Fine-tuning protein language models boosts predictions across diverse tasks

Robert Schmirler[1,2,3,*], Michael Heinzinger [1] & Burkhard Rost [1,4]

1 TUM (Technical University of Munich), School of Computation, Information and Technology (CIT), Faculty of Informatics, Chair of Bioinformatics & Computational Biology - i12, Boltzmannstr. 3, 85748 Garching/Munich, Germany
2 TUM Graduate School, Center of Doctoral Studies in Informatics and its Applications (CeDoSIA), Boltzmannstr. 11, 85748 Garching, Germany
3 AbbVie Deutschland GmbH & Co. KG, Innovation Center, BTS IR LU, Ludwigshafen, 67061, Germany
4 Institute for Advanced Study (TUM-IAS), Lichtenbergstr. 2a, 85748 Garching/Munich, Germany & TUM School of Life Sciences Weihenstephan (WZW), Alte Akademie 8, Freising, Germany
* Corresponding author: robert.schmirler@tum.de
Tel: +49-289-17-811 (email rost: assistant@rostlab.org)

Abstract

Prediction methods inputting embeddings from protein Language Models (pLMs) have reached or even surpassed state-of-the-art (SOTA) performance on many protein prediction tasks. In natural language processing (NLP) fine-tuning large Language Models (LLMs) has become the *de facto* standard. In contrast, most pLM-based protein predictions do not back-propagate to the pLM. Here, we compared the fine-tuning of three SOTA pLMs (ESM2, ProtT5, Ankh) on eight different tasks. Two results stood out. Firstly, task-specific supervised fine-tuning almost always improved downstream predictions. Secondly, parameter-efficient fine-tuning could reach similar improvements consuming substantially fewer resources at up to 4.5-fold acceleration of training over fine-tuning full models. Our results suggested to always try fine-tuning, in particular for problems with small datasets, such as for fitness landscape predictions of a single protein. For ease of adaptability, we provided easy-to-use notebooks to fine-tune all models used during this work for per-protein (pooling) and per-residue prediction tasks at https://github.com/RSchmirler/data-repo_plm-finetune-eval.

1. Introduction

How to speak protein? Transformer-based[1] language models (LMs) have revolutionized Natural Language Processing (NLP). Large language models (LLMs) now perform at or above average human level. The newest generative models (GPT4[2] or PaLM2[3]) expand upon NLP through impressive capabilities in coding, math, and even common sense reasoning[4]. The success of LLMs has led to their wide-spread application from computer vision[5] over time series forecasting[6] to biologic language models[7–13]. For instance, protein language models (pLMs) are trained on many protein sequences[14–17]. PLMs learn from large data sets without any experimental annotation other than the sequence. The information extracted by the pLM, more precisely the value describing the last hidden layers, dubbed the *embeddings*, can be readily transferred to any protein-related prediction task. This generality makes pLMs suitable to a wide variety of prediction tasks spanning from secondary structure[14] over membrane regions[18], intrinsic disorder[19], protein structure[16,20], and protein-protein interaction[21] to predictions of stability[22, 23] or solubility[24]. Successful applications to more function-related predictions include the identification of paratopes[25], epitopes[26], and signal peptides[27], as well as, other tasks, e.g., related to the effect of sequence variation[28–31]. Embedding-based predictions seem particularly advantageous when experimental data are very limited[32]. Effectively, the embeddings from pLMs condense the understanding of the language of life[7,14]. Over the last 30 years, the *de facto* standard in protein prediction has been the use of evolutionary information, i.e., of information from Multiple Sequence Alignments (MSAs) as input to machine learning[33]. Now, pLM-based predictions have reached and often even superseded the MSA-based state-of-the-art (SOTA) expert devices for many prediction tasks. Embeddings can be input to artificial feed-forward (ANN) or convolutional neural networks (CNN). More complex architectures have also been explored[34]. Continued unsupervised training can focus models on specific protein families[35] or enrich embeddings with structural information essentially creating a bi-lingual pLM[36]. Training specialist models from scratch on smaller, specific proteins, e.g., antibodies[25,37], seems an alternative to "continued training" (train pLM on large generic data and refine on specific proteins).

Here, we evaluated the impact of task-specific supervised fine-tuning. We added a simple ANN as *prediction head* on top of the pLM encoder and applied supervised training to the model (pLM encoder and prediction head). We compared the results to predictions using pre-trained embeddings (training the prediction head using pre-computed embeddings). For the larger models (ProtT5[14], ProstT5[36], both Ankh[15] models, and ESM2 3B[16]), we utilized Low Rank Adaptation (LoRA)[38], a particular version of a more general approach known as PEFT (Parameter Efficient Fine-Tuning)[39]. Freezing most of the model and updating only a small fraction of the weights (Table 3) accelerates training and prevents *catastrophic forgetting*[40,41]. This will become especially relevant, as pLMs[17] follow the NLP trend where bigger models usually translate to better downstream performance[17]. For pLMs, fine-tuning remains less studied than for NLP[39,42,43], although some prediction tasks have been reported to profit from supervised, task-specific fine-tuning[21,44–46]. We assessed diverse prediction tasks from eight previously established benchmarks through the lens of three pLMs.

2. Results and Discussion

Fine-tuning mostly successful. We trained 615 individual prediction methods (295 for fine-tuning, 320 using frozen embeddings from pre-trained pLMs - protein Language Models) comprising eight models (Table 1), each trained on eight different data sets (Table 2). We trained each model-task combination multiple times with different random seeds and all results constituted averages over those runs. The corresponding validation set selected the best training stop. For each prediction task (PT), we compared the performance between fine-tuning and pre-training (Eqn. 1). For ProtT5-XL-U50[14] (labeled ProtT5) and all five tested ESM2 versions[16] (differing in parameter size between 8M (8*10^6) and 3B (3*10^9)), not all improvements were statistically significant within the 95% confidence interval (CI: Methods). Nevertheless, supervised fine-tuning numerically increased performance for almost all combinations (Fig. 1, detailed results in supplementary online material (SOM) Tables S1-S6). The exceptions were ESM2-150M applied to *Stability* prediction, and both Ankh[15] models. Ankh gained significantly by fine-tuning only for the mutational landscape data (GFP, AAV and GB1: green in Fig. 1).

For these data, performance relied less on transfer from model pretraining (Fig. S7) and mainly depended on the underlying transformer architecture. This might explain why *Ankh* performed similar to *ProtT5* and the *ESM2*. For the diverse data sets this was not the case. Two major factors differentiate *Ankh* form the other pLMs. Firstly, the T5[47] masked span pre-training differs from that for BERT-like[48] objective used for the other models. Secondly, the training procedure and architecture of Ankh was optimized using data (GFP, GB1, subcellular location, and secondary structure) also utilized in this work[15]. This might have reduced the ability to fine-tune these models.

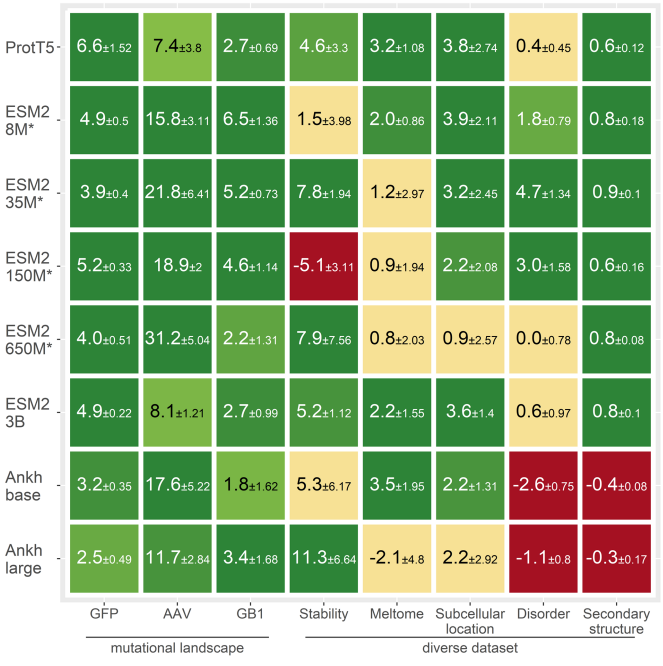

Fig. 1: Fine-tuning improved for most pLMs and tasks. Asterisks (*) mark fully fine-tuned models; the others were LoRA-optimized (SOM Fig. S1). Values reflect percentage differences between the fine-tuned and pre-trained models (Eqn. 1) for the eight prediction tasks (x-axis). We had to use different performance measures, namely the Spearman rank correlation ("GFP", "AAV", "GB1", "stability", "meltome" and "disorder"), 10-class accuracy (Q10: "sub-cellular location"), and 3-class per-residue accuracy (Q3: "secondary structure"). Each tile compares fine-tuning to raw embeddings for one task. Green tiles mark statistically significant increases (>1.96 standard errors; fine-tuning better), yellow tiles mark statistically insignificant changes (0 lies within the error margins of ±1.96 stderr) and for red tiles supervised fine-tuning significantly decreased performance. Error estimates (±percentage values) represent the 95% confidence intervals (CI, Methods).

For five of the 64 pLM/task combinations (tiles in Fig. 1), fine-tuning performed worse. The observation *ESM2-150M* on *stability* (Fig. 1 red tile) originated from instability in training picking

a suboptimal model (Fig. S5). The other four originated from the *Ankh* pLM family on *disorder* and *secondary structure*. We were not able to track down a root cause here but suspect that the different nature of the pre-training plays a role.

LoRA was competitive to alternative PEFT methods. For ProtT5 and sub-cellular location prediction, we compared three parameter efficient fine-tuning methods to LoRA[38]. Not having sufficient resources to do this analysis for all prediction tasks/pLMs, we chose this problem due to its limit in size and because the success of fine-tuning on this problem (configuration in Method and Fig. 2). The fraction of trained model parameters were 0.25% for LoRA, 0.28% for DoRA[49], 0.12% for IA3[50] and 0.5% for Prefix tuning[51]. Despite these differences, runtimes for training and testing (inference) were within ±10% between methods, with the exception of DoRA which was about 30% slower than the other three. In terms of prediction performance, LoRA and DoRA outperformed IA3 and Prefix-tuning (Fig. 2). Overall, all fine-tuning methods improved, on average, over pretrained embeddings (61.3% from Table S5). As no method improved significantly over the well-established LoRA, we used it throughout our experiments. Of course, these results for a single model and dataset must not hold true in general. We encourage to explore parameter efficient fine-tuning of pLMs, utilizing new combinations of high-quality datasets, state of the art models and PEFT methods in future work and hope the notebooks made available by us help to pursue this research more easily.

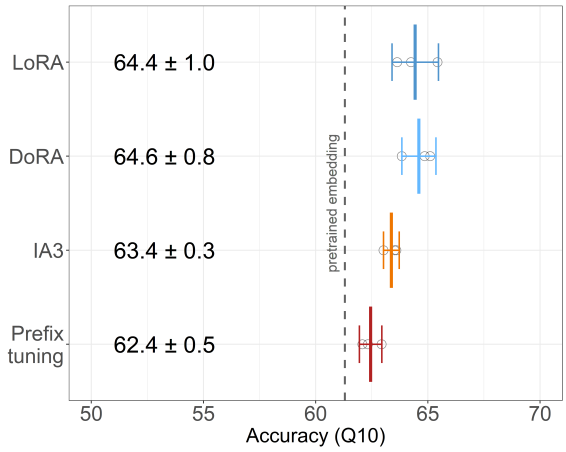

Fig. 2: Comparison of different PEFT methods. ProtT5 model assessed on the prediction of sub-cellular location (x-axis: 10-state per-protein accuracy Q10). Mean values as well as 95% confidence intervals are computed from three training re-runs for each of the four PEFT methods: LoRA[38], DoRA[49], IA3[50], and Prefix-tuning[51]. We used the same configuration for LoRA and DoRA. The IA3 target modules were the key, value and feed-forward layers. Prefix-tuning used 20 virtual tokens with 1024 dimensions to fit the ProtT5 dimensions. Circles represent individual training results. Differences between methods are mostly insignificant, with all four numerically out-performing the pre-trained embedding predictor on average (dashed grey line).

Insignificant gain for secondary structure prediction. For per-residue, three-class secondary structure prediction (helix, strand, other), fine-tuning improved only slightly (Fig. 2a; up to 1.2 percentage points for *CASP12* [52] and *NEW364* [14]). We confirmed this for the general-purpose ProtT5[14] and for the bilingual, structure-tuned ProstT5[36]. Two effects might have hindered substantial improvement. Firstly, secondary structure might already have been captured in unsupervised pre-training. In fact, embeddings already capture some aspects of inter-residue contact formation[10,20]. Secondly, performance may have reached an upper limit[53]. One limitation of the benchmark is highlighted by the two data sets (*CASP12* [52] and *NEW364* [14]). Both were introduced to gauge performance for unknown proteins. Other than that *CASP12* is much smaller (12 proteins vs. 364) implying higher statistical errors, there seems no *a priori* reason for choosing one over the other, and no method compared here is expected to have any systematic bias toward

any of the two. Thus, the difference between both should estimate the statistical error. In other words, bootstrapping error estimates should be similar to the difference between the two sets. This was not the case at all (Fig. 3a: differences between *CASP12* and *NEW364* exceeded standard errors marked by distributions). Arguably, secondary structure prediction assessment is the best solved task in protein structure prediction since decades[33,53]. Even for this relatively trivial problem, such a simple dichotomy seems not easily resolvable. In fact, the standard mantra: larger data sets without redundancy appears not to solve this dichotomy. These results underscore how difficult it is to just plug in standard data sets to assess the performance of prediction methods without updating data and adapting it to advancing methods.

Fine-tuning boosted disorder prediction. PLM-based SETH[19] reached the level of MSA-based SOTA methods, such as ODiNPred[54] in the prediction of per-residue protein disorder as described by CheZOD scores[54]. SETH inputs ProtT5 embeddings into a two-layer CNN.

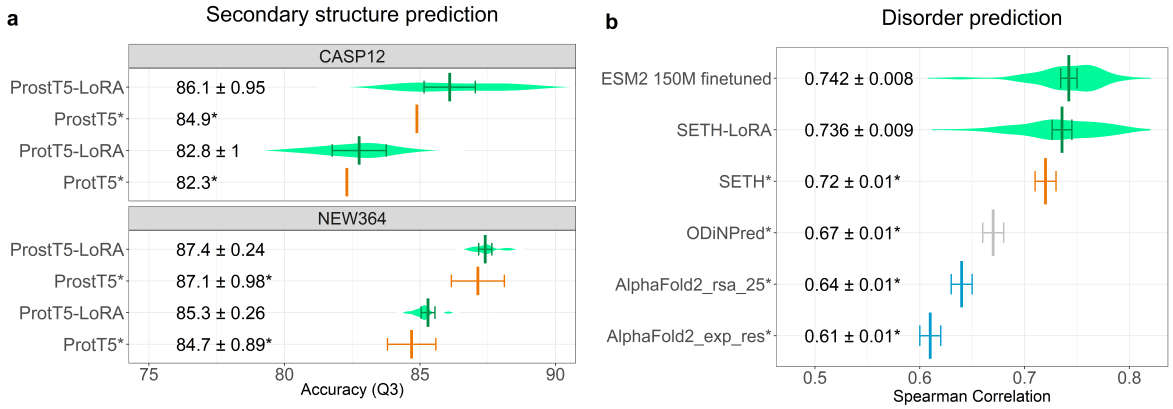

Fig. 3: Disorder prediction better, secondary structure prediction not. Error bars mark the 95% confidence intervals (CI), estimated through bootstrapping (not available for CASP12), violin plots reflect the data distribution. **(a)** Values for the pre-trained models (ProtT5[14] and ProstT5[36]) taken from literature[36] and marked by asterisk (*); fine-tuning in green, pre-trained embeddings in orange. We included two previously used data sets (CASP12[52] and NEW364[14]) to highlight the limitation of benchmarks. **(b)** Intrinsically disordered residues can be proxied by CheZOD scores[54]. The x-axis shows the Spearman correlation between experimental and predicted CheZOD scores for six methods. Values marked by asterisks (*) taken from the literature[19]. Fine-tuning results in green, pLM-based without MSA (SETH[19]) in orange, MSA-based SOTA in gray[54,55], and MSA-based AlphaFold2[56] in blue.

Keeping those hyper-parameters and adding LoRA fine-tuning (dubbed SETH-LoRA), improved performance by 2.2 percentage points (from Spearman 0.72 to 0.736, Fig. 3b). Fine-tuning the much smaller 150M parameter ESM2 model (Spearman: 0.742) improved over all solutions compared (Fig. 3b), including its larger counterparts (ESM2 with 650M / 3B parameters, Table S4). Compared to SETH-LoRA where only 2.5 million out of its 1.2 billion parameters are trained, for ESM2-150M all parameters were fine-tuned. Both approaches (2.5m for ProtT5 vs 150m for ESM2) performed similarly (Fig. 3b).

LoRA topped pooling for subcellular location. Most predictions of subcellular location input signals averaged over entire proteins (e.g., amino acid composition). Embedding-based solutions do this through pooling, i.e., through embeddings derived from averaging over all intrinsic residue-level embeddings[14]. Light Attention (LA) substantially improves over such coarse-grained

averaging by learning the optimal per-residue signal and combining this with the average[34]. LoRA fine-tuning combined the advantage of a small model (fewer free parameters) with the learned, weighted averaging of LA. Thereby, LoRA fine-tuning numerically surpassed LA, although the difference was statistically significant only at an 88% confidence interval (CI and not at the more common CI=95% Table S9).

Fine-tuning better captured effects of mutations. For predicting mutation landscapes (Fig. 1 leftmost three columns) fine-tuning any pLM succeeded substantially. As differences between fine-tuned models were small (Fig. S3), we averaged performance across all fine-tuned pLMs (Fig. 4), and compared to homology-based inference (HBI, using MMseqs2[57] search) and to reference-free analysis (RFA[58]). RFA fit a decent first-order model for the fitness landscape reflecting some mutations for GB1 (protein G domain B1[59]; all possible variants for four residues, i.e., at four positions). For AAV2[60] (adeno-associated virus 2) for which a much larger 28-residue window was mutated, RFA performed less well. For GFP (green fluorescent protein[61]) the RFA analyses failed, because some specific substitutions XnY (amino acid X at position n mutated to Y) occurred only in the test set. The fact that smaller and larger models performed alike on these tasks raised the prospect of using small, fine-tuned pLMs as computationally affordable, high-quality solutions for protein engineering.

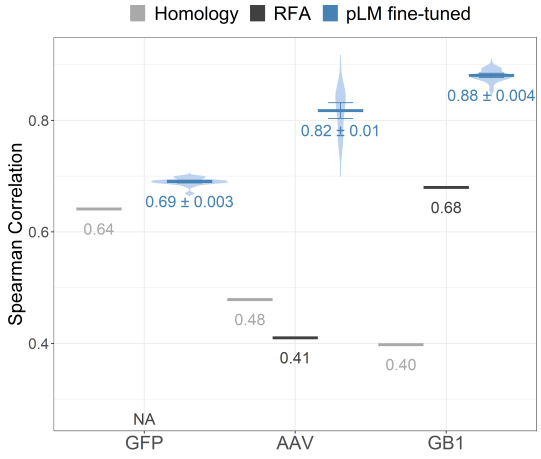

Fig. 4: Simple methods limited for mutational effects. Blue: average performance across all fine-tuned pLMs (error bars: 95% CI) with violin plots providing the underlying distribution; Gray: two simple baseline methods: *Homology* (HBI): MMseqs2[53] inferred the fitness value of test set proteins from most sequence-similar homolog in the training set. *RFA* (reference-free analysis[54]) fits models based on the assumption, that the most of the mutational effect can be described as the sum of low-order effects.

LoRA was substantially faster for larger models. The main drivers for the amount of computational resources required for model training were the parameter sizes of pLMs along with quadratic scaling of the attention mechanism (more resources for longer proteins). More recent GPUs used for LLM training (anything beyond 40GB of memory) will have sufficient memory to allow usage of all pLMs tested here. For less powerful hardware (Fig. 5b), mixed precision training nearly halved the required GPU memory without performance loss (both Ankh models were exceptional, as they do not support mixed precision training). Where GPU memory still was a bottle-neck, we applied gradient accumulation to reduce the actual on-device batch size as far as needed. When even an on-device batch size of 1 was insufficient, we used DeepSpeed to offload the optimizer and potentially parameters to CPU reduced GPU memory requirements further. As a trade-off, both gradient accumulation and CPU offloading slowed down training. Hence, both should be used cautiously. Implementing all these measures, we could fine-tune most pLMs tested

here even with on older GPUs with as little as 8GB memory (Fig. 5b). Unintuitively, both full model fine-tuning and parameter-efficient LoRA fine-tuning required the same amount of GPU memory and only differed in training speed (Fig. 5a) when CPU offloading was utilized. Embedding creation required much less GPU memory rendering it feasible even for datasets with very long sequences (Fig. S1).

Fine-tuning recipe. To ease the simplicity of fine-tuning pLMs for your data set, we added the following recommendations. Before starting model training, dataset splits to measure model generalization and prevent over-estimating performance[23,28] are essential. First off: you need at least three data sets: training (optimizing weights), cross-training/validation (optimization of hyper-parameters, e.g., to decide between CNN and ANN), and testing (only touched to estimate performance). Typically, all entities in the test set (e.g. proteins) should adhere to the same split required between training/validation and testing. In particular, proteins have to be made non-redundant. This requires clustering by sequence identity using standard alignment methods such as MMseqs2[57] (simpler solutions tend to lead more likely to information leakage). For structure-related tasks, redundancy is best removed through 3D clustering as realized by Foldseek[62]. To optimize the prediction of mutational landscapes for a single protein, it might be best to train on k-mers with k=1 (single amino acid variants) and test on k-mers with k>1[22,23] (although this approach might focus more on avoiding over-fitting than on generating the best optimal model).

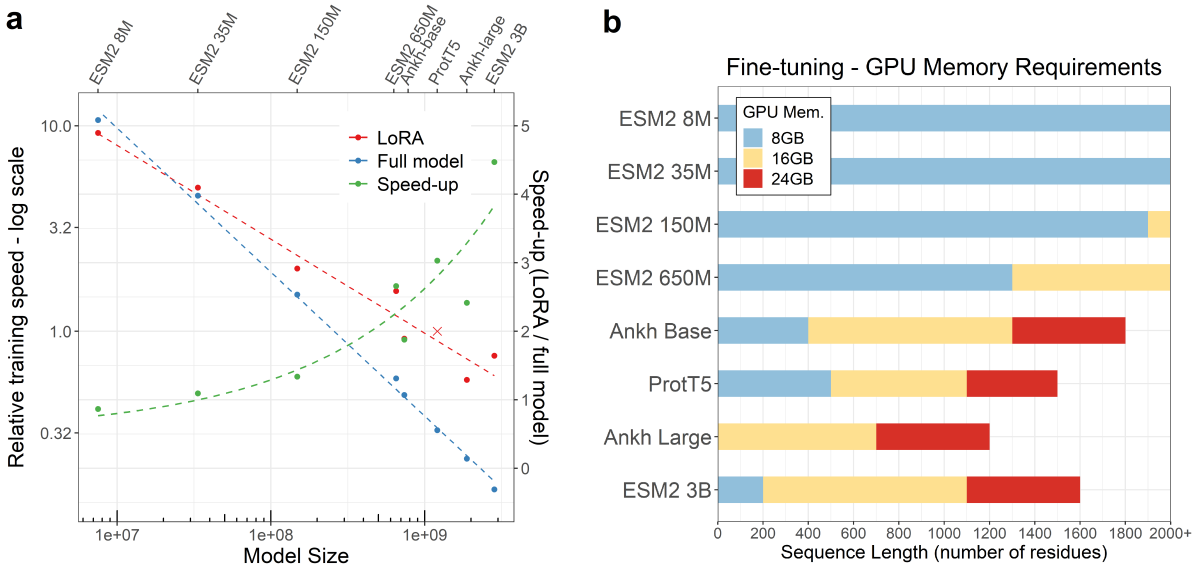

Fig. 5: Fine-tuning training speed and GPU requirements. (a) Relative training speed of full fine-tuning (blue) and LoRA (red) is shown on a logarithmic scale, ProtT5 LoRA fine-tuning served as reference speed with value of 1 (x). The resulting speed-up for each model (green) is shown on a normal scale. Experiments were performed with arbitrary sequences of length 1024 in a per-protein setting. For the smallest model (ESM2 8M), LoRA fine-tuning was marginally slower than training the entire model. The larger the model, the more advantageous LoRA became. For the largest model (ESM2 3B), LoRA was about a 4.5-fold faster. Panel (b) shows the maximum sequence length before the GPU runs out of memory (for 8, 18, and 24GB GPUs). All values obtained for memory-efficient training (mixed precision training, gradient accumulation with on-device batch size 1 and DeepSpeed CPU offloading). Experiments were done for per-protein predictions, but memory requirements for per-residue training will be similar. Results valid for full-model and LoRA fine-tuning.

To predict landscapes of mutational effects for specific proteins, a challenge encountered in protein engineering, we recommend to first fine-tune a smaller pLM (pre-trained embeddings were limited: Fig. S3). Optimize hyperparameters and head architectures on this smaller model. If done, you could explore additional improvements from larger pLMs. For the fine-tuning on diverse tasks, larger mostly out-performed smaller models (Fig. S3 & S4). Therefore, starting with raw embedding-based solutions to identify the best model and to then investigate different prediction heads appeared better than optimizing the fine-tuning directly. Applying parameter efficient LoRA fine-tuning and optimizing hyperparameters for the selected model afterwards, will probably lead to an even better solution.

For our tasks, over-fitting mostly originated from data set characteristics. On the one hand, given a data set prone to over-fitting (e.g. too small, uninformative or complex), neither hyperparameter nor model optimization could fully avoid the trap. On the other hand, for data sets not prone to over-fitting the training of fine-tuning was stable regardless of other factors. These factors affected raw embedding-based and fine-tuned models alike. Avoiding imbalanced datasets, providing sufficient high-quality training data, and choosing smaller models for limited data sets could mitigate over-fitting (SOM Sections 9 and 10).

On the computational side, we recommend *mixed precision training*. Gradient accumulation and DeepSpeed's CPU-offloading should only be reserved to avoiding memory constraints. With all these measures in place, a single 16 GB GPU enables fine-tuning in many cases (Fig. 5). Comparing different common PEFT methods (Fig. 2) did not suggest a clear winner. The established LoRA[38] method was among the best solutions and was stable across our experiments. The codebase provided by us simplifies experimenting with different PEFT approaches as it utilizes the Hugging Face PEFT[63] framework. We encourage you to compare different PEFT methods for your specific use cases. PEFT is memory efficient, but CPU-offloading could achieve the same. However, PEFT is also most compute-efficient for larger pLMs (Fig. 5a); it stabilizes training (SOM Section 5) and it renders saving model checkpoints orders of magnitude more memory efficient, as only the trained parameters need to be stored. Thus we recommend LoRA fine-tuning for all models larger than ESM2 150M (Fig. S2). We see little reason not to fully fine-tune smaller models.

In our hands, different random initialization seeds significantly altered results. These random variations reached the magnitude effect of hyperparameters or even model selection.

3. Conclusions

We applied fine-tuning[38, 39, 42,43] to a diversity of prediction tasks (spanning from the per-protein level subcellular location and thermo-stability to the per-residue level secondary structure and disorder) gauging success by comparing the performance of feeding embeddings from protein Language Models (pLMs: ProtT5[14], Ankh[15], ESM2[16]) into subsequent supervised prediction methods with task-specific supervised fine-tuning of the pLMs. Fine-tuning clearly improved performance, on average. The extent of this improvement varied by task and pLM/model. We could not afford to optimize all possible hyper-parameters for all model-task combinations. Instead, we compared fine-tuning to highly optimized raw embedding-based methods from previous work[14,19,34,36] (Fig. 3, Table S9). On average, LoRA fine-tuning improved for all prediction tasks. For per-protein predictions of location, the LoRA optimization may have learned weighted pooling of the last hidden layer, and that improved significantly over average pooling[34]. More

generally, the last hidden layer has been optimized for the un-supervised pre-training objective (learning to reproduce masked sequences). This optimization might be suboptimal for any downstream task[64,65]. PEFT (or finetuning in general) enables information from middle layers to flow to the last layer, making it accessible to downstream tasks. Lastly, the transformer models might extract additional information directly from the task-specific training. Randomly initialized smaller ESM2 models supported this view (Fig. S7, Table S17). While for our applications, fine-tuning proved relatively robust against hyperparameter changes, the gains from fine-tuning were impacted, by the amount of training data (Fig. S10), dataset balance (Table S8), models size (Fig. S3) and initial representation quality (Fig. S4). Overall, our results revealed the gains initially observed in NLP from supervised task-specific fine-tuning of LLMs[42,43] to also apply to large protein LMs (pLMs). Therefore, we suggest to add supervised fine-tuning whenever applying transfer-learning, i.e., when inputting pLM embeddings into subsequent supervised prediction tasks. Our results suggested that you will most often benefit from this. To ease this additional step, we provided all our resources and added step-by-step recommendations.

4. Methods

4.1. Pre-trained pLMs

The pLMs used differed in size and architecture (Table 1), ranging from 8 million (ESM2 8M) to 3 billion parameters (ESM2 3B). ESM2 pLMs are RoBERTa[66] based encoder models trained using an unsupervised masked language modeling objective[16]. The other three pLMs are built on T5[47], an encoder-decoder LLM pre-trained applying span masking[14,15]. We initialized our models using the pre-trained checkpoints available on Huggingface.

Table 1: Protein language models (pLMs) applied in study *

Model	*Architecture (pretraining)*	*Number of parameters (encoder)*	*Trained parameters LoRA*	*Encoder layers*	*Emb size*	*Huggingface model checkpoint*
Ankh Base	Encoder-Decoder	736 M	2,100 K	48	768	ankh-base
Ankh Large		1,900 M	4,900 K	48	1536	ankh-large
ProtT5		1,200 M	3,500 K	24	1024	prot_t5_xl_uniref50
ProstT5		1,200 M	3,500 K	24	1024	ProstT5
ESM2 8M	Encoder	8 M	163 K	6	320	esm2_t6_8M_UR50D
ESM2 35M		35 M	483 K	12	480	esm2_t12_35M_UR50D
ESM2 150M		150 M	1,600 K	30	640	esm2_t30_150M_UR50D
ESM2 650M		650 M	3,500 K	33	1280	esm2_t33_650M_UR50D
ESM2 3B		3,000 M	7,700 K	36	2560	esm2_t36_3B_UR50D

* *Emb size* provides the dimension of the embeddings of the corresponding pLM. Throughout the paper, we used the standard acronyms K for 10^3, M for 10^6, and G for 10^9.

4.2. Data

The data sets differentiated two aspects: *prediction task* level and *sequence diversity* (Table 2). The *prediction task* level collected cases of per-residue (e.g. secondary structure) and per-protein (e.g. sub-cellular location) prediction. *Sequence diversity* distinguished between data sets with many sequence-diverse proteins and those from mutational studies analyzing single proteins through deep mutational scanning (DMS also known as MAVE) experiments[67].

Mutational landscapes: these data described fitness landscapes for three individual proteins: the green fluorescent protein (GFP[61]), the adeno-associated virus 2 capsid protein VP-1 (AAV2[60]) and the first binding domain of the protein G (GB1[59]). All three constitute regression tasks that measure prediction performance by ranking the correlation between the predicted and the experimentally measured property for each set. For *GFP* the property/fitness was measured through fluorescence intensity (experimental data[68], data split[22]). Training and validation set sequences were all within Hamming distance 3 (i.e. all variants up to three changes from wild type). For the *AAV* task, fitness was measured as the viability for packaging DNA payloads by mutating 28-amino acid window[69]. We used the "2-vs-rest" data split from the FLIP benchmark[23]

(variants with ≤2 in training and validation sets, those with more in test set). The *GB1* fitness score measures stability and binding affinity. The original experiment[70] mutated four positions we took the „three-vs-rest" data split from FLIP.

Table 2: Task-specific datasets *

Prediction level	*Sequence diversity*	*Prediction task / data set*	*Number of sequences*			*Average length (number of residues/tokens)*
			Train	*Validation*	*Test*	
per-protein	Mutational landscapes	GFP[22,68]	21,446	5,362	27,217	237.0
		AAV[23,69]	28,626	3,181	50,776	736.3
		GB1[23,70]	2,691	299	5,743	265.0
	Diverse datasets	Stability[22,71]	53,614	2,512	12,851	45.0
		Meltome[23,72]	22,335	2,482	3,134	544.5
		SubCellLoc[34,73]	9,503	1,678	490	519.9
per-residue		Disorder[19,54]	1,056	118	117	118.1
		SecStr[14,74]	9,712	1,080	364	255.0

* Prediction level: *Per-protein* predictions make a single prediction for an entire protein; *per-residue* predictions provide one number for each residue (position) in a protein. Sequence diversity: distinguishes between tasks with experimental data specific for individual proteins (mutational landscapes) and those for which data mixes different proteins from different organisms. Prediction task: as described in Methods; *SubCellLoc*: sub-cellular location, *SecStr*: secondary structure prediction (in 3 states: helix, strand, other). Number of sequences (i.e., proteins, note this is NOT the number of samples, e.g., for secondary structure prediction N proteins – number given in table - correspond to over 200*N residues in the data set): typical cross-validation using Train to optimize fine-tuning, Validation to optimize hyperparameters, and Test only to assess performance (Methods).

Per-protein prediction tasks included three prediction tasks. The first two sets focused on stability prediction formulated as regression tasks in analogy to the fitness landscapes. *Stability* predictions were assessed on measurements of protease susceptibility to digestion by *de novo* designed mini-proteins[71]. We reused the TAPE data split[22] in which training and validation sets contain sequences from four design cycles while the test set holds neighborhoods at 1-Hamming distance to 17 promising candidates. *Meltome* utilizes data from measuring thermostability for proteins from 13 species[72]. We used the "mixed" split from the FLIP benchmark[23], which clusters proteins at >20% pairwise sequence identity (PIDE) through MMseqs2[57]. Excluding any pair within the same cluster from test and train/validation set, is the minimal means to reducing potential information leakage. The third data set included DeepLoc[73] data (incl. training/validation/testing splits) along with a novel test data set (*setHARD* [34]). The task is to predict sub-cellular location in one of ten classes. MMseqs2[57] removed all sequence pairs at >20 PIDE between training, validation, and test sets.

Per-residue prediction tasks included disorder and secondary structure. *Disorder predictions* used CheZOD data[54] from nuclear magnetic resonance (NMR) spectroscopy. We used previously published data splits[19] clustering at <20 PIDE with MMseqs2[57]. The task is to predict CheZOD

scores[55] which quantify the level of intrinsic disorder for each residue in a protein (each amino acid position) through a continuous scale. We bench-marked *secondary structure predictions* through data sets provided by *NetSurfP-2.0* [74] distinguishing three classes (H: helix, E: strand, and C: other/non-regular). In splitting these data, we used a recent split with more stringent redundancy-reduction, along with another test set (NEW364)[14].

Performance measures were copied from the data set developers (Table 2). All regression tasks were evaluated using Spearman rank correlations, i.e., all three mutational landscapes (GFP, AAV, GB1), both stability related data sets (*Stability* and *Meltome*), as well as, the per-residue regression of *Disorder*. For the classification tasks, accuracy was defined as a 10-class per-protein accuracy (Q10) for sub-cellular location, and as a 3-class per-residue accuracy (Q3) for secondary structure (for a detailed per class analysis see SOM Section 2).

4.3. Model training

Top-level comparisons contrasted pre-trained to fine-tuned models as follows. For the pre-trained results, we generated embeddings for all data sets. For per-protein tasks, we averaged over the sequence length; for each protein, this yielded a vector of dimension 1 x embedding_size. For per-residue tasks, we used all residue embeddings and their labels; this resulted in vectors of the same dimension as for the per-protein task (1 x embedding_size), albeit this time for each residue (i.e., L*embedding_size for a protein of length L). Next, we trained a single fully connected layer of size 32, inputting exclusively these embeddings and outputting either a single value (regression) or going into another output layer with one neuron for each possible output class, followed by a softmax layer to get a probability distribution. Reaching a plateau in training loss terminated training. We repeated each training five times with different random seeds. For fine-tuning, we added the same fully connected layer (size 32) to the pLM encoder as a prediction head. For ProtT5[14] and Ankh[15], we used average pooling of the last hidden states over the sequence length dimension during training on per-protein tasks. Following the author's advice for ESM2[16], we connected the prediction head only to the very first token (special token "<CLS>"). Training was repeated three times with different random seeds. Training terminated when training and validation loss flattened out. To increase training efficiency and to save time, we applied *Parameter Efficient Fine-Tuning* (PEFT[39]) to all models. In particular, we applied the Low-Rank Adaptation (LoRA[38]) implementation of PEFT. For the smaller ESM2 models (up to the 650M version), we could investigate full model fine-tuning (SOM Section 3 and Fig. S1).

For embedding and fine-tuning we reported the performance of the checkpoint with the lowest validation loss for each run. Finally, we simply computed the percentage differences between the fine-tuned and pre-trained models (Eqn.1):

$$\Delta(PT) = performance(PT)_{finetuned} - performance(PT)_{pretrained} \quad \text{(Eqn. 1)}$$

We also performed a limited hyperparameter optimization at the beginning (Table S11). Once selected, the hyperparameters were frozen for most comparisons (Tables S10 and S12). We used the Adam optimizer[75] with default parameters. For LoRA fine-tuning we reused a previously suggested configuration[50], namely rank 4, alpha 1, applied to query, key, value and the output of the attention layers. A minimal LoRA rank of four has also been suggested previously[21] for pLMs.

To realize the batches size (Table S12) for fine-tuning, we applied gradient accumulation as needed given our hardware. Initially, we fine-tuned models on *full precision*, but switched to *mixed precision* for larger models. Performance did not drop from this (data not shown). Embedding generation used *half precision*. The Ankh models only support full precision for both. All training ran on a single NVIDIA A10G GPU with 24GB. We used Torch version 1.13.1 with transformers version 4.26.1.

Training times for fine-tuning depended crucially on the available GPU. Nevertheless, the following basic trends were notable. Training times were mostly driven by model and dataset size, as well as, the quadratic scaling with sequence length. As all these factors accumulate, smaller models and shorter sequences led to much faster training (Fig. 5a). For instance, for the small *Disorder* data (average sequence length: 118 residues), fine-tuning the full ESM2-8M took 12 minutes, while the much larger ProtT5 (LoRA fine-tuning) took 190 minutes (16-fold increase). On the other end, for the AAV data (average length: 736 residues with nearly 30k proteins), ESM2-8M training took 130 minutes while a single ProtT5 training ran over 4K minutes (>30-fold increase). The time to train an embedding-based prediction method was mostly determined by the time needed to compute embeddings because the relatively small predictor models that we used required negligible runtime. The creation of embeddings took approximately as long as fine-tuning the same model for a single epoch (typically we needed 5-50 epochs, Table S12).

Per-residue secondary structure: We fine-tuned five models for ProtT5[14] and another five for ProstT5[36], initializing with different random seeds. For ProstT5, we added the prefix token *<AA2fold>* to each sequence to code the input type (amino acid rather than structure as ProstT5 is a bilingual pLM). We trained for five epochs and calculated the validation loss at the end. For both models (ProtT5 and ProstT5), utilizing the same two-layer CNN as applied previously[14,36] for pre-trained embeddings to simplify comparisons. Of the five, we selected the model with lowest validation loss and measured performance on common data sets (CASP12[52] and NEW364[14]). Bootstrapping established confidence intervals (Fig. 2a).

Per-residue disorder fine-tuning stacked up two model variants to compare to other methods[19]. *SETH-LoRA* used ProtT5 with the same two-layer CNN as the original SETH. *ESM2 150M* reused the ESM2 setup from the top level evaluation (last hidden states of *<CLS>* token with single dense layer prediction head). For both variants, we trained five models with different random seeds for ten epochs, and calculated the validation loss twice per epoch, selecting the model with lowest validation loss out of 100 checkpoints (5 random seeds, 10 epochs, 2 points per epoch). Bootstrapping provided confidence intervals (Fig. 3b).

Per-protein subcellular location: For the 10-class classification, we reused the single layer dense network from our top level evaluation. We trained five models with different random seeds for five epochs, calculating validation loss twice per epoch. We selected the model with the lowest validation loss in each run, and then calculated Q10 accuracy and standard errors from all five models on *setHARD* [34] and reported the averages (Table S9).

Fitness landscapes contrasted fine-tuned pLMs to two baselines, namely, *homology-based inference* (HBI) and *reference-free analysis* (RFA)[58]. We averaged test performance over all fine-tuned pLMs (checkpoints with lowest loss on validation set) for the three mutational landscape data sets (GFP, AAV, GB1).

HBI: MMSeqs2[57] searched each query (Q) in the test set against all proteins training set proteins. The fitness value of the top training hit (closest homolog) predicted that for Q.

RFA: We applied the R implementation of RFA without modification to the GB1 data. For AAV, we removed sequences not containing 735 residues (algorithm failed on insertions and deletions). This reduced the training data by about 1%. RFA failed for GFP because some substitutions in the test set were missing from the training set. For AAV and GB1, we fitted a first- and second-order model and reported results for the better of the two.

Data availability

All data sets analyzed are freely available through the original sources:
• GFP and stability: https://github.com/songlab-cal/tape
• AAV, GB1, meltome and secondary structure: https://github.com/J-SNACKKB/FLIP
• Sub-cellular location: https://github.com/HannesStark/protein-localization
• Disorder: https://github.com/DagmarIlz/SETH
Easing access, we re-packaged all data at https://github.com/RSchmirler/data-repo_plm- finetune-eval. When using those data, please quote and consult the authors of the original data sets. All data generated for this study and source data to generate figures and tables is also available in this repository.

Code availability

We made the notebooks for fine-tuning all our models (Table 1) for per-protein and per-residue tasks available at https://github.com/RSchmirler/data-repo_plm-finetune-eval. By default, these notebooks use LoRA fine-tuning; full-model fine-tuning is optional. To create prediction methods based on pre-trained pLM embeddings, we provided two additional notebooks (one to generate embeddings, another to train predictors) with sample data in the same repository.

RFA R scripts are available from https://github.com/whatdoidohaha/RFA.

References

1. Vaswani, A. et al. Attention is all you need. Adv. neural information processing systems 30 (2017).
2. OpenAI. GPT-4 Technical Report. Preprint at https://arxiv.org/abs/2303.08774 (2023).
3. Anil, R. et al. PaLM 2 Technical Report. Preprint at https://arxiv.org/abs/2305.10403 (2023).
4. Bubeck, S. et al. Sparks of Artificial General Intelligence: Early experiments with GPT-4. Preprint at https://arxiv.org/abs/2303.12712 (2023).
5. Liu, Z. et al. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. Proc. IEEE/CVF international conference on computer vision 10012–10022 (2021).
6. Zhou, H. et al. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. Proc. AAAI conference on artificial intelligence 35, 11106–11115 (2021).
7. Heinzinger, M. et al. Modeling aspects of the language of life through transfer-learning protein sequences. BMC bioinformatics 20, 1–17 (2019).
8. 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).
9. Bepler, T. & Berger, B. Learning protein sequence embeddings using information from structure. Preprint at https://arxiv.org/abs/1902.08661 (2019).
10. Rives, A. et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. Proc. Natl. Acad. Sci. 118, e2016239118 (2021).
11. Trotter, M. V., Nguyen, C. Q., Young, S., Woodruff, R. T. & Branson, K. M. Epigenomic language models powered by Cerebras. Preprint at https://arxiv.org/abs/2112.07571 (2021).
12. Dalla-Torre, H. et al. The Nucleotide Transformer: Building and Evaluating Robust Foundation Models for Human Genomics. Preprint at https://www.biorxiv.org/content/10.1101/2023.01.11.523679v3 (2023).
13. Yang, F. et al. scBERT as a large-scale pretrained deep language model for cell type annotation of single-cell RNA-seq data. Nat. Mach. Intell. 4, 852–866, DOI: 10.1038/s42256-022-00534-z (2022).
14. Elnaggar, A. et al. ProtTrans: Toward Understanding the Language of Life Through Self-Supervised Learning. IEEE Transactions on Pattern Analysis Mach. Intell. 44, 7112–7127, DOI: 10.1109/TPAMI.2021.3095381 (2022).
15. Elnaggar, A. et al. Ankh: Optimized Protein Language Model Unlocks General-Purpose Modelling. Preprint at https://arxiv.org/abs/2301.06568 (2023).
16. Lin, Z. et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. Science 379, 1123–1130 (2023).
17. Chen, B. et al. xTrimoPGLM: Unified 100B-Scale Pre-trained Transformer for Deciphering the Language of Protein. Preprint at https://www.biorxiv.org/content/10.1101/2023.07.05.547496v3 (2023).
18. Bernhofer, M. & Rost, B. TMbed: transmembrane proteins predicted through language model embeddings. BMC bioinformatics 23, 326 (2022).
19. Ilzhöfer, D., Heinzinger, M. & Rost, B. SETH predicts nuances of residue disorder from protein embeddings. Front. Bioinforma. 2 (2022).
20. Weißenow, K., Heinzinger, M. & Rost, B. Protein language-model embeddings for fast, accurate, and alignment-free protein structure prediction. Structure 30, 1169–1177 (2022).
21. Sledzieski, S. et al. Democratizing Protein Language Models with Parameter-Efficient Fine-Tuning. Preprint at https://www.biorxiv.org/content/10.1101/2023.11.09.566187v1 (2023).
22. Rao, R. et al. Evaluating Protein Transfer Learning with TAPE. Adv. neural information processing systems 32, 9689–9701 (2019).
23. Dallago, C. et al. FLIP: Benchmark tasks in fitness landscape inference for proteins. Preprint at https://www.biorxiv.org/content/10.1101/2021.11.09.467890v2 (2021).
24. Feng, J., Jiang, M., Shih, J. & Chai, Q. Antibody apparent solubility prediction from sequence by transfer learning. Iscience 25 (2022).

25. Leem, J., Mitchell, L. S., Farmery, J. H., Barton, J. & Galson, J. D. Deciphering the language of antibodies using self-supervised learning. Patterns 3 (2022).
26. Clifford, J. N. et al. BepiPred-3.0: Improved B-cell epitope prediction using protein language models. Protein Sci. 31, e4497 (2022).
27. Teufel, F. et al. SignalP 6.0 predicts all five types of signal peptides using protein language models. Nat. biotechnology 40, 1023–1025 (2022).
28. Groth, P. M., Michael, R., Salomon, J., Tian, P. & Boomsma, W. FLOP: Tasks for Fitness Landscapes Of Protein wildtypes. Preprint at https://www.biorxiv.org/content/10.1101/2023.06.21.545880v1 (2023).
29. Marquet, C. et al. Embeddings from protein language models predict conservation and variant effects. Hum. Genet. 141, 1629–1647, DOI: 10.1007/s00439-021-02411-y (2022).
30. Nijkamp, E., Ruffolo, J., Weinstein, E. N., Naik, N. & Madani, A. ProGen2: Exploring the Boundaries of Protein Language Models. Preprint at https://arxiv.org/abs/2206.13517 (2022).
31. Notin, P. et al. Tranception: protein fitness prediction with autoregressive transformers and inference-time retrieval. Int. Conf. on Mach. Learn. 16990–17017 (2022).
32. Littmann, M., Heinzinger, M., Dallago, C., Weissenow, K. & Rost, B. Protein embeddings and deep learning predict binding residues for various ligand classes. Sci. Reports 11, 23916 (2021).
33. Rost, B. & Sander, C. Prediction of protein secondary structure at better than 70% accuracy. J. molecular biology 232, 584–599 (1993).
34. Stärk, H., Dallago, C., Heinzinger, M. & Rost, B. Light attention predicts protein location from the language of life. Bioinforma. Adv. 1, vbab035, DOI: 10.1093/bioadv/vbab035 (2021).
35. Madani, A. et al. Large language models generate functional protein sequences across diverse families. Nat. Biotechnol. 1–8 (2023).
36. Heinzinger, M. et al. Bilingual Language Model for Protein Sequence and Structure. Preprint at https://www.biorxiv.org/content/10.1101/2023.07.23.550085v2 (2023).
37. Olsen, T. H., Moal, I. H. & Deane, C. M. AbLang: an antibody language model for completing antibody sequences. Bioinforma. Adv. 2, vbac046 (2022).
38. Hu, E. J. et al. LoRA: Low-Rank Adaptation of Large Language Models. Preprint at https://arxiv.org/abs/2106.09685 (2021).
39. Ding, N. et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. Nat. Mach. Intell. 5, 220–235 (2023).
40. French, R. M. Catastrophic forgetting in connectionist networks. Trends cognitive sciences 3, 128–135 (1999).
41. Kirkpatrick, J. et al. Overcoming catastrophic forgetting in neural networks. Proc. national academy sciences 114, 3521–3526 (2017).
42. Howard, J. & Ruder, S. Universal Language Model Fine-tuning for Text Classification. In Proc. 56th Annu. Meet. Assoc. for Comput. Linguist. 328–339, DOI: 10.18653/v1/P18- 1031 (2018).
43. Wortsman, M. et al. Robust fine-tuning of zero-shot models. Proc. IEEE/CVF Conf. on Comput. Vis. Pattern Recognit. 7959–7971 (2022).
44. Thumuluri, V. et al. NetSolP: predicting protein solubility in Escherichia coli using language models. Bioinformatics 38, 941–946 (2022).
45. Wang, D., Fei, Y. E. & Zhou, H. On Pre-training Language Model for Antibody. The Eleventh Int. Conf. on Learn. Represent. (2022).
46. Dumitrescu, A. et al. TSignal: a transformer model for signal peptide prediction. Bioinformatics 39, i347–i356 (2023).
47. Raffel, C. et al. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. The J. Mach. Learn. Res. 21, 5485–5551 (2020).
48. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Preprint at https://arxiv.org/abs/1810.04805 (2018).

49. Liu, S.-Y. et al. DoRA: Weight-Decomposed Low-Rank Adaptation. Preprint at https://arxiv.org/abs/2402.09353 (2024).
50. Liu, H. et al. Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning. Adv. Neural Inf. Process. Syst. 35, 1950–1965 (2022).
51. Li, X. L. & Liang, P. Prefix-Tuning: Optimizing Continuous Prompts for Generation. Preprint at https://arxiv.org/abs/2101.00190 (2021).
52. Abriata, L. A., Tamò, G. E., Monastyrskyy, B., Kryshtafovych, A. & Dal Peraro, M. Assessment of hard target modeling in CASP12 reveals an emerging role of alignment-based contact prediction methods. Proteins: Struct. Funct. Bioinforma. 86, 97–112 (2018).
53. Rost, B., Sander, C. & Schneider, R. Redefining the goals of protein secondary structure prediction. J. molecular biology 235, 13–26 (1994).
54. Dass, R., Mulder, F. A. & Nielsen, J. T. ODiNPred: comprehensive prediction of protein order and disorder. Sci. Reports 10, 14780 (2020).
55. Nielsen, J. T. & Mulder, F. A. Quantitative Protein Disorder Assessment Using NMR Chemical Shifts. Intrinsically Disord. proteins: methods protocols 303–317 (2020).
56. Jumper, J. et al. Highly accurate protein structure prediction with AlphaFold. Nature 596, 583–589 (2021).
57. Steinegger, M. & Söding, J. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. Nat. biotechnology 35, 1026–1028 (2017).
58. Park, Y., Metzger, B. P. & Thornton, J. W. The simplicity of protein sequence-function relationships. Preprint at https://www.biorxiv.org/content/10.1101/2023.09.02.556057v1 (2023).
59. Sauer-Eriksson, A. E., Kleywegt, G. J., Uhlén, M. & Jones, T. A. Crystal structure of the C2 fragment of streptococcal protein G in complex with the Fc domain of human IgG. Structure 3, 265–278 (1995).
60. Russell, S. et al. Efficacy and safety of voretigene neparvovec (AAV2-hRPE65v2) in patients with RPE65-mediated inherited retinal dystrophy: a randomised, controlled, open-label, phase 3 trial. The Lancet 390, 849–860 (2017).
61. Chalfie, M., Tu, Y., Euskirchen, G., Ward, W. W. & Prasher, D. C. Green fluorescent protein as a marker for gene expression. Science 263, 802–805 (1994).
62. van Kempen, M. et al. Fast and accurate protein structure search with foldseek. Nat. Biotechnol. 1–4 (2023).
63. Mangrulkar, S. et al. PEFT: State-of-the-art Parameter-Efficient Fine-Tuning methods. https://github.com/huggingface/peft (2022).
64. Li, F.-Z., Amini, A. P., Yang, K. K. & Lu, A. X. Pretrained protein language model transfer learning: is the final layer representation what we want. Proc. Mach. Learn. for Struct. Biol. Work. NeurIPS 2022 (2022).
65. Valeriani, L. et al. The geometry of hidden representations of large transformer models. Preprint at https://arxiv.org/abs/2302.00294 (2023).
66. Liu, Y. et al. RoBERTa: A Robustly Optimized BERT Pretraining Approach. Preprint at https://arxiv.org/abs/1907.11692 (2019).
67. Fowler, D. M. & Fields, S. Deep mutational scanning: a new style of protein science. Nat. methods 11, 801–807 (2014).
68. Sarkisyan, K. S. et al. Local fitness landscape of the green fluorescent protein. Nature 533, 397–401 (2016).
69. Bryant, D. H. et al. Deep diversification of an AAV capsid protein by machine learning. Nat. Biotechnol. 39, 691–696 (2021).
70. Wu, N. C., Dai, L., Olson, C. A., Lloyd-Smith, J. O. & Sun, R. Adaptation in protein fitness landscapes is facilitated by indirect paths. Elife 5, e16965 (2016).
71. Rocklin, G. J. et al. Global analysis of protein folding using massively parallel design, synthesis, and testing. Science 357, 168–175 (2017).
72. Jarzab, A. et al. Meltome atlas—thermal proteome stability across the tree of life. Nat. methods 17, 495–503 (2020).

73. Almagro Armenteros, J. J., Sønderby, C. K., Sønderby, S. K., Nielsen, H. & Winther, O. DeepLoc: prediction of protein subcellular localization using deep learning. Bioinformatics 33, 3387–3395, DOI: 10.1093/bioinformatics/btx548 (2017).
74. Klausen, M. S. et al. NetSurfP-2.0: Improved prediction of protein structural features by integrated deep learning. Proteins: Struct. Funct. Bioinforma. 87, 520–527 (2019).
75. Kingma, D. P. & Ba, J. Adam: Adam: A Method for Stochastic Optimization. Preprint at https://arxiv.org/abs/1412.6980 (2014).

Acknowledgements

Thanks to Nikita Kugut (TUM) for support with many aspects. Thanks to the anonymous reviewers for improving the work. M.H. and B.R. were supported by the Bavarian Ministry of Education through funding to the TUM, by a grant from the Alexander von Humboldt Foundation through the German Ministry for Research and Education (BMBF: Bundesministerium für Bildung und Forschung), and by a grant from Deutsche Forschungsgemeinschaft (DFG-GZ: RO1320/4-1). All computational resources for this work were provided by AbbVie. Last, but not least, thanks to all those who maintain public databases and make their resources publicly available.

Author contributions statement

All authors contributed to the conception of this study. R.S. performed all experimental work, results analysis and creation of figures. The first draft of the manuscript was written by R.S. and all authors commented on subsequent versions of the manuscript.

Additional information

R.S. is an employee of AbbVie. The design, study conduct, and financial support for this research were provided by AbbVie. AbbVie participated in the interpretation of data, review, and approval of the publication.