental datasets, representing proteins of varying sizes, folds and functions: GFP, DLG4-Abundance (DLG4-A), DLG4-Binding (DLG4-B), GB1, GRB2-Abundance (GRB2-A), GRB2-Binding (GRB2-B), Pab1, PTEN-Abundance (PTEN-A), PTEN-Activity (PTEN-E), TEM-1 and Ube4b (Supplementary Table 3). The METL-Global pretraining data contain proteins with sequence and structural similarity to DLG4, GRB2 and TEM-1 (Supplementary Table 4), although their sequence identities are all below 40%. We observed no meaningful performance advantage for these proteins compared to others when using METL-Global to predict Rosetta scores (before fine-tuning) or experimental function (after fine-tuning).

We compared METL to established baseline methods that provide zero-shot or stand-alone predictions, including Rosetta's total score, the evolutionary model of variant effect (EVE)[15] and rapid stability prediction (RaSP)[16]. We also evaluated supervised learning and fine-tuning methods, including linear regression with a one-hot amino acid sequence encoding (Linear), an augmented EVE model that includes the EVE score as an input feature to linear regression in combination with the amino acid sequence (Linear-EVE)[17], a non-parametric transformer for proteins (ProteinNPT)[18] and the ESM-2 (ref. 19) PLM fine-tuned on experimental sequence–function data. We created comprehensive training, validation and test splits, encompassing small training set sizes and difficult extrapolation tasks, and we tested multiple split replicates to account for variation in the selection of training examples.

We evaluated the models' ability to learn from limited data by sampling reduced training sets and evaluating performance as

https://doi.org/10.1038/s41592-025-02776-2

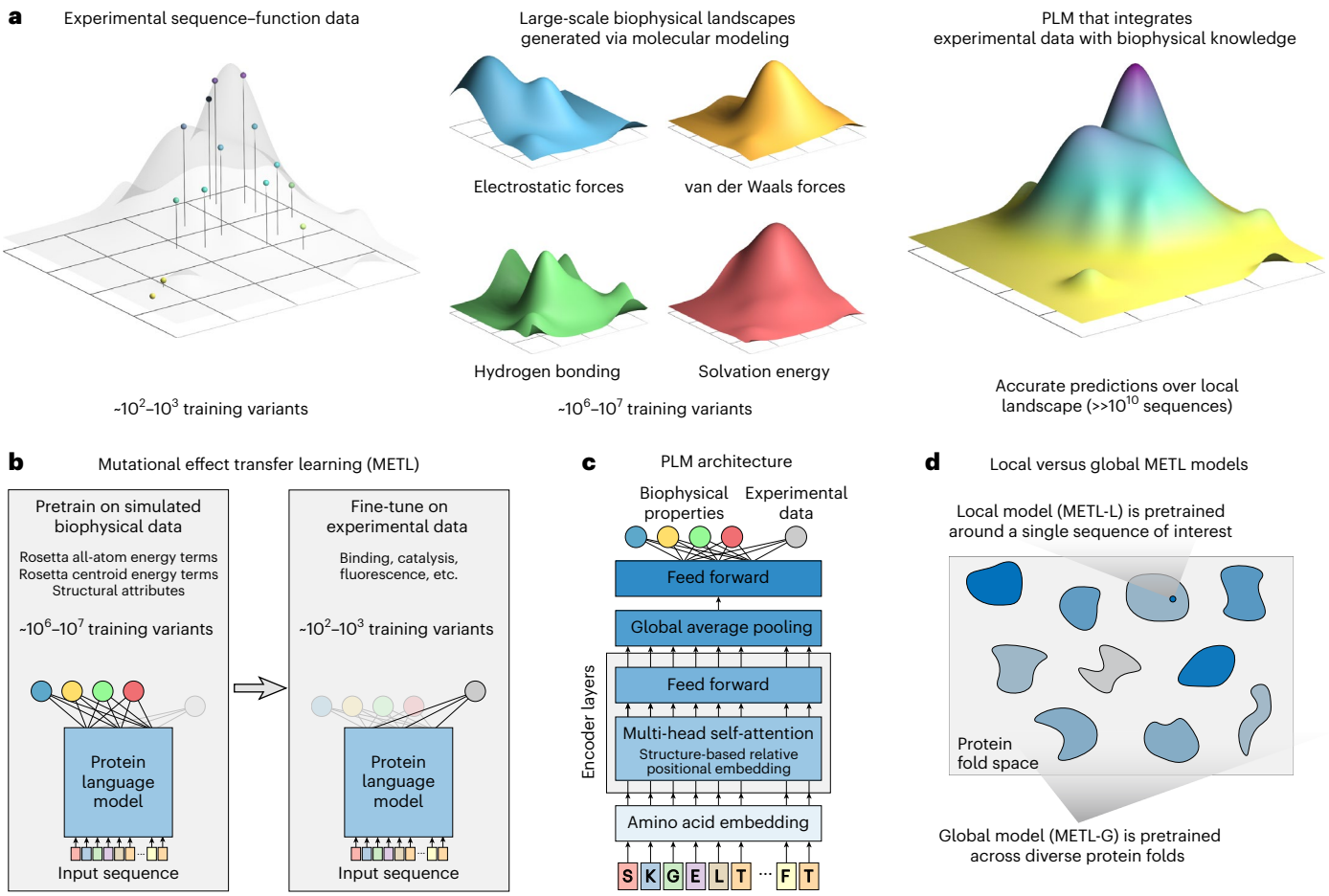


Fig. 1 | METL. a, METL combines sparse experimental protein sequence–function data with dense biophysical simulation data to learn biophysics-informed sequence–function landscapes. **b**, The pretraining phase involves generating millions of protein sequence variants and computing biophysical attributes for them with Rosetta, which are then used to pretrain a PLM. The model is subsequently fine-tuned with experimental sequence–function data to predict protein properties such as binding, enzyme activity, thermostability and expression. **c**, The METL architecture consists of a transformer encoder with a structure-based relative position embedding. **d**, METL-Local and METL-Global differ in the sequences included in the pretraining data. METL-Local trains on the local sequence space around a protein of interest, learning a representation specific to that protein. METL-Global trains on diverse sequences across protein fold space, learning a general-purpose protein representation.

a function of training set size (Fig. 2). The protein-specific models METL-Local, Linear-EVE and ProteinNPT consistently outperformed the general protein representation models METL-Global and ESM-2 on small training sets. Among the protein-specific approaches, the best-performing method on small training sets tended to be either METL-Local or Linear-EVE, with METL-Local demonstrating particularly strong performance on GFP and GB1. While ProteinNPT sometimes surpassed METL-Local on small training sets, ProteinNPT was still generally outperformed by Linear-EVE in those instances. The relative merits of METL-Local versus Linear-EVE partly depend on the respective correlations of Rosetta total score and EVE with the experimental data. However, as the number of training examples increases, the METL-Local performance becomes dominated by dataset-specific effects rather than Rosetta total score relevance (Supplementary Fig. 3). For the general protein models, METL-Global and ESM-2 remained competitive with each other for small- to mid-size training sets, with ESM-2 typically gaining an advantage as training set size increased.

We implemented four challenging extrapolation tasks—mutation, position, regime and score extrapolation—to simulate realistic protein engineering scenarios, such as datasets lacking mutations at certain positions, having biased score distributions with predominantly low-scoring variants and consisting of solely single-substitution variants (Fig. 3). Mutation extrapolation evaluates a model's ability to generalize across the 20 amino acids and make predictions for specific amino acid substitutions not present in the training data[20] (Fig. 3a). The model observes some amino acid types at a given position and must infer the effects of unobserved amino acids. We found ProteinNPT, ESM-2, METL-Local, Linear-EVE and METL-Global all performed well at this task, achieving average Spearman correlations across datasets ranging from ~0.70 to ~0.78. Position extrapolation evaluates a model's ability to generalize across sequence positions and make predictions for amino acid substitutions at sites that do not vary in the training data[20–22] (Fig. 3b). This task is more challenging than mutation extrapolation and requires the model to possess substantial prior knowledge or a structural understanding of the protein[23]. ProteinNPT and METL-Local displayed the strongest average position extrapolation performance with Spearman correlations of 0.65 and 0.59, respectively. METL-Local's success in mutation and position extrapolation relative to METL-Global is likely the result of the local pretraining data, which includes all mutations at all positions, providing the model with comprehensive prior knowledge of the local landscape.

Regime extrapolation tests a model's ability to predict how mutations combine by training on single amino acid substitutions and predicting the effects of multiple substitutions[21,22,24,25] (Fig. 3c and Supplementary Fig. 4). The supervised models generally performed well at regime extrapolation, achieving average Spearman

https://doi.org/10.1038/s41592-025-02776-2

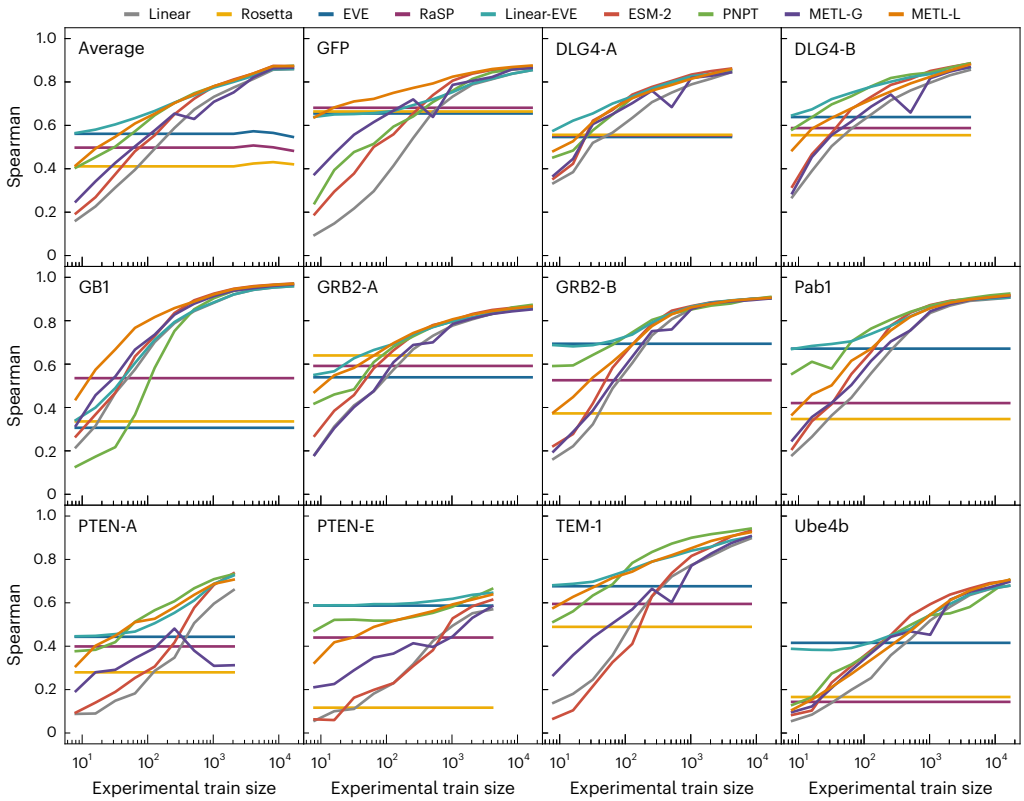


Fig. 2 | Comparative performance of Linear, Rosetta total score, EVE, RaSP, Linear-EVE, ESM-2, ProteinNPT, METL-Global and METL-Local across different training set sizes. Learning curves for 11 datasets showing the test set Spearman correlation between true and predicted protein function scores across a number of training set sizes ranging from 8 to 16,384 examples. We tested multiple replicates for each training set size, starting with 101 replicates for the smallest training set size and decreasing to three replicates for the largest size. We show the median Spearman correlation across these replicates. 'Average' shows the mean of the learning curves across the 11 datasets.

correlations above 0.75. The strong performance of linear regression, which relies on additive assumptions, suggests the sampled functional landscape is dominated by additive effects. ProteinNPT performed slightly worse than the other supervised models, with an average Spearman correlation of 0.67, partly driven by lower performance on the GFP dataset. Score extrapolation tests a model's ability to train on variants with lower-than-wild-type scores and predict variants with higher-than-wild-type scores[25] (Fig. 3d). This proves to be a challenging extrapolation task, with all models achieving a Spearman correlation less than 0.3 for most datasets. The GB1 dataset is an exception for which all supervised models achieved Spearman correlations of at least 0.55, and both METL-Local and METL-Global displayed correlations above 0.7. The difficulty of score extrapolation might be attributed to the fact that the mechanisms to break a protein are distinctly different than those to enhance its activity. It is notable that Rosetta total score and EVE, which are not trained on experimental data, performed worse at score extrapolation than they did at the other extrapolation tasks. This suggests these methods are largely capturing whether a sequence is active or inactive, rather than the finer details of protein activity.

We performed the above prediction and extrapolation tasks with several additional baselines, including METL-Local with random initialization (Supplementary Fig. 5), augmented linear regression with Rosetta's total score as an input feature (Supplementary Fig. 6) and sequence convolutional networks and fully connected networks (Supplementary Fig. 7). METL-Local outperformed these additional baselines on nearly every prediction task for every dataset or provided much better scalability. We evaluated the recall of the top 100 test variants as an alternative metric (Supplementary Fig. 8), which showed that strong Spearman correlation does not necessarily imply strong recall performance. Further, we conducted a systematic evaluation of the METL architecture to investigate one-dimensional (1D; sequence-based) versus 3D (structure-based) relative position embeddings (Supplementary Fig. 9), feature extraction versus fine-tuning (Supplementary Fig. 10), global model sizes (Supplementary Figs. 11 and 12) and the extent of overfitting to the pretraining biophysical data (Supplementary Fig. 13).

Information value of simulated versus experimental data

METL models are trained on both simulated and experimental data. Generating simulated data is orders of magnitude faster and less expensive than experimental data. We wanted to understand how these two sources of data interact and if simulated data can partially compensate for a lack of experimental data. To quantify the relative information value of simulated versus experimental data, we measured the performance of the GB1 METL-Local model pretrained on varying amounts of simulated data and fine-tuned with varying amounts of experimental data (Fig. 4). Increasing both data sources improves model performance, and there are eventually diminishing returns for adding additional simulated and experimental data. The shaded regions of Fig. 4 define iso-performance lines with simulated and experimental data combinations that perform similarly. For instance, a METL-Local model pretrained on 1,000 simulated data points and fine-tuned on 320 experimental data points performs similarly to one pretrained on 8,000 simulated data points and fine-tuned on only 80 experimental data points. In this example, adding 7,000 simulated data points is equivalent to adding 240 experimental data points; thus, ~29 simulated data points give the same performance boost as a single experimental data point.

We observe distinct patterns in how different proteins respond to increasing amounts of simulated pretraining data (Supplementary Fig. 14). For larger proteins like GFP (237 residues), TEM-1 (286 residues) and PTEN (403 residues), we see a threshold effect wherein

https://doi.org/10.1038/s41592-025-02776-2

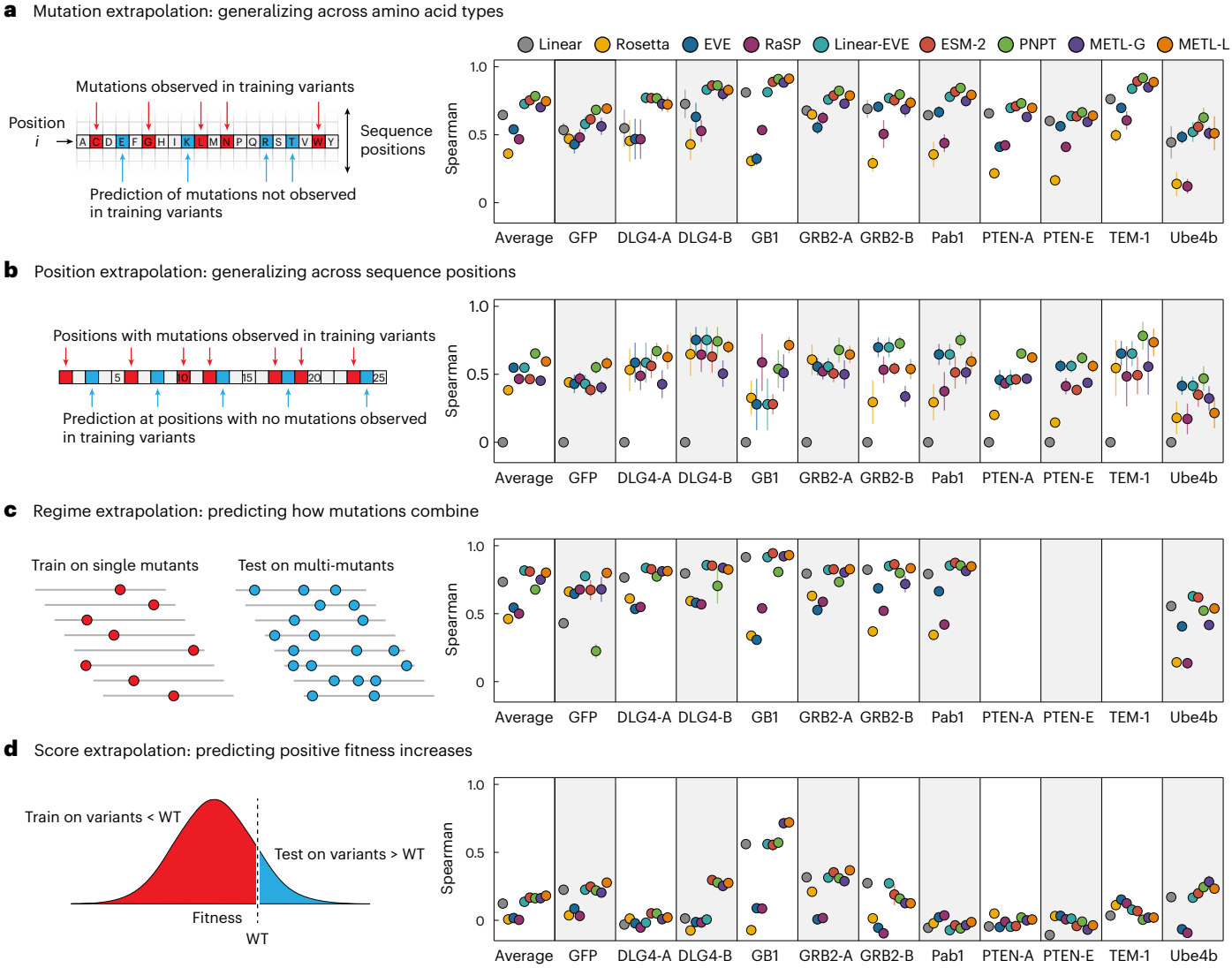


Fig. 3 | Comparative performance across extrapolation tasks. a–d, Correlation performance of Linear, Rosetta total score, EVE, RaSP, Linear-EVE, ESM-2, ProteinNPT, METL-Global and METL-Local on mutation (**a**), position (**b**), regime (**c**) and score (**d**) extrapolation. We tested nine replicates for each type of extrapolation and show the median. Error bars indicate one standard deviation.

performance for a given experimental dataset size remains relatively flat until reaching a critical mass of simulated examples, at which point there is a sharp improvement in downstream performance. In contrast, smaller proteins like GB1 (56 residues), GRB2 (56 residues) and Pab1 (75 residues) show a more gradual response to increased simulated data over the tested dataset sizes. The performance gains are more modest, particularly when experimental data is abundant, but occur more consistently across the range of pretraining data sizes, until hitting a point of diminishing returns. A number of factors could influence this information gain phenomenon, including the protein's size, the protein's structural and functional properties, the experimental assay characteristics and Rosetta's modeling accuracy. Finally, we observe diminishing returns and saturated performance starting with simulated dataset sizes as small as ~16,000 examples, depending on the protein and number of experimental examples. The point of diminishing returns occurs at a substantially smaller number of simulated examples than the ~20 million used for our main results, suggesting that less simulated data could be used to train METL-Local in practice.

Synthetic data pretraining imparts biophysical knowledge

The purpose of METL's pretraining is to learn a useful biophysics-informed protein representation. To further probe METL's pretraining and gain insights into what the PLM has learned, we examined attention maps and residue representations for the GB1 METL-Local model after pretraining on molecular simulations but before fine-tuning on experimental data (Extended Data Fig. 1). Our METL PLMs with 3D relative position embeddings start with a strong inductive bias and include the wild-type protein structure as input. After pretraining, the METL attention map for the wild-type GB1 sequence closely resembles the residue distance matrix of the wild-type GB1 structure (Extended Data Fig. 1a,b). In contrast, an alternative METL model with 1D relative position embeddings that does not use the GB1 structure while training fails to learn an attention map that resembles the GB1 contacts (Extended Data Fig. 1c). The 3D relative position embedding and pretraining successfully allows METL to focus attention on residue pairs that are close in 3D space and may be functionally important.

We further explored the information encoded in the pretrained GB1 METL model by visualizing residue-level representations at each sequence position, averaged across amino acid types (Extended Data Fig. 1d). These residue-level representations show strong clustering based on a residue's relative solvent accessibility (RSA) and weaker organization based on a residue's location in the 3D structure, as observed through visual inspection and qualitative cross-checking with residue–residue distance patterns. Analysis of the additional

https://doi.org/10.1038/s41592-025-02776-2

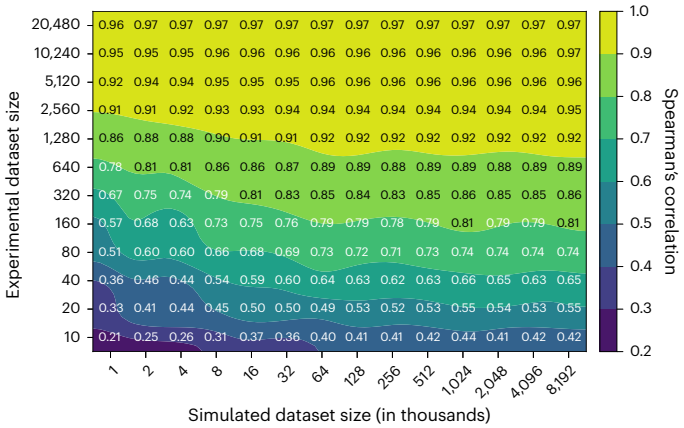


Fig. 4 | Relationship between experimental and simulated data quantities for GB1. The contour plot illustrates the test set Spearman's correlation resulting from training METL-Local with varying amounts of simulated (pretraining) and experimental (fine-tuning) data. The plot displays a grid of Spearman's correlation values corresponding to discrete combinations of experimental and simulated dataset sizes. The model benefits from larger quantities of experimental and simulated data, with the latter producing diminishing returns after approximately 128,000 examples.

datasets in our study reaffirmed these findings: models with 3D relative position embeddings consistently focused attention on spatially proximate residues, and residue representations showed RSA-based clustering patterns across all datasets (Supplementary Figs. 15 and 16). This suggests the pretrained METL models have an underlying understanding of protein structure and important factors like residue burial, even before they have seen any experimental data.

To test whether METL pretraining learns underlying epistatic interactions, we evaluated GB1 variants with well-characterized epistatic effects[26]. The pretrained METL-Local model successfully identifies known interacting positions in GB1's dynamic β1-β2 loop region, with pairwise combinations of positions 7, 9 and 11 all ranking in the top 10% of predicted positional epistasis. The pretrained model also captures strong negative epistasis in the G41L/V54G double mutant (top 0.5% of predicted epistasis), consistent with the known compensatory exchange of small-to-large and large-to-small residues. However, METL underestimates the disulfide-driven positive epistasis in the p.Tyr3Cys/p.Ala26Cys variant, likely due to Rosetta's lack of automatic disulfide bond modeling while generating pretraining data. Overall, these findings demonstrate that METL's pretrained representations capture biologically relevant structural information driving epistasis, while also highlighting a potential limitation of Rosetta-based pretraining.

Function-specific simulations improve METL representations

METL models are pretrained on general structural and biophysical attributes but are not tailored to any particular protein property such as ligand binding, enzyme activity or fluorescence. There is a great body of research using molecular simulations to model protein conformational dynamics, small-molecule ligand and protein docking, enzyme transition state stabilization and other function-specific characteristics[27–31]. These function-specific simulations can be used to generate METL pretraining data that are more closely aligned with target functions and experimental measurements. Similarity between pretraining and target tasks is important to achieve strong performance and avoid detrimental effects in transfer learning[32].

To demonstrate how function-specific simulations can improve the initial pretrained METL model and its performance after fine-tuning, we customized the GB1 simulations to more closely match the experimental conditions. The GB1 experimental data measured the binding interaction between GB1 variants and immunoglobulin G (IgG)[26]. To match this experimentally characterized function, we expanded our Rosetta pipeline to model the GB1–IgG complex and computed 17 attributes related to energy changes upon binding (Supplementary Table 5). These function-specific attributes are more correlated with the experimental data than the general biophysical attributes (Supplementary Fig. 17), suggesting they could provide a valuable signal for model pretraining.

We pretrained a METL PLM that incorporates the IgG binding attributes into its pretraining data and refer to it as METL-Bind (Fig. 5a). METL-Bind is a variant of METL-Local and is specific to GB1. METL-Bind outperformed a standard METL-Local PLM, pretrained only with GB1 biophysical attributes, when fine-tuned on limited experimental data (Fig. 5b,c and Supplementary Fig. 18). We calculated the predictive error for each residue position in the GB1 sequence to understand if the two models specialize on distinct structural regions (Fig. 5d,e). METL-Bind performed better across most residue positions and was notably better at predicting mutation effects at the GB1–IgG interface. The residue where METL-Bind showed the largest improvement was glutamate 27, an interface residue vital for the formation of a stable GB1–IgG complex[33].

While both models converge to similar performance with abundant training data, METL-Bind's superior performance with limited data shows that pretraining on the additional GB1–IgG complex attributes successfully improved the model's learned representation. Many important protein properties can only be assayed accurately using low-throughput techniques. METL-Bind is a promising proof of concept for enhancing predictions when those properties can be approximated computationally. Pretraining on function-specific simulations provides METL with an initial awareness of protein function that can be integrated with limited experimental data.

METL generalization to design diverse GFP variants

Predictive models can guide searches over the sequence–function landscape to enhance natural proteins or design new proteins[6,34,35]. However, these models often face the challenge of making predictions based on limited training data or extrapolating to unexplored regions of sequence space. To demonstrate METL's potential for real protein engineering applications, we tested METL-Local's ability to prioritize fluorescent GFP variants in these challenging design scenarios. We used METL-Local to design 20 GFP sequences that were not part of the original dataset, and we experimentally validated the resulting variants to measure their fluorescence brightness (Fig. 6).

We intentionally set up the design tasks to mimic real protein engineering settings with limited data and extrapolation. We fine-tuned a METL-Local PLM on only 64 GFP variants randomly sampled from the full dataset. The 64 sampled variants had an average of 3.9 amino acid substitutions and a fitness distribution similar to the full dataset (Supplementary Figs. 19 and 20). We designed variants with either 5 or 10 amino acid substitutions, forcing the model to perform regime extrapolation. Furthermore, we tested two design scenarios, Observed AA and Unobserved AA, in which designed variants were constrained to either include or exclude amino acid substitutions observed in the training set, respectively. The Unobserved AA setting forces the model to perform mutation and/or position extrapolation. We designed five variants at each extrapolation distance (5 and 10 mutants) and design setting (Observed AA and Unobserved AA; Supplementary Fig. 21 and Supplementary Table 6). We used simulated annealing to search sequence space for GFP designs that maximize METL-Local's predicted fitness and clustered the designs to select diverse sequences. We also sampled random variants under the same scenarios as the METL designs to serve as baselines.

We had the genes for the 20 GFP METL designs and the 20 random baselines synthesized and cloned into an expression vector as a fusion protein with the fluorescent protein mKate2, emulating the conditions

https://doi.org/10.1038/s41592-025-02776-2

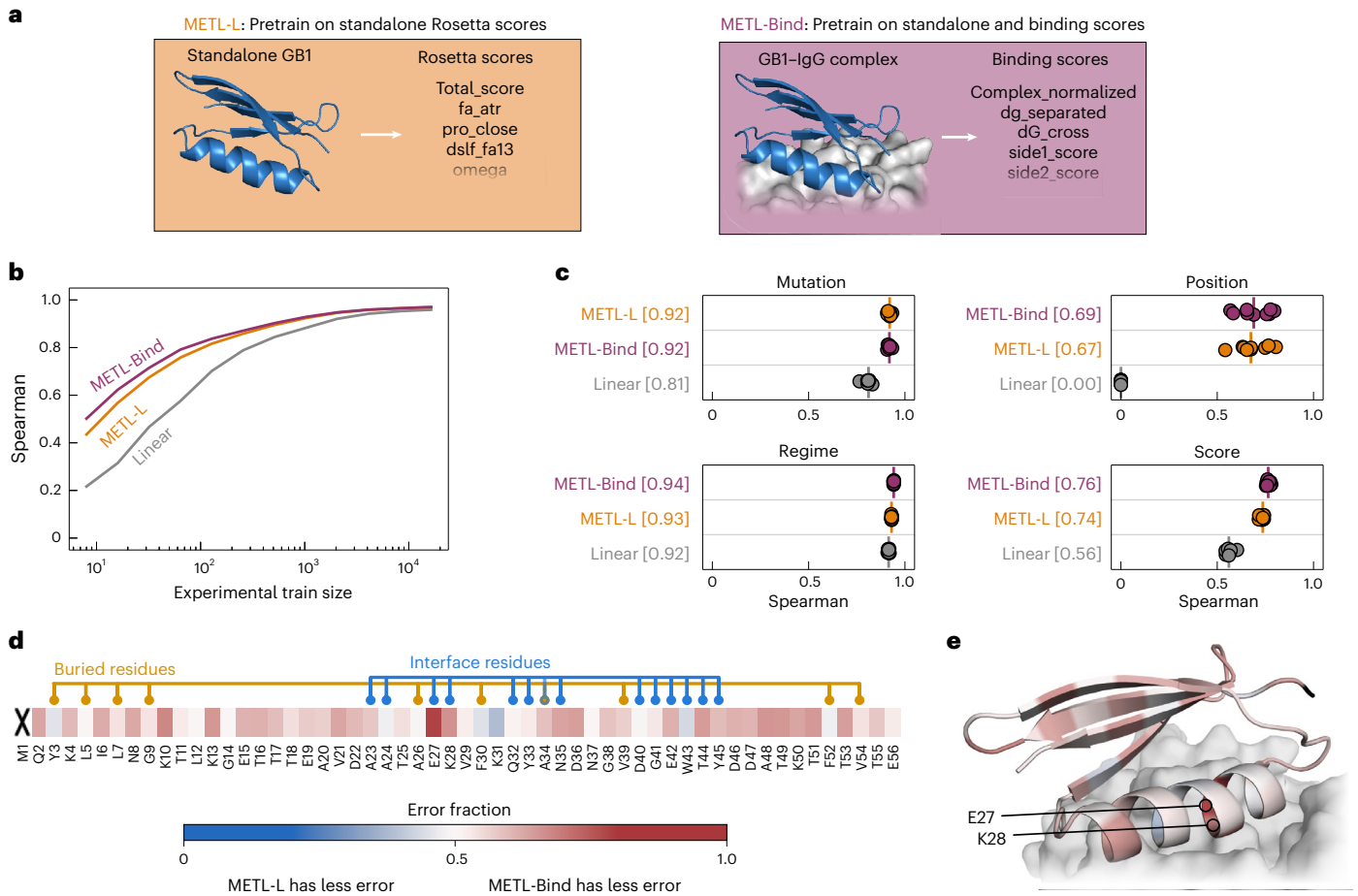


Fig. 5 | Function-specific simulations improve METL pretraining for GB1. **a**, METL-Local (METL-L) pretrains on general Rosetta biophysical attributes from the stand-alone GB1 structure. METL-Bind pretrains on both general biophysical attributes from the stand-alone GB1 structure and binding-specific scores from the GB1–IgG complex structure. **b**,**c**, Learning curves and extrapolation performance for Linear, METL-Local and METL-Bind on the GB1 dataset. We pretrained METL-Local and METL-Bind on the same variants, differing only in the Rosetta score terms. We used the same fine-tuning dataset splits and replicates as in Fig. 2. Each point represents 1 of 9 extrapolation replicates. The vertical bars denote the medians of the replicates, and the square brackets indicate the median Spearman correlations. **d**,**e**, The heat map shows the fraction of test set variants for which METL-Bind has lower error than METL-Local, broken down by sequence position. Results are shown for training set size 32 and averaged across replicates. Position 1 is marked with an 'X' because the dataset does not contain variants with mutations in that position. METL-Bind has less error for 44 of 55 sequence positions. The structure shows the GB1–IgG interface with the GB1 structure colored using the same error fraction as the heat map.

used to collect the training data[36]. The mKate2 is constant in each fusion protein, while the GFP sequence varies. The ratio of a GFP variant's fluorescence to mKate2's fluorescence provides an intrinsic measure of the GFP variant's 'relative brightness' that is independent of the absolute protein expression level[37]. Overall, METL was successful at designing functional GFP variants, with 16 of the 20 designs exhibiting measurable fluorescence (Fig. 6c). Each design setting had notable differences in the success rates and fluorescence characteristics of the designed GFP sequences. The Observed design setting was 100% successful at designing fluorescent five (5/5) and ten (5/5) mutants, demonstrating METL's robust ability to learn from very limited data and extrapolate to higher mutational combinations. The more challenging Unobserved design setting had an 80% (4/5) hit rate with five mutants and a 40% (2/5) hit rate with ten mutants. The Unobserved designs were less bright than wild-type GFP and the Observed designs.

The random baselines provide context for evaluating the designed variants and METL-Local's predictions (Fig. 6d). Across all design scenarios, the random baseline variants exhibited minimal or no fluorescence activity, with the exception of one of the Observed five-mutant baselines, which fluoresced. METL-Local assigns a high predicted score to this variant, showing its ability to recognize functional sequences (Supplementary Fig. 22). Conversely, METL-Local did not predict high scores for any of the other random baselines. This suggests that the functional METL-designed variants likely emerged from the model's understanding of the GFP fluorescence landscape rather than random chance.

The mKate2 fluorescence signal provides additional insight into the designs (Supplementary Fig. 23). The mKate2 protein is constant, so changes in its fluorescence signal are caused by changes in mKate2–GFP fusion protein concentration and thus provide an indirect readout of the GFP designs' folding, stability, solubility and aggregation. The Observed designs all exhibit higher mKate2 fluorescence than wild-type GFP, possibly indicating moderate stabilization, while the Unobserved designs mostly exhibit lower mKate2 fluorescence than wild-type GFP, suggesting destabilization.

Accessing METL tools

In addition to making the METL code, models and datasets available (Methods), we also made them accessible through multiple web interfaces. We provide a Hugging Face interface to download and use our METL models (https://huggingface.co/gitter-lab/METL/) and a Hugging Face Spaces demo (https://huggingface.co/spaces/gitter-lab/METL_demo/)[38]. The Gradio[39] web demo supports generating predictions with our pretrained METL models for a list of sequence variants

https://doi.org/10.1038/s41592-025-02776-2

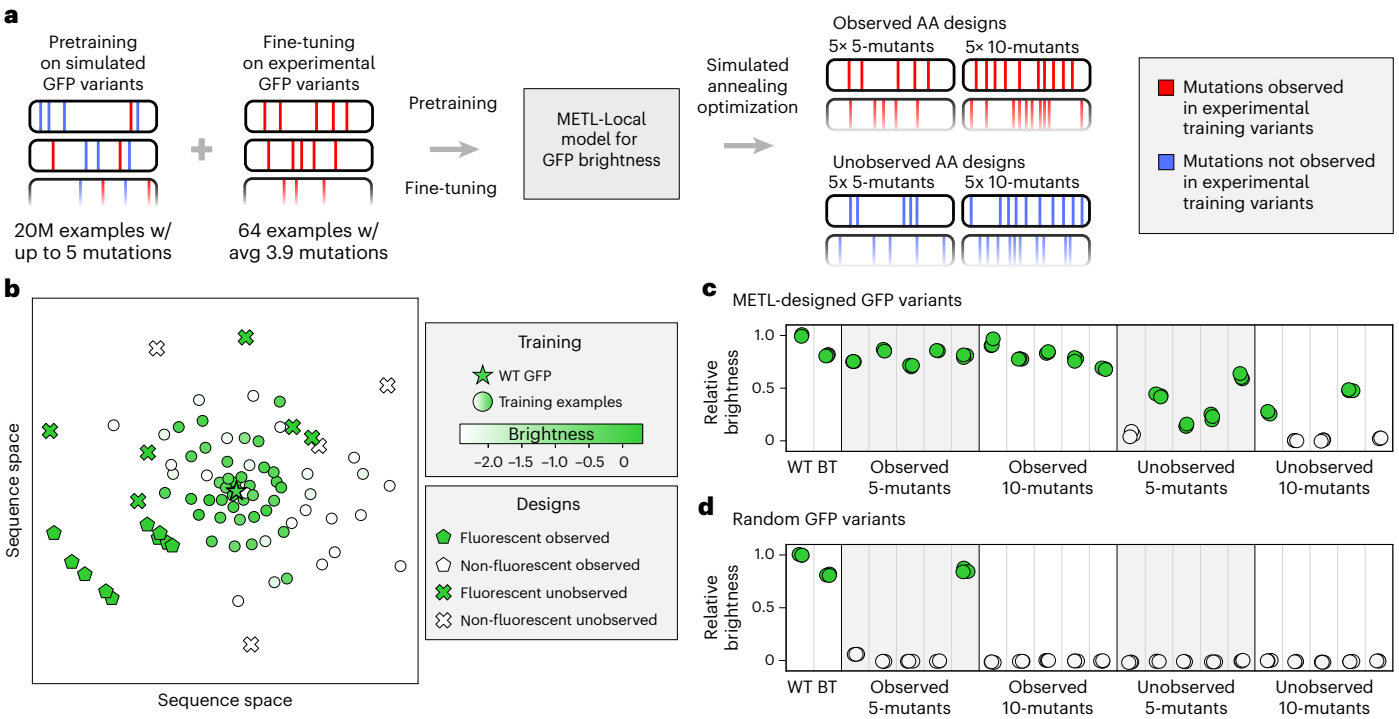


Fig. 6 | Low-N GFP design. a, Overview of the GFP design experiment. We used METL-Local to guide GFP design in a low-N setting with only $N = 64$ experimental training examples. We tested two different design constraints: Observed AA, where sequences contain only amino acid substitutions observed in the training set, and Unobserved AA, where sequences exclude any amino acid substitutions observed in the training set. **b**, Multidimensional scaling (MDS) sequence space visualization of the wild-type GFP sequence, the 64 GFP training sequences and the 20 designed proteins. The designed sequences contain either 5 or 10 amino acid substitutions from the wild-type sequence. Training set sequences are colored on a gradient according to their experimental brightness score. Designed sequences are colored according to whether they exhibited fluorescence, which we define as having at least 10% of wild-type GFP's brightness. **c**, Experimentally characterized brightness of the designed sequences, the best training set sequence (BT) and the wild-type sequence (WT). Each dot represents one distinct sample of the three replicates. **d**, Experimentally characterized brightness of the random baselines.

and visualizes those variants on the protein structure[40]. We created two Colab notebooks to run METL workflows with GPU support, which are available from https://github.com/gitter-lab/metl/. One notebook is for loading a pretrained METL model and fine-tuning it with user-specified protein sequence–function data. The other is for making predictions with pretrained METL models, the same functionality as the Hugging Face Spaces demo but better suited for large datasets. These Colab notebooks are part of the Open Protein Modeling Consortium[41]. Finally, the METL GitHub repository also links to a Jupyter notebook to generate Rosetta pretraining data at scale in the Open Science Pool[42] for eligible researchers.

Discussion

Motivated by decades of research into biophysics, molecular dynamics and protein simulation[11,27,28,31,43], we present METL, which leverages synthetic data from molecular simulations to pretrain biophysics-aware PLMs. These biophysical pretraining signals are in contrast to existing PLMs or multiple sequence alignment (MSA)-based methods that train on natural sequences and capture signals related to evolutionary selective pressures[2,7,8,15,44,45]. By pretraining on large-scale molecular simulations, METL learns a biophysically informed representation of protein space, which provides valuable context for understanding protein sequence–function relationships. Pretrained METL models can be fine-tuned on experimental data to produce models that integrate biophysical knowledge and are capable of predicting properties such as binding, thermostability and expression. METL excels at challenging protein engineering tasks such as learning from limited data and extrapolating to mutations not observed in the training data, enabling the design of new proteins with desired properties.

Our results highlight important differences between evolutionary data and biophysical simulations, especially in terms of their effectiveness for pretraining PLMs to understand sequence–function relationships and predict experimental functions. Evolutionary data, consisting of massive collections of naturally evolved protein sequences, capture information relevant to organismal fitness, including protein expression, folding, stability and biological function. However, the precise selective pressures for each protein are different and largely unknown, and evolutionary patterns can be confounded by historical events, phylogenetic biases and unequal sequence sampling[46]. In contrast, biophysical simulations allow precise control of the input sequence distribution, even sequences with nonnatural amino acids[47,48], and capture fundamental properties of protein structure and energetics. Yet, biophysical simulations are only imperfect approximations of the true physics.

Generally, we found that the protein-specific models METL-Local, Linear-EVE and ProteinNPT demonstrated superior performance compared to general protein representation models METL-Global and ESM-2. The relative performance of METL-Local and Linear-EVE was partly determined by a dataset's correlation with Rosetta total score and EVE, respectively. Certain protein properties and experimental measurements more closely align with either biophysical or evolutionary signals[49–51], providing guidance on where different models may excel. One of METL's key strengths is its ability to incorporate function-specific molecular modeling and simulations. For instance, pretraining on GB1–IgG binding data led to improved performance compared to our standard METL-Local model, which was pretrained only on GB1 structure-derived data. This opens the door to incorporating more sophisticated simulations, such as dynamic simulations of conformational transitions in allosteric regulation,

quantum mechanics/molecular mechanics studies of enzyme catalysis, coarse-grained models of macromolecular machines and small-molecule docking to assess binding specificity. It would also be straightforward to extend METL to make multitask predictions, such as both GB1 thermostability and GB1–IgG binding affinity.

METL-Global represents an initial step toward a universal biophysics-based protein representation, providing comparable or better performance than a similarly sized ESM-2 model when fine-tuning with small training sets (Fig. 2). Future iterations could expand the number and diversity of protein structures used for pretraining[52,53] and use meta-learning strategies[54–56] to alleviate overfitting during pretraining (Supplementary Fig. 1). In this study, we intentionally examined biophysical and evolutionary signals separately by training METL models from scratch and comparing them to evolutionary-based models. Future METL-Global models could integrate these signals by leveraging evolutionary PLMs as a pretrained foundation, potentially enhancing generalization by combining complementary information from both domains. Although sequence-based PLMs can learn about protein structure from evolutionary statistics[19,57–59], many recent PLMs directly incorporate structural information[60–64], and we envision METL-Global would continue to use this prior knowledge when it is available.

Prior studies have integrated biophysics and machine learning by using biophysics-based features as input to machine learning models[17,65–72]. Unlike a fine-tuned METL-Local model, these approaches must run biophysics calculations for each sequence prediction, which could limit their scalability to search sequence space for protein design. Other related work uses machine learning to approximate molecular simulations, often with the goal of obtaining much faster approximate models[73–80]. This scenario is similar to METL's pretraining stage. METL's pretraining on biophysical attributes is also related to the long-standing problem of predicting protein stability[16,81–91]. Finally, machine learning has been integrated into Rosetta to guide its sampling[92].

Machine learning-guided protein engineering is often constrained by limited experimental data[34,93–98]. We demonstrated METL's performance in realistic low data (low-N) and extrapolation settings. PLMs are an important component in many existing methods for low-N protein engineering[3,18,99–107]. Other computational strategies include Gaussian processes[100,108,109], augmented regression models[17,110,111], custom protein representations that can produce pretraining data[112], representations of proteins' 3D shape[113], active learning[114], few-shot learning[115], meta learning[116,117], contrastive fine-tuning[118] and causal inference[119].

Our GFP design experiments demonstrated METL's ability to learn from only 64 training examples and generalize to distant and unexplored regions of sequence space. METL's success in the Unobserved AA setting was notable because it required the model to infer the effects of unseen mutations and predict how these mutations combine in five and ten mutants. While none of the designed GFPs exceeded wild-type brightness, many exhibited higher absolute fluorescence signals, suggesting improved expression and stability. In limited data settings, METL-Local's biophysical prior may indirectly improve designs through stabilizing effects rather than directly improving the brightness.

Examples across diverse scientific domains have demonstrated the power of combining simulations and machine learning[31,120–124]. METL fits within this broader trend and represents an important step toward effectively integrating biophysics insights with machine learning-based protein fitness prediction. The METL framework pretrains PLMs on molecular simulations to capture accumulated biophysical knowledge, and this pretraining strategy will benefit from continued advances in computation and molecular simulation. METL can pretrain on general structural and energetic terms or more focused function-specific terms, offering the potential to model completely nonnatural protein functions with nonexistent evolutionary signals. PLMs fluent in fundamental biophysical dialect will push the boundaries of protein design to new realms of sequence–function space.

Online content

Any methods, additional references, Nature Portfolio 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/s41592-025-02776-2.

References

1. Chandra, A., Tünnermann, L., Löfstedt, T. & Gratz, R. Transformer-based deep learning for predicting protein properties in the life sciences. *eLife* **12**, e82819 (2023).
2. Bepler, T. & Berger, B. Learning the protein language: evolution, structure, and function. *Cell Systems* **12**, 654–669 (2021).
3. Biswas, S., Khimulya, G., Alley, E. C., Esvelt, K. M. & Church, G. M. Low-N protein engineering with data-efficient deep learning. *Nat. Methods* **18**, 389–396 (2021).
4. Yang, K. K., Wu, Z., Bedbrook, C. N. & Arnold, F. H. Learned protein embeddings for machine learning. *Bioinformatics* **34**, 2642–2648 (2018).
5. Munsamy, G., Lindner, S., Lorenz, P. & Ferruz, N. ZymCTRL: a conditional language model for the controllable generation of artificial enzymes. *Machine Learning in Structural Biology Workshop at the 36th Conference on Neural Information Processing Systems* (2022).
6. Madani, A. et al. Large language models generate functional protein sequences across diverse families. *Nat. Biotechnol.* **41**, 1099–1106 (2023).
7. 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).
8. 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).
9. Notin, P., Rollins, N., Gal, Y., Sander, C. & Marks, D. Machine learning for functional protein design. *Nat. Biotechnol.* **42**, 216–228 (2024).
10. Bornscheuer, U. T. et al. Engineering the third wave of biocatalysis. *Nature* **485**, 185–194 (2012).
11. Alford, R. F. et al. The Rosetta all-atom energy function for macromolecular modeling and design. *J. Chem. Theory Comput.* **13**, 3031–3048 (2017).
12. Vaswani, A. et al. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **30**, 5998–6008 (2017).
13. Shaw, P., Uszkoreit, J. & Vaswani, A. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Vol. 2 (Short Papers), 464–468. https://doi.org/10.18653/v1/N18-2074 (Association for Computational Linguistics, 2018).
14. Kosciolek, T. & Jones, D. T. De novo structure prediction of globular proteins aided by sequence variation-derived contacts. *PLoS ONE* **9**, e92197 (2014).
15. Frazer, J. et al. Disease variant prediction with deep generative models of evolutionary data. *Nature* **599**, 91–95 (2021).
16. Blaabjerg, L. M. et al. Rapid protein stability prediction using deep learning representations. *eLife* **12**, e82593 (2023).
17. Hsu, C., Nisonoff, H., Fannjiang, C. & Listgarten, J. Learning protein fitness models from evolutionary and assay-labeled data. *Nat. Biotechnol.* **40**, 1114–1122 (2022).
18. Notin, P., Weitzman, R., Marks, D. S. & Gal, Y. ProteinNPT: improving protein property prediction and design with non-parametric transformers. *Adv. Neural Inf. Process. Syst.* **36**, 33529–33563 (2023).

19. Lin, Z. et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science* **379**, 1123–1130 (2023).
20. Gelman, S., Fahlberg, S. A., Heinzelman, P., Romero, P. A. & Gitter, A. Neural networks to learn protein sequence–function relationships from deep mutational scanning data. *Proc. Natl Acad. Sci. USA* **118**, e2104878118 (2021).
21. Chen, L. et al. Learning protein fitness landscapes with deep mutational scanning data from multiple sources. *Cell Syst.* **14**, 706–721 (2023).
22. Michael, R. et al. Assessing the performance of protein regression models. Preprint at *bioRxiv* https://doi.org/10.1101/2023.06.18.545472 (2023).
23. Sandhu, M. et al. Investigating the determinants of performance in machine learning for protein fitness prediction. *Protein Sci.* **34**, e70235 (2025).
24. Luo, Y. et al. ECNet is an evolutionary context-integrated deep learning framework for protein engineering. *Nat. Commun.* **12**, 5743 (2021).
25. Dallago, C. et al. FLIP: benchmark tasks in fitness landscape inference for proteins. In *Proc. Neural Information Processing Systems Track on Datasets and Benchmarks* (eds Vanschoren, J. & Yeung, S.) (NeurIPS 2021).
26. Olson, C. A., Wu, N. C. & Sun, R. A comprehensive biophysical description of pairwise epistasis throughout an entire protein domain. *Curr. Biol.* **24**, 2643–2651 (2014).
27. Villà, J. & Warshel, A. Energetics and dynamics of enzymatic reactions. *J. Phys. Chem. B* **105**, 7887–7907 (2001).
28. Borrelli, K. W., Vitalis, A., Alcantara, R. & Guallar, V. PELE: protein energy landscape exploration. A novel Monte Carlo based technique. *J. Chem. Theory Comput.* **1**, 1304–1311 (2005).
29. Cross, J. B. et al. Comparison of several molecular docking programs: pose prediction and virtual screening accuracy. *J. Chem. Inf. Model.* **49**, 1455–1474 (2009).
30. 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).
31. Eastman, P. et al. OpenMM 8: molecular dynamics simulation with machine learning potentials. *J. Phys. Chem. B* **128**, 109–116 (2024).
32. Wang, Z., Dai, Z., Póczos, B. & Carbonell, J. Characterizing and avoiding negative transfer. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 11285–11294 https://doi.org/10.1109/CVPR.2019.01155 (2019).
33. Sloan, D. J. & Hellinga, H. W. Dissection of the protein G B1 domain binding site for human IgG Fc fragment. *Protein Sci.* **8**, 1643–1648 (1999).
34. Bedbrook, C. N., Yang, K. K., Rice, A. J., Gradinaru, V. & Arnold, F. H. Machine learning to design integral membrane channelrhodopsins for efficient eukaryotic expression and plasma membrane localization. *PLoS Comput. Biol.* **13**, e1005786 (2017).
35. Johansson, K. E., Lindorff-Larsen, K. & Winther, J. R. Global analysis of multi-mutants to improve protein function. *J. Mol. Biol.* **435**, 168034 (2023).
36. Sarkisyan, K. S. et al. Local fitness landscape of the green fluorescent protein. *Nature* **533**, 397–401 (2016).
37. Cranfill, P. J. et al. Quantitative assessment of fluorescent proteins. *Nat. Methods* **13**, 557–562 (2016).
38. Wolf, T. et al. Transformers: state-of-the-art natural language processing. in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* 38–45 https://doi.org/10.18653/v1/2020.emnlp-demos.6 (Association for Computational Linguistics, 2020).
39. Abid, A. et al. Gradio: hassle-free sharing and testing of ML models in the wild. Preprint at https://arxiv.org/abs/1906.02569 (2019).
40. Rego, N. & Koes, D. 3Dmol.js: molecular visualization with WebGL. *Bioinformatics* **31**, 1322–1324 (2015).
41. Su, J. et al. SaprotHub: making protein modeling accessible to all biologists. Preprint at *bioRxiv* https://doi.org/10.1101/2024.05.24.595648 (2024).
42. Pordes, R. et al. The Open Science Grid. *J. Phys. Conf. Ser.* **78**, 012057 (2007).
43. Hollingsworth, S. A. & Dror, R. O. Molecular dynamics simulation for all. *Neuron* **99**, 1129–1143 (2018).
44. Elnaggar, A. et al. ProtTrans: toward understanding the language of life through self-supervised learning. *IEEE Trans. Pattern Anal. Mach. Intell.* **44**, 7112–7127 (2022).
45. Yang, K. K., Fusi, N. & Lu, A. X. Convolutions are competitive with transformers for protein sequence pretraining. *Cell Systems* https://doi.org/10.1016/j.cels.2024.01.008 (2024).
46. Ding, F. & Steinhardt, J. N. Protein language models are biased by unequal sequence sampling across the tree of life. Preprint at *bioRxiv* https://doi.org/10.1101/2024.03.07.584001 (2024).
47. Renfrew, P. D., Choi, E. J., Bonneau, R. & Kuhlman, B. Incorporation of noncanonical amino acids into Rosetta and use in computational protein-peptide interface design. *PLoS ONE* **7**, e32637 (2012).
48. Li, Y. & Dalby, P. A. Engineering of enzymes using non-natural amino acids. *Bioscience Rep.* **42**, BSR20220168 (2022).
49. Høie, M. H., Cagiada, M., Frederiksen, A. H. B., Stein, A. & Lindorff-Larsen, K. Predicting and interpreting large-scale mutagenesis data using analyses of protein stability and conservation. *Cell Rep.* https://doi.org/10.1016/j.celrep.2021.110207 (2022).
50. Gerasimavicius, L., Livesey, B. J. & Marsh, J. A. Correspondence between functional scores from deep mutational scans and predicted effects on protein stability. *Protein Sci.* **32**, e4688 (2023).
51. Notin, P. et al. ProteinGym: large-scale benchmarks for protein fitness prediction and design. *Adv. Neural Info. Processing Syst.* **36**, 64331–64379 (2023).
52. Berman, H. M. et al. The Protein Data Bank. *Nucleic Acids Res.* **28**, 235–242 (2000).
53. Varadi, M. et al. AlphaFold Protein Structure Database: massively expanding the structural coverage of protein-sequence space with high-accuracy models. *Nucleic Acids Res.* **50**, D439–D444 (2022).
54. Finn, C., Abbeel, P. & Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, 1126–1135 (PMLR, 2017).
55. Li, D., Yang, Y., Song, Y. -Z. & Hospedales, T. Learning to generalize: meta-learning for domain generalization. *Proceedings of the AAAI Conference on Artificial Intelligence* https://doi.org/10.1609/aaai.v32i1.11596 (2018).
56. Scalia, G. et al. A high-throughput phenotypic screen combined with an ultra-large-scale deep learning-based virtual screening reveals novel scaffolds of antibacterial compounds. Preprint at *bioRxiv* https://doi.org/10.1101/2024.09.11.612340 (2024).
57. Wu, R. et al. High-resolution de novo structure prediction from primary sequence. Preprint at *bioRxiv* https://doi.org/10.1101/2022.07.21.500999 (2022).
58. Fang, X. et al. A method for multiple-sequence-alignment-free protein structure prediction using a protein language model. *Nat. Mach. Intell.* **5**, 1087–1096 (2023).
59. Zhang, Z. et al. Protein language models learn evolutionary statistics of interacting sequence motifs. *Proc. Natl Acad. Sci. USA* **121**, e2406285121 (2024).

60. Zhang, Z. et al. A systematic study of joint representation learning on protein sequences and structures. Preprint at https://arxiv.org/abs/2303.06275 (2023).
61. Su, J. et al. SaProt: protein language modeling with structure-aware vocabulary. In *Proceedings of the Twelfth International Conference on Learning Representations* (2024).
62. Li, M. et al. ProSST: protein language modeling with quantized structure and disentangled attention. Preprint at *bioRxiv* https://doi.org/10.1101/2024.04.15.589672 (2024).
63. Hayes, T. et al. Simulating 500 million years of evolution with a language model. *Science.* **387**, 850–858 (2025).
64. Heinzinger, M. et al. Bilingual language model for protein sequence and structure. *NAR Genom. Bioinform.* **6**, lqae150 (2024).
65. Clark, T. et al. Machine learning-guided antibody engineering that leverages domain knowledge to overcome the small data problem. Preprint at *bioRxiv* https://doi.org/10.1101/2023.06.02.543458 (2023).
66. Yutzy, L. et al. Augmentation of structure information to the sequence-based machine learning-assisted directed protein evolution. Preprint at *ChemRxiv* https://doi.org/10.26434/chemrxiv-2023-llpnk (2023).
67. Harmalkar, A. et al. Toward generalizable prediction of antibody thermostability using machine learning on sequence and structure features. *mAbs* **15**, 2163584 (2023).
68. Wang, A., Amini, A. P., Lu, A. X. & Yang, K. K. Learning from physics-based features improves protein property prediction. in *Machine Learning for Structural Biology Workshop at the 36th Conference on Neural Information Processing Systems* (2022).
69. Nisonoff, H., Wang, Y. & Listgarten, J. Coherent blending of biophysics-based knowledge with Bayesian neural networks for robust protein property prediction. *ACS Synth. Biol.* **12**, 3242–3251 (2023).
70. Nordquist, E. et al. Incorporating physics to overcome data scarcity in predictive modeling of protein function: a case study of BK channels. *PLoS Comput. Biol.* **19**, e1011460 (2023).
71. Venanzi, N. A. E. et al. Machine learning integrating protein structure, sequence, and dynamics to predict the enzyme activity of bovine enterokinase variants. *J. Chem. Info. Model.* **64**, 2681–2694 (2024).
72. Wittmann, B. J., Yue, Y. & Arnold, F. H. Informed training set design enables efficient machine learning-assisted directed protein evolution. *Cell Syst.* **12**, 1026–1045 (2021).
73. Hou, C. & Shen, Y. SeqDance: a protein language model for representing protein dynamic properties. Preprint at *bioRxiv* https://doi.org/10.1101/2024.10.11.617911 (2024).
74. Wang, T. et al. Ab initio characterization of protein molecular dynamics with AI2BMD. *Nature* **635**, 1019–1027 (2024).
75. Lipsh-Sokolik, R. et al. Combinatorial assembly and design of enzymes. *Science* **379**, 195–201 (2023).
76. Weinstein, J. Y. et al. Designed active-site library reveals thousands of functional GFP variants. *Nat. Commun.* **14**, 2890 (2023).
77. Omar, S. I., Keasar, C., Ben-Sasson, A. J. & Haber, E. Protein design using physics informed neural networks. *Biomolecules* **13**, 457 (2023).
78. Ramírez-Palacios, C. & Marrink, S. J. Super high-throughput screening of enzyme variants by spectral graph convolutional neural networks. *J. Chem. Theory Comput.* **19**, 4668–4677 (2023).
79. Zheng, S. et al. Predicting equilibrium distributions for molecular systems with deep learning. *Nat. Mach. Intell.* **6**, 558–567 (2024).
80. Ni, B., Kaplan, D. L. & Buehler, M. J. ForceGen: end-to-end de novo protein generation based on nonlinear mechanical unfolding responses using a language diffusion model. *Sci. Adv.* **10**, eadl4000 (2024).
81. Capriotti, E., Fariselli, P. & Casadio, R. I-Mutant2.0: predicting stability changes upon mutation from the protein sequence or structure. *Nucleic Acids Res.* **33**, W306–W310 (2005).
82. Folkman, L., Stantic, B., Sattar, A. & Zhou, Y. EASE-MM: sequence-based prediction of mutation-induced stability changes with feature-based multiple models. *J. Mol. Biol.* **428**, 1394–1405 (2016).
83. Cao, H., Wang, J., He, L., Qi, Y. & Zhang, J. Z. DeepDDG: predicting the stability change of protein point mutations using neural networks. *J. Chem. Info. Model.* **59**, 1508–1514 (2019).
84. Chen, Y. et al. PremPS: predicting the impact of missense mutations on protein stability. *PLoS Comput. Biol.* **16**, e1008543 (2020).
85. Li, B., Yang, Y. T., Capra, J. A. & Gerstein, M. B. Predicting changes in protein thermodynamic stability upon point mutation with deep 3D convolutional neural networks. *PLoS Comput. Biol.* **16**, e1008291 (2020).
86. Wang, S., Tang, H., Zhao, Y. & Zuo, L. BayeStab: predicting effects of mutations on protein stability with uncertainty quantification. *Protein Sci.* **31**, e4467 (2022).
87. Hummer, A. M., Schneider, C., Chinery, L. & Deane, C. M. Investigating the volume and diversity of data needed for generalizable antibody-antigen $\Delta\Delta G$ prediction. *Nat. Comput. Sci.* **5**, 635–647 (2025).
88. Zhou, Y., Pan, Q., Pires, D. E. V., Rodrigues, C. H. M. & Ascher, D. B. DDMut: predicting effects of mutations on protein stability using deep learning. *Nucleic Acids Res.* **51**, W122–W128 (2023).
89. Dieckhaus, H., Brocidiacono, M., Randolph, N. Z. & Kuhlman, B. Transfer learning to leverage larger datasets for improved prediction of protein stability changes. *Proc. Natl Acad. Sci. USA* **121**, e2314853121 (2024).
90. Boyer, S., Money-Kyrle, S. & Bent, O. Predicting protein stability changes under multiple amino acid substitutions using equivariant graph neural networks. Preprint at https://arxiv.org/abs/2305.19801 (2023).
91. Sun, J., Zhu, T., Cui, Y. & Wu, B. Structure-based self-supervised learning enables ultrafast protein stability prediction upon mutation. *Innovation* **6**, 100750 (2025).
92. Ertelt, M., Moretti, R., Meiler, J. & Schoeder, C. T. Self-supervised machine learning methods for protein design improve sampling but not the identification of high-fitness variants. *Sci. Adv.* **11**, eadr7338 (2025).
93. Liao, J. et al. Engineering proteinase K using machine learning and synthetic genes. *BMC Biotechnol.* **7**, 16 (2007).
94. Saito, Y. et al. Machine-learning-guided mutagenesis for directed evolution of fluorescent proteins. *ACS Synth. Biol.* **7**, 2014–2022 (2018).
95. Wu, Z., Kan, S. B. J., Lewis, R. D., Wittmann, B. J. & Arnold, F. H. Machine learning-assisted directed protein evolution with combinatorial libraries. *Proc. Natl Acad. Sci. USA* **116**, 8852–8858 (2019).
96. Saito, Y. et al. Machine-learning-guided library design cycle for directed evolution of enzymes: the effects of training data composition on sequence space exploration. *ACS Catalysis* **11**, 14615–14624 (2021).
97. Greenhalgh, J. C., Fahlberg, S. A., Pfleger, B. F. & Romero, P. A. Machine learning-guided acyl-ACP reductase engineering for improved in vivo fatty alcohol production. *Nat. Commun.* **12**, 5825 (2021).
98. Wait, S. J. et al. Machine learning-guided engineering of genetically encoded fluorescent calcium indicators. *Nat. Comput. Sci.* **4**, 224–236 (2024).
99. Yamaguchi, H. & Saito, Y. Evotuning protocols for Transformer-based variant effect prediction on multi-domain proteins. *Brief. Bioinform.* **22**, bbab234 (2021).

100. Hoffbauer, T. & Strodel, B. TransMEP: transfer learning on large protein language models to predict mutation effects of proteins from a small known dataset. Preprint at *bioRxiv* https://doi.org/10.1101/2024.01.12.575432 (2024).
101. Shanehsazzadeh, A., Belanger, D. & Dohan, D. Is transfer learning necessary for protein landscape prediction? Preprint at https://arxiv.org/abs/2011.03443 (2020).
102. Jiang, K. et al. Rapid in silico directed evolution by a protein language model with EVOLVEpro. *Science*. https://doi.org/10.1126/science.adr6006 (2024).
103. Barbero-Aparicio, J. A., Olivares-Gil, A., Rodríguez, J. J., García-Osorio, C. & Díez-Pastor, J. F. Addressing data scarcity in protein fitness landscape analysis: a study on semi-supervised and deep transfer learning techniques. *Inf. Fusion* **102**, 102035 (2024).
104. Hawkins-Hooker, A., Kmec, J., Bent, O. & Duckworth, P. Likelihood-based fine-tuning of protein language models for few-shot fitness prediction and design. Preprint at *bioRxiv* https://doi.org/10.1101/2024.05.28.596156 (2024).
105. Chen, Y., Xu, Y., Liu, D., Xing, Y. & Gong, H. An end-to-end framework for the prediction of protein structure and fitness from single sequence. *Nat. Commun.* **15**, 7400 (2024).
106. Li, M. et al. SESNet: sequence-structure feature-integrated deep learning method for data-efficient protein engineering. *J. Cheminform.* **15**, 12 (2023).
107. Zhou, Z. et al. Enhancing efficiency of protein language models with minimal wet-lab data through few-shot learning. *Nat. Commun.* **15**, 5566 (2024).
108. Romero, P. A., Krause, A. & Arnold, F. H. Navigating the protein fitness landscape with Gaussian processes. *Proc. Natl Acad. Sci. USA* **110**, E193–E201 (2013).
109. Vornholt, T. et al. Enhanced sequence-activity mapping and evolution of artificial metalloenzymes by active learning. *ACS Cent. Sci.* **10**, 1357–1370 (2024).
110. Illig, A. -M., Siedhoff, N. E., Schwaneberg, U. & Davari, M. D. A hybrid model combining evolutionary probability and machine learning leverages data-driven protein engineering. Preprint at *bioRxiv* https://doi.org/10.1101/2022.06.07.495081 (2022).
111. Boca, A. & Mathis, S. Predicting protein variants with equivariant graph neural networks. Preprint at https://arxiv.org/abs/2306.12231 (2023).
112. Wirnsberger, G., Pritišanac, I., Oberdorfer, G. & Gruber, K. Flattening the curve—how to get better results with small deep-mutational-scanning datasets. *Proteins* **92**, 886–902 (2024).
113. Qiu, Y. & Wei, G. -W. Persistent spectral theory-guided protein engineering. *Nat. Comput. Sci.* **3**, 149–163 (2023).
114. Li, F. -Z. et al. Evaluation of machine learning-assisted directed evolution across diverse combinatorial landscapes. Preprint at *bioRxiv* https://doi.org/10.1101/2024.10.24.619774 (2024).
115. Bikman, M., Kolodny, R. & Osadchy, M. Few-shot prediction of the experimental functional measurements for proteins with single point mutations. *ICLR 2024 Workshop on Generative and Experimental Perspectives for Biomolecular Design* (2024).
116. Minot, M. & Reddy, S. T. Meta learning addresses noisy and under-labeled data in machine learning-guided antibody engineering. *Cell Syst.* **15**, 4–18 (2024).
117. Beck, J. et al. Metalic: meta-learning in-context with protein language models. Preprint at *arXiv* https://doi.org/10.48550/arXiv.2410.08355 (2024).
118. Zhao, J., Zhang, C. & Luo, Y. Contrastive fitness learning: reprogramming protein language models for low-N learning of protein fitness landscape. Preprint at *bioRxiv* https://doi.org/10.1101/2024.02.11.579859 (2024).
119. Tian, B. et al. Enhancing enzyme activity with mutation combinations guided by few-shot learning and causal inference. *Res. Sq.* https://doi.org/10.21203/rs.3.rs-5354708/v1 (2024).
120. Cranmer, K., Brehmer, J. & Louppe, G. The frontier of simulation-based inference. *Proc. Natl Acad. Sci. USA* **117**, 30055–30062 (2020).
121. Wu, Z. & Sinha, S. SPREd: a simulation-supervised neural network tool for gene regulatory network reconstruction. *Bioinform. Adv.* **4**, vbae011 (2024).
122. Ahmad, W., Simon, E., Chithrananda, S., Grand, G. & Ramsundar, B. ChemBERTa-2: towards chemical foundation models. Preprint at https://arxiv.org/abs/2209.01712 (2022).
123. Yu, S. et al. ClimSim: a large multi-scale dataset for hybrid physics-ML climate emulation. Preprint at https://arxiv.org/abs/2306.08754 (2023).
124. Eastman, P. et al. SPICE, a dataset of drug-like molecules and peptides for training machine learning potentials. *Sci. Data* **10**, 11 (2023).

Methods

Generating Rosetta pretraining data

The Rosetta pretraining data consist of protein sequences and their corresponding score terms, computed by modeling the sequences with Rosetta. We refer to the METL models pretrained on the Rosetta biophysical attributes as source models. The data used for local and global source models differ in what sequences are included. Rosetta data for local source models contain protein variants within the local sequence space surrounding the protein of interest. Rosetta data for global source models contain protein variants from a diverse range of base sequences and structures.

We generated local Rosetta datasets for each target protein from the experimental datasets (Supplementary Table 3). We acquired the target protein structures from the RCSB Protein Data Bank (PDB)[52] and AlphaFold Protein Structure Database[53]. For cases where the acquired structure did not match the reference sequence of the target protein, we used Rosetta comparative modeling or truncated the acquired structure to match the reference sequence. For each local pretraining dataset, we generated ~20 million protein sequence variants with a maximum of five amino acid substitutions. See Supplementary Table 7 for additional details regarding local Rosetta dataset structures and variants, including exceptions to the above.

We generated the global Rosetta dataset based on 150 diverse protein structures identified in Supplementary Table 1 of Kosciolek and Jones[14]. We downloaded the 150 structures from the RCSB PDB[52]. Some structures contained modified or missing residues. We replaced modified residues with canonical amino acids and used the RosettaRemodel application to fill in the structure of missing residues. We were unable to remodel PDB IDs 1J3A and 1JBE; thus, we excluded these structures from the final dataset. For each of the remaining 148 structures (Supplementary Table 2), we generated ~200,000 variants with a maximum of five amino acid substitutions, for a total of ~30 million variants.

To assess the similarity between the 148 proteins used for pretraining METL-Global and the 8 proteins used for evaluation, we clustered all proteins based on sequence and structure representations. We ran sequence-based clustering with MMseqs2 (v15.6f452)[125] and structure-based clustering with Foldseek (v9.427df8a)[126]. We set a coverage threshold of 0.5 for both sequence-based and structure-based clustering and identified clusters that contained structures from the METL-Global pretraining collection and one of the eight unique structures used for evaluation. At this threshold, both MMseqs2 and Foldseek identified structures in the METL-Global pretraining collection that are similar to the GRB2 and TEM-1 structures. Foldseek also detected a match for DLG4 and a second matching structure for GRB2. We aligned the pairs of co-clustered structures with the RCSB PDB pairwise structure alignment tool[127] and the TM-align alignment method[128].

We implemented a custom subvariants sampling algorithm to generate the variants for both the local and global datasets. The algorithm iteratively samples a random variant with five amino acid substitutions from the wild-type sequence and then generates all possible four-substitution, three-substitution, two-substitution and one-substitution subvariants with the same amino acid substitutions as the five-substitution variant. Duplicate variants generated through this process are discarded. The iterations terminate when the target number of variants is reached. For the global dataset, we used the subvariants sampling algorithm to generate all of the ~200,000 variants per base sequence. For the local datasets, we first generated all possible one-substitution or one-substitution and two-substitution variants, and then we used the subvariants sampling algorithm to generate the remainder of the ~20 million variants per target protein (Supplementary Table 7).

Once sequence variants were generated, we used Rosetta to compute biophysical attributes for each variant sequence. We first prepared each base PDB file for use with Rosetta by following the recommendation in the Rosetta documentation. We ran Rosetta's `clean_pdb.py` and relaxed the structure with all-heavy-atom constraints. We generated ten structures and selected the lowest energy structure to serve as the base structure for subsequent steps. We used Rosetta (v3.13)[11] to compute full-atom energy terms (ref2015 score function), centroid-atom energy terms (score3 score function) and custom filter terms based on Rocklin et al.[129]. For each variant, we introduced the variant's mutations to the corresponding base structure using a Rosetta resfile. Then, to generate the full-atom energy terms, we used FastRelax to relax the mutated structure using the ref2015 score function, only repacking residues within 10 Å of the mutated residues, with one repeat. To generate the centroid-atom energy terms, we used score_jd2 to score the resulting structure using the score3 score function. Finally, to generate the remainder of the score terms used in the standard version of METL, we used a RosettaScript to compute custom filter terms on the relaxed structure. To calculate additional binding scores for METL-Bind, we used the Rosetta InterfaceAnalyzer protocol. See Supplementary Tables 1 and 5 for a list and description of each term. We designed a computing workflow based on HTCondor[130] to orchestrate the Rosetta scoring on the Open Science Pool[42]. Rosetta simulation times scale with protein sequence length. Average runtimes per variant ranged from ~37–50 s for smaller proteins (56–102 residues, for example, GB1, GRB2, Pab1 and Ube4b) to 135–215 s for larger proteins (237–403 residues, for example, GFP, PTEN and TEM-1).

Preprocessing Rosetta pretraining data

Before training neural networks, we preprocessed the raw Rosetta data by dropping variants with NaN values for any of the biophysical attributes (343 in the global dataset and 645 in the GB1 dataset, corresponding to 0.0012% and 0.0051% of each dataset, respectively). No variants with NaN values were present in the other datasets. We also removed duplicates by randomly selecting one of the duplicates to keep and filtered out variants with outlier total_score values. We grouped variants by base PDB and removed outliers independently for each group using a modified *z*-score method, which uses the median and median absolute deviation instead of the mean and standard deviation. For each data point i, we calculated the modified z-score using equation (1):

$$s_i = \frac{|x_i - \tilde{x}|}{\mathrm{MAD}}, \tag{1}$$

where s_i is the modified z-score, x_i is the Rosetta total_score, $\tilde{x}$ is the median total_score of the group, and MAD is the median absolute deviation, defined as $\mathrm{MAD} = \mathrm{median}\,(|x_j - \tilde{x}|)\,\forall\, x_j \in \{x\}$, or the median of the absolute deviations of all data points from the median of the group. We removed variants with $s_i > 6.5$ from the dataset.

Additionally, we standardized the Rosetta scores to equalize the contribution of each score term to the model's loss function and to ensure score terms are comparable across different base PDBs in the global dataset. Once again, we grouped variants by base PDB, and then we standardized each group and score term independently by subtracting the mean and dividing by the standard deviation. We calculated the mean and standard deviation using only the training set data. This process scales the score terms to have zero mean and a standard deviation of one.

We excluded the following score terms from the final dataset because the values were zero for a large portion of base PDBs: dslf_fa13 (from ref2015 score function), linear_chainbreak and overlap_chainbreak (from score3 score function) and filter_total_score (custom filter term). We also discarded res_count_all (custom filter term that counts the residues in the protein) because it did not vary among variants of an individual protein. After these removals, 55 score terms remained (Supplementary Table 1).

METL source model architecture

The METL source model architecture accepts amino acid sequences as input and output predictions for each of the 55 Rosetta score terms.

The main component of the source model architecture is a transformer encoder based on the original transformer architecture[12], with the notable differences being the use of a relative positional embedding[13] instead of a sinusoidal positional encoding and pre-layer normalization instead of post-layer normalization[131]. METL-Local source models total ~2.5 million parameters and have transformer encoders consisting of a 256 embedding size, 3 encoder layers, 4 attention heads, a 1,024 feed-forward hidden size and 0.1 dropout. METL-Global source models total ~20 million parameters and have transformer encoders consisting of a 512 embedding size, 6 encoder layers, 8 attention heads, a 2,048 feed-forward hidden size and 0.1 dropout. We also evaluated a METL-Global source model with ~50 million parameters, consisting of a similar architecture as the 20-million-parameter METL-Global source model but with 16 encoder layers instead of 6 encoder layers. After the transformer encoder, source models implement an additional normalization layer, a global average pooling layer, a nonlinear fully connected layer and a linear output layer with 55 output nodes corresponding to the 55 Rosetta score terms. The global average pooling layer computes the mean of the per-residue encodings, which are output from the encoder, to produce a sequence-level representation of the same size as the embedding dimension. This sequence-level encoding is fed into a fully connected layer with 256 hidden nodes for the local model and 512 hidden nodes for the global model. We used the rectified linear unit activation function for the transformer encoder and final fully connected layer.

We implemented relative position embeddings as described by Shaw et al.[13]. In contrast to the absolute position encoding used in the original transformer architecture[12], the relative position embedding enables the network to consider positional representations of the inputs in terms of distances between sequence positions. We consider two distinct ways to encode relative distances, generating what we refer to as 1D positional embeddings and 3D positional embeddings. In the 1D approach, relative distances are based on the protein amino acid sequence alone. This approach is identical to the implementation of relative position embeddings described by Shaw et al. In the 3D approach, relative distances are based on the 3D protein structure.

In the 1D approach, we calculate relative distances by determining the offset between each pair of sequence positions (i,j) in the input. The relative distance is defined as $d = j - i$, representing how far sequence position j is relative to position i. A negative value signifies that j precedes i in the sequence, and a positive value signifies that j succeeds i. We map each of the possible relative distances to a pair of learnable embedding vectors, corresponding to attention keys and values. When calculating attention between sequence positions i and j, we add the key and value positional embedding vectors to the keys and values, respectively. As was hypothesized by Shaw et al., precise relative position information might not be useful beyond a certain distance. Thus, we clipped the possible relative distances to ±8.

In the 3D approach, we calculate relative distances using the protein 3D structure instead of the amino acid sequence. When using 3D relative position embeddings, the model requires a protein structure in the form of a PDB file, corresponding to the base protein that the input variant sequence is based on. We first represent the protein structure as an undirected graph, where each node corresponds to a residue. We place an edge between any pair of nodes if the beta carbon atoms (Cβ) of the residues are within 8 Å of each other in the 3D space. We define the relative distance between residues (i,j) as the minimum path length from node i to node j in the graph. Unlike the 1D approach, relative distances computed using the 3D approach cannot be negative values. We clip the 3D relative distances at 3, effectively transforming distances greater than 3 into a relative distance of 3. A relative distance of 0 represents a node with itself, 1 signifies direct neighbors, 2 signifies second degree neighbors and 3 encapsulates any other node not covered by the previous categories. As in the 1D approach, each possible relative distance in the 3D approach is mapped to a pair of embedding vectors corresponding to keys and values. These vectors are learned during training and are added to keys and values during the attention calculation.

METL source model training

We split the Rosetta source data into randomly sampled training, validation, test and withheld sets. For each dataset, we first withheld 5% of the data to be used for final evaluations. We split the remaining data into 80% training, 10% validation and 10% test sets.

We trained source models for 30 epochs using the AdamW optimizer[132] with a learning rate of 0.001. We applied a linear warm-up learning rate scheduler, with a warm-up period of 2% of the total training steps. Additional AdamW hyperparameters were weight_decay = 0.01, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1 \times 10^{-8}$. We computed mean squared error loss independently for each of the 55 prediction tasks (corresponding to the 55 Rosetta biophysical attributes) and took the sum to compute the final loss for the network. We applied gradient norm clipping with a maximum norm of 0.5. We used distributed data parallel training with four GPUs using PyTorch Lightning[133,134]. We trained local source models with an effective batch size of 2,048 (512 × 4 GPUs) and global source models with an effective batch size of 1,024 (256 × 4 GPUs). For the METL-Bind experiment, we trained both standard METL-Local and METL-Bind using the same process, except using two GPUs instead of four and a batch size of 1,024 instead of 512, which yielded an effective batch size 2,048, identical to the source models trained for the main experiment. METL-Bind was trained on 17 additional binding scores, for a total of 55 + 17 = 72 tasks, but was otherwise identical to the standard METL-Local model.

The global source data contain variants of 148 base sequences, with most having different sequence lengths. This complicates the process of encoding data into a single fixed-length batch. Padding is a commonly used approach in such scenarios. However, incorporating different sequence lengths and base structures in a single batch would negatively impact efficiency of computing attention with our implementation of relative position embeddings. Thus, we implemented a PDB-based data sampler that ensures each batch only contains variants from a single base PDB structure. Due to the use of distributed data parallel training with four GPUs, each aggregated training batch effectively contains variants from four base PDBs.

Pretraining times depend on the model size, protein size, amount of simulated data and computational resources. For a 2-million parameter METL-Local model with a simulated data size of ~20 million examples, running on 4× NVIDIA A100 or A30 GPUs, pretraining times ranged from 6 h to 13 h for 30 epochs (12 min to 26 min per epoch) for most proteins, with the exception of PTEN, which took ~33 h for 30 epochs (1.1 h per epoch). With a smaller training size of ~1 million examples and just a single GPU, training times ranged from 6 h to 26 h for 100 epochs for most proteins (4 min to 16 min per epoch). Pretraining METL-Global with 20 million parameters took ~50 h on 4× A100s and ~142 h with 50 million parameters.

Experimental datasets for target model training

The METL target model architecture accepts amino acid sequences as input and outputs predictions for one specific protein function. We evaluated METL on experimental datasets representing proteins of varying sizes, folds and functions: GFP[36], DLG4-2021 (ref. 135), DLG4-Abundance[136], DLG4-Binding[136], GB1 (ref. 26), GRB2-Abundance[136], GRB2-Binding[136], Pab1 (ref. 137), PTEN-Abundance[138], PTEN-Activity[139], TEM-1 (ref. 140) and Ube4b[141] (Supplementary Table 3). We acquired raw datasets from published manuscript supplements, MaveDB[142] and the NCBI Gene Expression Omnibus[143]. We transformed raw data into a standardized format, making sure that functional scores were log-transformed and normalized so that the wild-type score is 0, and rounded to seven decimal places. We removed variants with mutations to stop codons and converted variant indexing to be zero-based.

For DLG4-2021 and GB1, we filtered variants to ensure a minimum number of reads. See Supplementary Table 8 for additional details about dataset transformations. We opted to use the DLG4 dataset instead of the DLG4-2021 dataset in our main analysis due to weak correlation between the two datasets (Supplementary Fig. 24) and because linear regression yielded better results on the DLG4 dataset, suggesting a cleaner signal.

We used GB1 as an exploratory dataset during method development to make modeling decisions such as what size validation set is needed to enable model selection, where to place the prediction head on the source model, whether to use a linear or nonlinear prediction head, and others. Due to this, there is potential we overfit to GB1 and that our final results are optimistic for GB1. That said, we took precautions to limit the potential impact of using GB1 as our development dataset. The results presented for the small training set size experiment use an evaluation dataset that was completely held out, even during method development. The randomly sampled training and validation sets used to generate the final results are also different splits than the ones we used during method development. Additionally, the results presented for the extrapolation experiments use different splits than the ones we used to test extrapolation during method development.

We adjusted the GFP dataset preprocessing after seeing early small training set size results. Performance was lower than expected, which led us to realize that the dataset scores were not normalized so the wild-type score is 0. We modified the GFP dataset to normalize the scores and set the wild-type score to 0 by subtracting the wild-type score from all the scores. All other datasets were already normalized so the wild-type score is 0.

METL target model architecture

METL target models are made up of a backbone and a head. The backbone contains network layers from the METL source model, pretrained to predict Rosetta biophysical attributes. The head is a new, randomly initialized linear layer placed on top of the backbone to predict experimental functional scores. We also added a dropout layer with dropout rate of 0.5 between the backbone and the head. For METL-Local source models, we attach the head immediately after the final fully connected layer. For METL-Global source models, we attach the head immediately after the global pooling layer. METL target models have a single output node corresponding to the experimental functional score prediction.

METL target model training

We implemented two training strategies for PLM target models: feature extraction and fine-tuning. Feature extraction is a training strategy where only the head is trained, and the backbone weights are not updated during the training process. In contrast, fine-tuning is a training strategy where both the backbone and head weights are updated during training. For feature extraction, we trained the head using scikit-learn[144] ridge regression with alpha = 1.0 and the cholesky solver. This provides a closed-form solution for the ridge regression weights.

For fine-tuning, we implemented a dual-phase fine-tuning strategy[145]. In the first phase, we froze the backbone and trained only the head for 250 epochs. In the second phase, we trained both the backbone and the head for an additional 250 epochs at a reduced learning rate. We used the AdamW optimizer with a learning rate of 0.001 in the first phase and 0.0001 in the second phase. We applied a learning rate scheduler with linear warm-up and cosine decay to each phase, with a warm-up period of 1% of the total training steps. Additional AdamW hyperparameters were set as follows: weight_decay = 0.1, $\beta_1 = 0.9, \beta_2 = 0.999$ and $\epsilon = 1 \times 10^{-8}$. We used a batch size of 128 and mean squared error loss. We applied gradient norm clipping with a maximum norm of 0.5.

After the full training period, we selected the model from the epoch with the lowest validation set loss. We only performed model selection if the validation set size was ≥32 for METL-Local and ≥128 for METL-Global and ESM-2. We found the optimization was more stable for METL-Local than METL-Global and ESM-2; thus, smaller validation sets were still reliable. For validation sets smaller than those thresholds, we did not perform model selection. Instead, we used the model from the last epoch of training. We determined these thresholds using the GB1 dataset, which we designated as our development dataset, by selecting the dataset size along the learning curve where using model selection started to outperform not using model selection. In retrospect, these thresholds were too low for other datasets, leading to the dips in METL-Global correlations observed in Fig. 2.

Fine-tuning METL-Local was relatively quick, with training times scaling with the experimental dataset size. For a dataset size of 320 examples, fine-tuning typically took ~2–5 min; for 20,480 examples, it took ~20–42 min. Fine-tuning METL-Global (20 million parameters) took 7–45 min for small datasets (320 examples) and 40–150 min for large datasets (20,480 examples).

Target model dataset splits

We created comprehensive training, validation and test splits to evaluate performance with small training set sizes and a range of extrapolation tasks, including position, mutation, regime and score extrapolation. For small training set sizes, we first sampled a random 10% test set from each full dataset. Then, from the remaining data, we sampled datasets of sizes 10, 20, 40, 80, 160, 320, 640, 1,280, 2,560, 5,120, 10,240 and 20,480. To account for especially easy or difficult training sets that may be sampled by chance, we generated multiple replicates for each dataset size. The number of replicates decreases as the dataset size increases: 101 replicates for the smallest dataset size, followed by 23, 11, 11, 11, 11, 7, 7, 5, 5, 3 and 3 replicates for the largest dataset size. We split the sampled datasets into 80% training and 20% validation sets. We used the same test set across all dataset sizes and replicates. We report median performance metrics across replicates.

Whereas the small dataset splits are sampled randomly, the extrapolation splits are specially designed to assess the models' ability to generalize to more challenging test sets. For position, mutation and score extrapolation, we randomly resampled any datasets with >50,000 variants down to 50,000 variants before generating the extrapolation splits. To account for random effects, we generated nine replicate splits for each extrapolation type. We report the median across the nine replicates.

Position extrapolation tests the ability of a model to generalize to sequence positions not present in the training data. To generate position extrapolation splits, we first randomly designated 80% of sequence positions as training and the other 20% as testing. Then, we divided all variants (single-mutant and multi-mutant) into training and testing pools depending on whether the variants contain mutations only in positions designated as training or only in positions designated as testing. If a multi-mutant variant had mutations in both training and testing positions, we discarded it. To create the final training, validation and test sets, we split the training pool into randomly sampled 90% training and 10% validation sets. We used the entire test pool as the test set.

Mutation extrapolation tests the ability of a model to generalize to mutations not present in the training data. To generate mutation extrapolation splits, we followed a similar procedure as position extrapolation, except with mutations instead of sequence positions. We randomly designated 80% of mutations present in the dataset as training and the other 20% as testing. We divided all variants (single-mutant and multi-mutant) into training and testing pools depending on whether the variants contain only mutations designated as training or only designated as testing. If a multi-mutant variant had mutations that were designated as training and testing, we discarded it. We split the training pool into randomly sampled 90% training and 10% validation sets and used the entire test pool as the test set.

Regime extrapolation tests the ability of the model to generalize from lower numbers of amino acid substitutions to higher numbers of amino acid substitutions. For datasets with single and double

substitution variants, we divided the variants into a training pool comprising the single-substitution variants and a test pool comprising the double substitution variants. We split the training pool into an 80% training and a 20% validation set. We sampled a 10% test set from the test pool. For datasets containing greater than double substitution variants, we also implemented another regime extrapolation split where the training pool comprised single and double substitution variants and the test pool comprised variants with three or more substitutions.

Score extrapolation tests the ability of a model to generalize from low-scoring variants to high-scoring variants. We divided variants into training and testing pools depending on whether the variant had a score less than wild type (train pool) or greater than wild type (test pool). We split the training pool into a 90% training and a 10% validation set and used the entire test pool as the test set.

Baseline models

We implemented and evaluated additional baseline models: Linear, a fully connected neural network (FCN), a sequence convolutional neural network (CNN), METL-Local with random initialization, Rosetta's total score as a stand-alone prediction and linear regression with Rosetta total score (Linear-RTS).

'Linear' is a linear regression model that uses one-hot encoded sequences as inputs. One-hot encoding captures the specific amino acid at each sequence position. It consists of a length 21 vector where each position represents one of the possible amino acids or the stop codon. All positions are zero except the position of the amino acid being encoded, which is set to a value of one. Note that we removed variants containing mutations to the stop codon during dataset preprocessing, so this feature was not used in our analysis. We implemented linear regression using scikit-learn's ridge regression class, which incorporates L2 regularization. We set the solver to cholesky to calculate a closed-form solution for the ridge regression weights. We set alpha, the constant that controls regularization strength, to the default value of 1.0. We set all other parameters to the default scikit-learn values.

For baseline neural networks, we tested an FCN, a CNN and a transformer encoder with a similar architecture as METL-Local, but with a random initialization. The FCN and CNN used one-hot encoded sequences as input. The FCN consisted of one hidden layer with 1,024 nodes followed by a dropout layer with a dropout rate of 0.2. The CNN consisted of one convolutional layer with a kernel size of 7, 128 filters and zero-padding to ensure the output has the same shape as the input (padding mode 'same' in PyTorch's Conv2d class). Following the convolutional layer, we placed a fully connected layer with 256 nodes and a dropout layer with a dropout rate of 0.2. We used the rectified linear unit activation function for both models. In addition to the FCN and CNN, we tested a randomly initialized transformer encoder neural network with a similar architecture as METL-Local. Unlike METL-Local, this randomly initialized version was set up with a single output node corresponding to the experimental functional score instead of multiple output nodes corresponding to Rosetta scores.

We trained the FCN, CNN and randomly initialized METL-Local for 500 epochs using the AdamW optimizer with a base learning rate of 0.001. We applied a learning rate scheduler with linear warm-up and cosine decay, with a warm-up period of 2% of the total training steps. Additional AdamW hyperparameters were set as follows: weight_decay = 0.1, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1 \times 10^{-8}$. We used a batch size of 128 and mean squared error loss. We applied gradient norm clipping with a maximum norm of 0.5. Similarly to METL-Local fine-tuning, we selected the model from the epoch with the lowest validation loss when the validation set size was ≥32. Otherwise, we used the model from the last epoch of training.

We evaluated Rosetta's total score as a stand-alone, unsupervised prediction, as well as an additional input feature for linear regression, which we refer to as Linear-RTS. By default, the lower Rosetta's total score, the more stable the structure is predicted to be. Thus, when using Rosetta's total score as an unsupervised prediction, we multiplied it by −1 before computing correlation with the experimental functional score. We also tested Rosetta's total score as part of a supervised learning framework. Linear-RTS is identical to Linear, but it uses Rosetta's total score as an additional input feature in combination with the one-hot encoded sequence in an augmented regression setting[17]. We standardized the total score for use as an input feature by first calculating its mean and standard deviation in the training set. Then, we subtracted the mean and divided by the standard deviation.

Comparison to ESM-2

We used the ESM-2 (ref. 19) 35-million-parameter model with identifier `esm2_t12_35M_UR50D` as our default ESM model so that the comparisons with the 20-million-parameter METL-Global model would primarily emphasize their different pretraining strategies rather than model size. We incorporated several additional layers to match the METL architecture, including a global mean pooling layer, a dropout layer with dropout rate 0.5, and a linear prediction head. We attached these additional layers immediately after layer 12. We trained the ESM-2 models using the same training procedures we used for the METL models. We also explored feature extraction with larger 150-million-parameter and 650-million-parameter ESM-2 models (identifiers `esm2_t30_150M_UR50D` and `esm2_t33_650M_UR50D`). For these larger models, we attached the additional layers after layers 30 and 33, respectively.

Comparison to RaSP

We compared METL to RaSP[16] using the pretrained model weights for both the cavity model and downstream models shared by the authors on their GitHub repository (https://github.com/KULL-Centre/_2022_ML-ddG-Blaabjerg/). RaSP is a relevant comparison to METL since it is trained on $\Delta\Delta G$ values predicted using the Rosetta `cartesian_ddg` protocol[16]. Since RaSP is a point mutation stability predictor, it is not designed to handle mutants with multiple mutations (multi-mutants). After consulting with the authors, we adapted RaSP to handle multi-mutants by assuming an additive effect for each mutation in multi-mutants. As a result, we scored multi-mutants by scoring each point mutation individually and adding their scores. We used default parameters to extract the atomic environment for each mutant. After extracting the atomic environment, we used RaSP's cavity model to get a vectorized representation of the atomic environment followed by the ensemble of downstream models to predict the stability effect of a variant. We used RaSP in a zero-shot setting, that is, we did not fine-tune the model on examples from our target experimental dataset. Lastly, the authors note that RaSP is neither trained nor evaluated on disulfide-bonded cystine residues because they cannot be predicted using the Rosetta protocol used to generate RaSP's training data[16]. Since the goal of our evaluations was to test how well models predict protein functions measured by the various assays, we do not filter our test data based on this criterion.

Comparison to EVE

We obtained MSAs for GB1, Ube4b, GFP and Pab1 from the EVcouplings web server[146] in March 2023. We obtained MSAs for TEM-1, GRB2 and DLG4 in July 2023 and for PTEN in September 2024. We queried the UniRef100 database with search parameters consistent with those in EVMutation[147]: a position sequence filter of 70%, a sequence fragment filter of 50%, 100% removal of similar sequences and 80% down-weighting of similar sequences. We started our bitscore value at 0.5 bits per residue. If we did not have 80% sequence coverage, we increased the threshold by 0.05 bits per residue until the constraint was satisfied. If the number of effective sequences in the alignment was less than ten times the length of the protein, we decreased the bits per residue until the requirement was satisfied. We prioritized the number of effective sequences objective if the two were in conflict. We trained EVE using the default training parameters of 40,000 training iterations, sampling 20,000 evolutionary indices, and a default theta reweighting value of

0.2 to preprocess the MSA. We made mutation effect predictions for every position in the sequence by capitalizing all amino acids in the MSA.

In addition to using EVE as a stand-alone zero-shot method, we incorporated the EVE score into a supervised learning model. We selected EVE for augmented regression instead of the models evaluated by Hsu et al.[17] because EVE outperforms them in ProteinGym's zero-shot substitution deep mutational scanning evaluation[51], therefore providing a stronger baseline. The augmented regression model Linear-EVE is identical to the Linear model described above, but it uses the EVE score as an additional input feature in combination with the one-hot encoded protein sequence. We standardized the EVE score for use as an input feature by first calculating its mean and standard deviation in the training set. Then, we subtracted the mean and divided by the standard deviation.

Comparison to ProteinNPT

We ran the full ProteinNPT[18] pipeline, including the optional step of computing zero-shot fitness predictions with MSA Transformer[148] and incorporating them as auxiliary labels. The authors state this optional step helps improve performance, especially for position extrapolation. We followed the instructions from the ProteinNPT GitHub repository (https://github.com/OATML-Markslab/ProteinNPT/) and used the model configuration defined in `PNPT_final.json`. This configuration specifies the MSA Transformer model with identifier `esm_msa1b_t12_100M_UR50S` as the sequence embedding model and 10,000 total training steps. We used the same MSAs obtained from the EVCouplings web server that we used for EVE (see above).

Calculating predicted epistasis

For the GB1 epistasis analysis, we computed predicted epistasis using the pretrained METL-Local GB1 model. Let score (S) denote the model-predicted Rosetta total_score for variant S, and let wt_score denote the predicted total_score for the wild-type sequence. For each possible single and double variant, we first computed its effect relative to wild type: $w(S)$ = score(S) − wt_score. Then, we computed epistasis as given by equation (2):

$$E(S) = w(S) - \sum_{m \in S} w(m), \tag{2}$$

where m represents each single mutation in variant S. To compute pairwise positional epistasis, we calculated the mean absolute epistasis across all variants with mutations in the given pair of positions.

GFP sequence design

We fine-tuned a pretrained METL-Local model on 64 randomly sampled variants from the GFP dataset. The selected variants had 1 to 11 mutations, and their experimental score distribution was bimodal (Supplementary Fig. 19), similar to the distribution of the full GFP dataset. We refer to the fine-tuned METL-Local GFP model in this low-N setting as METL-L-GFP. We inspected the extrapolation behaviors of the METL-L-GFP model for increasing numbers of mutations. For increasing numbers of mutations selected with simulated annealing, the predicted brightness approximately stabilized at a positive value (Supplementary Fig. 25), in contrast to what has been observed in CNNs[149]. Conversely, for increasing numbers of randomly selected mutations, the predicted brightness stabilized at a negative value (Supplementary Fig. 26). That the predicted scores did not continue to increase positively or negatively with the number of mutations was a basic verification of the METL-L-GFP model's extrapolation properties.

We performed in silico optimization with METL-L-GFP to design a total of 20 variants distributed evenly across four different design criteria. These criteria are the product of two primary design categories: the number of mutations (either 5 or 10) and the constraints on mutation selection (either Observed or Unobserved). In the Observed constraint, the designed sequences contain only amino acid substitutions found in the 64-variant training set. Conversely, in the Unobserved constraint, the designed sequences exclude any amino acid substitutions found in the 64-variant training set. The combinations of these categories resulted in the four design criteria: Observed 5-mutant, Unobserved 5-mutant, Observed 10-mutant and Unobserved 10-mutant. We designed five sequences for each criterion, resulting in a total of 20 designed sequences.

To perform the in silico optimization, we ran simulated annealing 10,000 times for each design criterion. For each simulated annealing run, we changed the random seed and executed the Monte Carlo optimization for 10,000 steps. Each step consisted of suggesting a mutation for the currently sampled variant and deciding whether to accept the new variant according to the Metropolis–Hastings criteria. We decreased the optimization temperature according to a logarithmic gradient beginning at 10^{1} and ending at 10^{-2}. The initial temperature was chosen by randomly sampling 10,000 variants, predicting their brightness with METL-L-GFP, and calculating the absolute value of the difference between the lowest and highest predicted brightness, rounded to the nearest power of 10. The final temperature was determined by calculating the absolute value of the smallest difference in predicted brightness between any two variants in the 64-variant training set, rounded to the nearest power to 10. The initial temperature encouraged acceptance of all variants, while the final temperature meant that only variants better than the current ones would be accepted.

The simulation began by randomly selecting a variant with the necessary number of mutations depending on the design criterion. We determined how many mutations to change at each step by sampling from a Poisson distribution. To generate a new variant from an existing one, we first determined the difference between the number of mutations to change and the maximum allowable mutations, which indicated the number of current mutations to keep, m. We randomly sampled which m mutations to keep, and reset the other mutations to wild type. Subsequently, we compiled all feasible single mutations of the sequence with the m existing mutations and randomly sampled new mutations without replacement until the variant mutation count reached the maximum allowable mutations.

The optimization process described above yielded 10,000 designs for each criterion, which we downsampled to five designs for each criterion via clustering. Our downsampling approach prioritized diversity and was predicated on the idea that repeated convergence to similar sequences may correlate with higher true fitness values, as these regions of the fitness landscape would have broader peaks and allow more room for error in the model predictions or optimization process. We clustered the 10,000 designs using scikit-learn's agglomerative clustering with complete linkage and a BLOSUM62-based distance metric. Because selecting 10, 20 or 50 clusters did not substantially impact the diversity of the selected mutations, we chose 20 clusters. We then removed clusters that contained less than 100 sequences, which represented 1% of the simulated annealing solutions.

To select 5 (or 10) clusters from those remaining, we used an iterative, greedy approach. We identified a representative sequence for each cluster, choosing the one with the lowest average BLOSUM62-based distance to all other sequences within the same cluster. To initialize, we selected the largest cluster. We then proceeded iteratively, selecting additional clusters one at a time. In each iteration, we calculated the distances between the representative sequences of the already selected clusters and the remaining unselected clusters. We selected the cluster with the largest mean distance to the already selected clusters to promote sequence diversity. The GFP sequence designs were the representative sequences from the selected clusters.

To generate the baseline random GFP variants, we used two different random sampling algorithms corresponding to the different design criteria. For the Observed random variants, we randomly sampled individual mutations without replacement from the 209 unique mutations in the training set. For the Unobserved random variants, we randomly sampled individual mutations without replacement from all other all possible mutations excluding those 209 in the training set.

Cloning and experimental validation of GFP variants

We modeled our expression system on that used in Sarkysian et al.[36], which uses a pQE-30 vector (Qiagen) to express GFP as a fusion protein with mKate2. To generate the expression construct, we used the vector backbone from a related pQE-30 system that expresses KillerOrange (Addgene, 74748) and ordered a gene encoding the mKate2–GFP fusion protein from Twist Biosciences. We first removed a BsaI restriction site in the *ampR* gene from the backbone using site-directed mutagenesis (NEB, M0554S) and then used Golden Gate cloning to replace the KillerOrange gene with the fusion protein. We incubated (1 h, 37 °C) the backbone and insert with BsaI (15 U, NEB, R3733), T4 Ligase (1,000 U, NEB, M0202) and T4 Ligase Buffer (NEB, B0202) to assemble the vector. The assembly was cleaned up with a PCR Clean and Concentrate column (Zymogen, D4003) and transformed into in-house DH5a cells. Plasmid stock was purified from an overnight culture starting from a single colony using a Qiagen Miniprep kit (Qiagen, 27104), and the vector was on-boarded with Twist Biosciences. All GFP variants were ordered as clonal genes from Twist Biosciences wherein the wild-type *gfp* gene sequence was replaced with the variant sequence. For each variant, the nucleotide sequence was kept the same as the wild-type sequence except at mutated residues. We selected new codons for mutated residues based on an *Escherichia coli* codon usage index[150] to mitigate poor expression due to rare codons.

Clonal genes ordered from Twist Biosciences were transformed into NEBExpress Iq Competent *E. coli* (NEB, C3037I) cells and plated on Luria Broth plates with carbenecillin selection (0.1 mg ml^{-1}). Proteins were expressed as previously described in Sarkysian et al.[36]. Briefly, freshly plated transformants were incubated overnight at 37 °C and then moved to 4 °C the following day. After 24 h, plates were washed with 4 ml Luria Broth and normalized to 1 OD. This wash was used to create triplicate expression cultures where protein expression was induced for 2 h with 1 mM IPTG at 23 °C. An empty pQE-30 vector was used as a negative expression control.

To prepare cultures for fluorescence measurement, expression cultures were pelleted (3,000*g*, 5 min) and resuspended in sterile 1× PBS to a concentration of 1 OD. Cells were diluted twofold into 96-well plates to measure fluorescence and culture density. Measurements were taken with either the Tecan Spark 10M using the SparkControl 2.3 software or the Tecan Infinite M1000 Pro. Measurements for GFP (excitation 405 nm, emission 510 nm), mKate2 (excitation 561 nm, emission 670 nm) and OD600 (absorbance of 600 nm) were collected.

Relative brightness was reported as the ratio of GFP fluorescence to mKate2 fluorescence averaged across replicates. First, raw fluorescence measurements were normalized to cell density by dividing by the sample's OD_{600} value. The background fluorescence signal was subtracted from each sample. The background fluorescence signals for GFP and mKate2 were measured from negative control cells containing no fluorescent protein. A sample's relative brightness was calculated for each sample by dividing the normalized background-subtracted GFP fluorescence by the normalized background-subtracted mKate2 fluorescence. All fluorescence values were normalized to wild-type avGFP.

Visualizations

We used FreeSASA[151] to compute RSA, which was used to color the points in Extended Data Fig. 1d and Supplementary Figs. 15 and 16. We used MDS from scikit-learn to visualize GFP designs in Fig. 6b. MDS is a dimensionality reduction technique that preserves relative distances between observations[152]. We used Hamming distance between sequences, which had the effect of making variants show up in concentric circles around the wild-type sequence based on the number of mutations from wild type.

Unique biological materials

All unique biological materials are available upon request from the authors.

Reporting summary

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

Data availability

Pretrained METL models are available at https://doi.org/10.5281/zenodo.11051644 (ref. 153). Rosetta simulation datasets are available at https://doi.org/10.5281/zenodo.10967412 (ref. 154). Additional data are available in the GitHub repository (https://github.com/gitter-lab/metl-pub/), which is archived at https://doi.org/10.5281/zenodo.10819536 (ref. 155). The PDB structure identifiers used to train METL-Global are listed in Supplementary Table 2. The PDB and AlphaFold database structure identifiers used for METL-Local are listed in Supplementary Table 7. Experimental datasets used for model evaluation are listed in Supplementary Table 8 with references and corresponding file names.

Code availability

We provide a collection of METL software repositories to reproduce the results of this manuscript and run METL on new data. Code for pretraining and fine-tuning METL PLMs is available at https://github.com/gitter-lab/metl/. An archived version is available at https://doi.org/10.5281/zenodo.10819483 (ref. 156). Code for generating biophysical attributes with Rosetta is available at https://github.com/gitter-lab/metl-sim/. An archived version is available at https://doi.org/10.5281/zenodo.10819523 (ref. 157). Code for making predictions with pretrained METL PLMs is available at https://github.com/gitter-lab/metl-pretrained/. An archived version is available at https://doi.org/10.5281/zenodo.10819499 (ref. 158). Supporting code and data to reproduce this study is available at https://github.com/gitter-lab/metl-pub/. An archived version is available at https://doi.org/10.5281/zenodo.10819536 (ref. 155). A Hugging Face wrapper and demo for METL models is available at https://huggingface.co/gitter-lab/METL/. All code is available under the MIT license.

References

125. Steinegger, M. & Söding, J. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nat. Biotechnol.* **35**, 1026–1028 (2017).
126. van Kempen, M. et al. Fast and accurate protein structure search with Foldseek. *Nat. Biotechnol.* **42**, 243–246 (2024).
127. Bittrich, S., Segura, J., Duarte, J. M., Burley, S. K. & Rose, Y. RCSB Protein Data Bank: exploring protein 3D similarities via comprehensive structural alignments. *Bioinformatics* **40**, btae370 (2024).
128. Zhang, Y. & Skolnick, J. TM-align: a protein structure alignment algorithm based on the TM-score. *Nucleic Acids Res.* **33**, 2302–2309 (2005).
129. Rocklin, G. J. et al. Global analysis of protein folding using massively parallel design, synthesis, and testing. *Science* **357**, 168–175 (2017).
130. Thain, D., Tannenbaum, T. & Livny, M. Distributed computing in practice: the Condor experience. *Concurrency Comput. Pract. Exp.* **17**, 323–356 (2005).
131. Xiong, R. et al. On layer normalization in the transformer architecture. Preprint at https://arxiv.org/abs/2002.04745 (2020).
132. Loshchilov, I. & Hutter, F. Decoupled weight decay regularization. in *7th International Conference on Learning Representations* (2019).
133. Paszke, A. et al. PyTorch: an imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, vol. 32, 8024–8035 (Curran Associates, 2019).
134. Falcon, W. & The PyTorch Lightning team. PyTorch Lightning. *Zenodo* https://doi.org/10.5281/zenodo.3828935 (2022).
135. Nedrud, D., Coyote-Maestas, W. & Schmidt, D. A large-scale survey of pairwise epistasis reveals a mechanism for evolutionary expansion and specialization of PDZ domains. *Proteins* **89**, 899–914 (2021).

136. Faure, A. J. et al. Mapping the energetic and allosteric landscapes of protein binding domains. *Nature* **604**, 175–183 (2022).
137. Melamed, D., Young, D. L., Gamble, C. E., Miller, C. R. & Fields, S. Deep mutational scanning of an RRM domain of the *Saccharomyces cerevisiae* poly(A)-binding protein. *RNA* **19**, 1537–1551 (2013).
138. Matreyek, K. A., Stephany, J. J., Ahler, E. & Fowler, D. M. Integrating thousands of *PTEN* variant activity and abundance measurements reveals variant subgroups and new dominant negatives in cancers. *Genome Med.* **13**, 165 (2021).
139. Mighell, T. L., Evans-Dutson, S. & O'Roak, B. J. A saturation mutagenesis approach to understanding PTEN lipid phosphatase activity and genotype-phenotype relationships. *Am. J. Hum. Genet.* **102**, 943–955 (2018).
140. Gonzalez, C. E. & Ostermeier, M. Pervasive pairwise intragenic epistasis among sequential mutations in TEM-1 β-lactamase. *J. Mol. Biol.* **431**, 1981–1992 (2019).
141. Starita, L. M. et al. Activity-enhancing mutations in an E3 ubiquitin ligase identified by high-throughput mutagenesis. *Proc. Natl Acad. Sci. USA* **110**, E1263–E1272 (2013).
142. Esposito, D. et al. MaveDB: an open-source platform to distribute and interpret data from multiplexed assays of variant effect. *Genome Biol.* **20**, 223 (2019).
143. Barrett, T. et al. NCBI GEO: archive for functional genomics data sets—update. *Nucleic Acids Res.* **41**, D991–D995 (2013).
144. Pedregosa, F. et al. Scikit-learn: machine learning in Python. *J. Mach. Learning Res.* **12**, 2825–2830 (2011).
145. Kumar, A., Raghunathan, A., Jones, R. M., Ma, T. & Liang, P. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *10th International Conference on Learning Representations* (2022).
146. Hopf, T. A. et al. The EVcouplings Python framework for coevolutionary sequence analysis. *Bioinformatics* **35**, 1582–1584 (2019).
147. Hopf, T. A. et al. Mutation effects predicted from sequence co-variation. *Nat. Biotechnol.* **35**, 128–135 (2017).
148. Rao, R. M. et al. MSA Transformer. In *Proceedings of the 38th International Conference on Machine Learning*, 8844–8856 (PMLR, 2021).
149. Freschlin, C. R., Fahlberg, S. A., Heinzelman, P. & Romero, P. A. Neural network extrapolation to distant regions of the protein fitness landscape. *Nat. Commun.* **15**, 6405 (2024).
150. Boël, G. et al. Codon influence on protein expression in *E. coli* correlates with mRNA levels. *Nature* **529**, 358–363 (2016).
151. Mitternacht, S. FreeSASA: an open source C library for solvent accessible surface area calculations. *F1000Res.* https://doi.org/10.12688/f1000research.7931.1 (2016).
152. Borg, I. & Groenen, P. J. F. *Modern Multidimensional Scaling*. Springer Series in Statistics (Springer New York, 2005).
153. Gelman, S. et al. METL models. *Zenodo* https://doi.org/10.5281/zenodo.11051644 (2024).
154. Gelman, S. et al. METL Rosetta datasets. *Zenodo* https://doi.org/10.5281/zenodo.10967412 (2024).
155. Gelman, S. et al. Archived supporting code and data for METL. *Zenodo* https://doi.org/10.5281/zenodo.10819536 (2024).
156. Gelman, S. et al. Archived code for pretraining and finetuning METL. *Zenodo* https://doi.org/10.5281/zenodo.10819483 (2024).
157. Gelman, S. et al. Archived code for generating biophysical attributes with Rosetta. *Zenodo* https://doi.org/10.5281/zenodo.10819523 (2024).
158. Gelman, S. et al. Archived code for inference with pretrained METL models. *Zenodo* https://doi.org/10.5281/zenodo.10819499 (2024).
159. Center for High Throughput Computing. Center for High Throughput Computing. https://doi.org/10.21231/GNT1-HW21 (2006).
160. Sfiligoi, I. et al. The pilot way to grid resources using glideinWMS. in *2009 WRI World Congress on Computer Science and Information Engineering* Vol. 2, 428–432. https://doi.org/10.1109/CSIE.2009.950 (2009).
161. OSG. Open Science Pool. https://doi.org/10.21231/906P-4D78 (2006).

Acknowledgements

This research was supported by National Science Foundation awards 2226383 (to P.A.R.) and 2226451 (to A.G.), National Institutes of Health awards R01GM135631 (to A.G.) and R35GM119854 (to P.A.R.), the John W. and Jeanne M. Rowe Center for Research in Virology at the Morgridge Institute for Research, generous support from Jeanne M. Rowe, and the University of Wisconsin-Madison Office of the Vice Chancellor for Research and Graduate Education with funding from the Wisconsin Alumni Research Foundation. The funders had no role in study design, data collection and analysis, decision to publish or preparation of the manuscript. We thank B. Gelman for insightful discussions regarding the transformer architecture, attention mechanism and effects of data normalization; J. Duan for the initial molecular simulations implementation; B. Aydemir for testing and feedback on the molecular simulation workflow; and K. Amritkar for collecting and processing PDB files for METL-Global. The research was performed using the computing resources and assistance of the University of Wisconsin-Madison Center for High Throughput Computing[159] and services provided by the OSG Consortium[42,160,161], which is supported by National Science Foundation awards 2030508 and 1836650.

Author contributions

S.G., A.G. and P.A.R. conceived the original idea for the study. S.G. developed the METL framework and conducted the computational experiments and analyses. S.G., A.G., P.A.R., B.J. and A.S. contributed to discussions on computational experiment design and results interpretation. B.J. implemented the EVE baseline, generated GFP variant designs and created an interactive Jupyter notebook to run simulations. A.S. implemented the RaSP baseline. S.D. acquired and prepared PDB structures for METL-Global. J.P. developed the Hugging Face integration, interactive demo and Google Colab notebooks. C.R.F. designed and performed the wet lab experiments. A.G. and P.A.R. supervised the research. S.G., A.G., P.A.R., C.R.F., B.J. and A.S. wrote the manuscript. All authors reviewed and edited the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

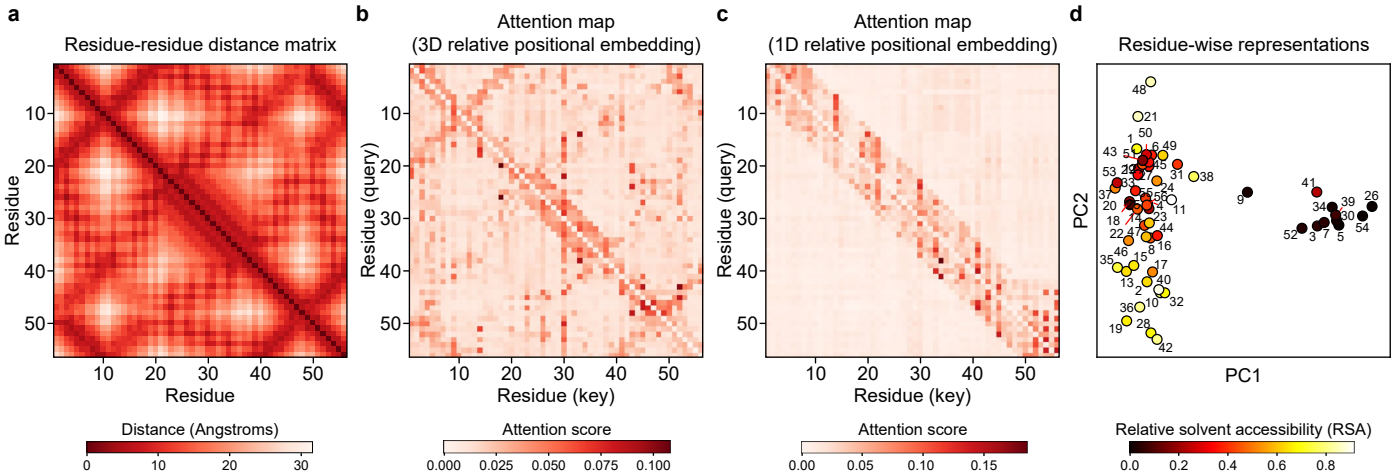
Extended data is available for this paper at https://doi.org/10.1038/s41592-025-02776-2.

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41592-025-02776-2.

Correspondence and requests for materials should be addressed to Anthony Gitter or Philip A. Romero.

Peer review information *Nature Methods* thanks Ava Amini, and the other, anonymous, reviewer(s) for their contribution to the peer review of this work. Primary Handling Editor: Arunima Singh, in collaboration with the *Nature Methods* team.

https://doi.org/10.1038/s41592-025-02776-2



Extended Data Fig. 1 | METL attention maps and residue representations relate to structure and biophysical properties. (a) The residue distance matrix shows Cβ distances between residues for the wild-type GB1 structure. (b-c) The attention maps show the mean attention across layers and attention heads for the wild-type GB1 sequence when it is fed as input to the pretrained GB1 METL-Local model. The 3D structure-based relative position embeddings (RPEs) enable the network to focus attention on residues that are close in 3D space, effectively capturing GB1's structural contacts. The 1D sequence-based RPEs do not. (d) Principal component analysis (PCA) of the residue representations output by the pretrained GB1 METL-Local model, averaged across the 20 possible amino acids at each sequence position. Points are colored according to relative solvent accessibility (RSA) computed from the wild-type GB1 structure.

nature portfolio

Corresponding author(s): Anthony Gitter and Philip Romero

Last updated by author(s): Apr 22, 2025

Reporting Summary

Nature Portfolio wishes to improve the reproducibility of the work that we publish. This form provides structure for consistency and transparency in reporting. For further information on Nature Portfolio policies, see our Editorial Policies and the Editorial Policy Checklist.

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: The experimental data was collected on a Tecan plate reader using the Magellan software.

Data analysis:

For running molecular simulations
Rosetta (3.13) - Modeling and analysis of protein structures
Python (3.9) - Primary programming language for all analysis
NumPy (1.19.5) and Pandas (1.2.1) - Numerical computations and data manipulation
Matplotlib (3.3.4) and Seaborn (0.11.1) - Plotting and data visualization
ShortUUID (1.0.1) - Assigning unique identifiers to computational runs
Biopython (1.78) - Manipulating protein structures
JupyterLab (3.0.12) - Interactive environment for method development and data analysis
SQLAlchemy (1.4.39) - SQL data management
OpenSSL (3.1.1) - Encrypting software

Custom code for running molecular simulations can be found at https://github.com/gitter-lab/metl-sim

For training and evaluating neural networks
Python (3.9.13) - Primary programming language for all analysis
PyTorch (1.12.1), PyTorch Lightning (1.7.7), TorchMetrics (0.10.2), and TorchExtractor (0.3.0) - Training and evaluating neural networks
NumPy (1.23.4), Pandas (2.0.3), SciPy (1.9.3) - Numerical computations and data manipulation
Scikit-learn (1.1.3) - Baseline linear regression models and utility functions
NetworkX (2.6.3) - Processing protein structures as graphs

Matplotlib (3.7.2), Seaborn (0.12.2), and AdjustText (0.8) - Plotting and visualizing data
UMAP-Learn (0.5.5) - Data dimensionality reduction and visualization
Jupyter (1.0.0), JupyterLab (4.1.2), and IPyWidgets (8.1.2) - Interactive environment for method development and data analysis
WandB (0.13.5) - Weights and biases experiment tracking
ShortUUID (1.0.1) - Assigning unique identifiers to computational runs
BioPandas (0.2.7) and Biopython (1.83) - Manipulating protein structures
SQLAlchemy (1.4.43), PyTables (3.7.0), and ConnectorX (0.3.2) - SQL data management
Fair-ESM (2.0.0) - Benchmarking ESM
MMseqs2 (15.6f452) and Foldseek (9.427df8a) - Sequence- and structure-based clustering
SparkControl (2.3) - Collect data from Tecan plate reader

Custom code for training and evaluating neural networks can be found at https://github.com/gitter-lab/metl and https://github.com/gitter-lab/metl-pretrained and https://github.com/gitter-lab/metl-pub

In addition to these software libraries, we note that numerous software dependencies were automatically installed by our package managers (Conda, Mamba, and pip — multiple versions), as part of the environment set up. These dependencies are important to the functionality of the primary software packages listed above.

For manuscripts utilizing custom algorithms or software that are central to the research but not yet described in published literature, software must be made available to editors and reviewers. We strongly encourage code deposition in a community repository (e.g. GitHub). See the Nature Portfolio guidelines for submitting code & software for further information.

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 description of any restrictions on data availability
- For clinical datasets or third party data, please ensure that the statement adheres to our policy

Pretrained METL models are available at doi:10.5281/zenodo.11051644. Rosetta simulation datasets are available at doi:10.5281/zenodo.10967412. Additional data is available in the GitHub repository https://github.com/gitter-lab/metl- pub, which is archived at doi:10.5281/zenodo.10819536. The PDB structure identifiers used to train METL-Global are listed in Supplementary Table 2. The PDB and AlphaFold DB structure identifiers used for METL-Local are listed in Supplementary Table 7. Experimental datasets used for model evaluation are listed in Supplementary Table 8 with references and corresponding filenames.

Human research participants

Policy information about studies involving human research participants and Sex and Gender in Research.

Reporting on sex and gender	n/a
Population characteristics	n/a
Recruitment	n/a
Ethics oversight	n/a

Note that full information on the approval of the study protocol must also be provided in the manuscript.

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	We selected 20 GFP variants for experimental testing to strike a balance between practical constraints—such as limits on DNA synthesis and screening capacity—and the need to evaluate a diverse range of sequences. This set size also allowed us to test a variety of design scenarios. For computational sampling, we chose sample sizes proportionate to the training set sizes. In cases with larger training sets, we observed low variance in outcomes, suggesting that our chosen sample sizes were sufficient.
Data exclusions	No data was excluded

Replication	The experimental findings were confirmed through three experimental replicates of independent protein expression samples.
Randomization	Randomization and blinding were not applicable in this study because all GFP variants were synthesized and tested under identical conditions in a single, controlled experiment. Since all samples were processed uniformly and measured at the same time, there was no risk of bias that randomization or blinding would typically mitigate.
Blinding	Randomization and blinding were not applicable in this study because all GFP variants were synthesized and tested under identical conditions in a single, controlled experiment. Since all samples were processed uniformly and measured at the same time, there was no risk of bias that randomization or blinding would typically mitigate.

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 and archaeology
☒	☐ Animals and other organisms
☒	☐ Clinical data
☒	☐ Dual use research of concern

Methods

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