Benchmarking text-integrated protein language model embeddings and embedding fusion on diverse downstream tasks

Young Su Ko[1], Jonathan Parkinson[1], and Wei Wang[1,2,*]
[1]Department of Chemistry and Biochemistry, University of California, San Diego, CA, USA
[2]Department of Cellular and Molecular Medicine, University of California, San Diego, CA, USA
*email: wei-wang@ucsd.edu

Abstract

Protein language models (pLMs) have traditionally been trained in an unsupervised manner using large protein sequence databases with an autoregressive or masked-language modeling training paradigm. Recent methods have attempted to enhance pLMs by integrating additional information, in the form of text, which are referred to as “text+protein” language models (tpLMs). We evaluate and compare six tpLMs (OntoProtein, ProteinDT, ProtST, ProteinCLIP, ProTrek, and ESM3) against ESM2, a baseline text-free pLM, across six downstream tasks designed to assess the learned protein representations. We find that while tpLMs outperform ESM2 in five out of six benchmarks, no tpLM was consistently the best. Thus, we additionally investigate the potential of embedding fusion, exploring whether the combinations of tpLM embeddings can improve performance on the benchmarks by exploiting the strengths of multiple tpLMs. We find that combinations of tpLM embeddings outperform single tpLM embeddings in five out of six benchmarks, highlighting its potential as a useful strategy in the field of machine-learning for proteins. To facilitate the practical application of embedding fusion, we outline a heuristic framework to efficiently identify the optimal combination of embeddings, reducing the exponential time complexity of an exhaustive combination search down to a manageable linear time complexity. Using our embedding fusion framework, we achieve state-of-the-art performances on the protein-protein interaction prediction and homologous sequence recovery tasks without any specific model adjustments or hyperparameter tuning. Our experiments suggest embedding fusion is a useful tool in the machine-learning for proteins toolbox. Lastly, this study highlights the potential of future research on additional strategies for maximizing the utility of pLMs.

Introduction

The use of pre-trained protein language model (pLM) embeddings for various machine learning based protein tasks such as variant effect prediction (1), structure prediction (2), and protein engineering (3,4) have become increasingly popular. pLMs have grown larger in training data and parameter size (5), with the anticipation that their capabilities will continue to grow. pLMs such as ESM2 (2), have shown to produce rich feature representations, capturing the co-evolutionary information between residues (6). As shown by Li et al., ESM continues to show improvements in secondary structure prediction as it increases in size from 43M to 76M to 650M parameters (7). Furthermore, secondary structure prediction improves when using representations from deeper pLM layers, indicating that the pLM continues to learn useful

features throughout pre-training. Thus, the secondary structure prediction task is well-aligned with the current pre-training paradigm. However, Li et al. also show that performance on other tasks, such as mutational landscape fitness prediction, does not follow the same trends seen with secondary structure prediction, indicating scaling up pLMs trained with the standard masked-language-modeling (MLM) pre-training task is unlikely to resolve the current issues in tackling key biological challenges, especially those aimed at understanding and predicting protein function.

Recent works have explored the integration of text-based protein descriptions into pre-training. These models, referred to as “text+protein” language models (tpLMs) attempt to enhance the learned protein representations by injecting known protein information, such as a protein’s function. This contextualization is aimed at improving a pLM’s usefulness in downstream applications. For example, OntoProtein utilized Gene Ontology and Gene Annotation terms during the pre-training step and showed improvements on the TAPE benchmarks (8), protein-protein interaction prediction, and protein function prediction (9). ProtST leveraged Swiss-Prot text annotations embedded by PubMedBERT during pre-training to learn protein representations that capture both information for additional improvements in downstream performance (10). The contrastive learning paradigm between protein sequence and text description has become popular. ProteinDT introduced the Contrastive Language and Protein pre-training (ProteinCLAP) approach to align the representations between Swiss-Prot annotations embedded by SciBert (11) and protein sequences embedded by ProtBert (12). Similarly, ProteinCLIP used contrastive learning to align protein sequence representations from pre-trained pLMs to UniProt function descriptions embedded by OpenAI’s text embedding model (13). The tri-modal pLM ProTrek was trained with a contrastive learning objective between sequence, structure, and function, aligning GPT-4-summarized UniProt descriptions embedded by PubMedBERT and protein sequence embedded by ESM2 (14). Lastly, ESM3 takes a different tri-modal approach, representing sequence, structure, and function annotations as discrete tokens and predicting the masked tokens across the three different modalities as the pre-training task (15). Despite these advancements and demonstrated improvements, tpLMs have yet to be widely adopted by practitioners. Most studies continue to rely on more established pLMs, such as ESM2 or ProtT5 (12), possibly due to their established reputations and the relative recency of tpLMs.

Moreover, the practical advantages of tpLMs have yet to be fully established. While the authors of these models compare with baseline pLMs such as ESM2 or ProtBert, ProteinCLIP is the only model that has been compared against the 3 billion parameter ESM2 (ESM2 3B), a model with significantly more parameters than tpLMs. This lack of direct comparisons leaves practitioners uncertain about whether tpLMs truly offer a practical advantage over simply using a larger text-free pLM. To address this gap, we have conducted comprehensive benchmarking of six state-of-the-art tpLMs across six downstream tasks, ranging from mutational fitness prediction to subcellular location prediction. By using ESM2 3B as our baseline, we aim to elucidate whether the integration of textual protein information provides advantages unattainable by simply scaling up traditional pLM pre-training.

From our experiments, while we observed at least one tpLM outperform ESM2 3B on five out of six benchmarks, no tpLM was the best across all datasets. Thus, we were curious if we could combine the strengths of multiple tpLMs. While feature fusion has long been a cornerstone technique in machine learning, its application in the context of pre-trained language model embeddings is less common. Embedding fusion has been explored in the field of natural language processing where the concatenations of text embeddings improved the performance on downstream tasks such as hate detection (16–18). While this idea has also been explored in the field of computational biology, in protein function prediction (19) and post-translational modification prediction (20), the use of embedding fusion in this field remains rare and no study on the broad applicability of embedding fusion in the field of ML for proteins has been performed. We explore the effectiveness of embedding fusion, in the context of transfer learning, by evaluating all 63 possible combinations of the six tpLM embeddings on the same benchmark datasets.

We observed that for five out of six benchmarks, there existed a combination of tpLM embeddings that outperformed the best single tpLM embedding. While the results were encouraging, identifying the best embedding combination required an exhaustive search of all possible combinations, a task with an exponential time complexity, posing a major hurdle for the practical usefulness of embedding fusion. To address this limitation, we outline a heuristic framework based on a greedy search algorithm to efficiently identify an optimal combination of embeddings in linear time. Using this heuristic, we achieve state-of-the-art performances on the protein-protein interaction prediction and homologous sequence recovery tasks. Our experiments highlight that embedding fusion is an effective and practical method for exploiting the strengths of multiple pLMs, and underscore the importance of not only developing new pLMs but also exploring strategies on how to fully exploit existing pLMs.

Results

Our study is composed of three parts: benchmarking of tpLMs and comparison with ESM2 3B (**Fig. 1A**), evaluating the potential of embedding fusion with tpLMs (**Fig. 1B**), and applying embedding fusion to the protein-protein interaction prediction and homologous sequence recovery tasks (**Fig. 1C**).

tpLMs can outperform ESM2 3B

Starting with the list of tpLMs included in the survey by Pei et al. (21), we identified four tpLMs for analysis by filtering for two criteria. First, the models do not require any other information besides the protein sequence as input. Second, the authors should indicate the hidden representations of the models are intended to be used as embeddings for downstream tasks. Based on these filters, we selected OntoProtein, ProtST, ProteinDT, and ProteinCLIP. While not mentioned by Pei et al. due to their recency, we add ProTrek and ESM3 as they both fit the criteria, for a total of six tpLMs.

We then assessed the performances of these six tpLM representations and ESM2 3B on six benchmark datasets (**see Methods**) to answer the following questions. First, do tpLMs outperform ESM2 3B, highlighting the power of text integration and encouraging the use of tpLMs over ESM2 3B? Second, is there a “best” tpLM that consistently outperforms all other tpLMs?

As shown in **Fig. 1D**, based on the average performance (accuracy for the Location benchmark, Spearman's ρ for all others) across five random seeds, one of the six tpLM outperforms ESM2 3B on all benchmarks except AAV (**see Discussion**). However, we also observe that not all tpLMs outperform ESM2 3B–GFP and Location are the only benchmarks in which more than one tpLM outperforms ESM2 3B. In addition, no model is consistently the best. Out of the tpLMs, ProTrek ranks first three times, OntoProtein ranks first twice, and ESM3 ranks first once. Individual runs are recorded in **Supplementary Table 1,2**.

Combinations of tpLMs can outperform using a single tpLM

A key observation was that no embedding was consistently the best across all tested benchmarks. This signaled that first, in any task, it is worth assessing multiple tpLMs/pLMs to identify which representation is the best for the task at hand, like many projects have already done (22). Second, it encouraged the exploration of embedding fusion. As the tpLMs have different pre-training strategies and training data, we believed the strengths of different tpLMs could be exploited. We looked to answer the question: do combinations of tpLM embeddings outperform using a single tpLM embedding?

To do so, we assessed every single possible combination of tpLM embeddings on the six benchmarks (**Supplementary File 1**). Given the six tpLMs, there were 63 total combinations, where a combination is created by the concatenation of the individual embeddings, where order does not matter. For each benchmark, we compared the best combination against the best single embedding. However, given the larger input dimension of the combined embeddings, we recognized the projection layer will generally have more parameters than when using a single embedding. In order to distinguish changes due to combining embeddings from changes due to difference in parameter count, we apply the following procedure. For each benchmark, after identifying the best combined embedding, we calculate the combined input dimension size. Then, we concatenate the best performing single embedding multiple times to itself to match the input dimension size of the best combined embedding. If the input dimension size cannot be exactly matched, we round up to allow the repeated single embedding to have a larger input dimension. If repeating single embeddings decreases performance, we use the non-repeated single embedding for comparison. The best embedding combinations and best repeated single embeddings for each benchmark are shown in **Supplementary Table 3**.

In four out of six benchmarks, we found that combining different embeddings led to a noticeable increase in Spearman's rank correlation coefficient (ρ), compared to using just one embedding. These improvements were statistically significant based on non-overlapping 95% confidence intervals (**Fig. 1E**). While we did not see a statistically significant increase for Stability, we

nonetheless observe a numerical increase in Spearman's ρ. However, we see a non-significant decrease in accuracy for Location (**see Discussion**). We do not observe a pattern to the best combinations, meaning that the best performing combination is dataset dependent. As a result, although we find the combination of embeddings generally outperform using a single embedding, the embedding fusion strategy is largely burdened by the computational cost of checking every combination to find the best performing combination.

Heuristic identifies optimal combinations of embeddings in linear time

As we did not observe a generalizable pattern to the best embedding combinations, the exponential-scaling time complexity required to assess all possible combinations of embeddings and identify the best combination posed a significant hurdle for applying embedding fusion in practice. However, as we saw its potential, we looked to improve the practicality of embedding fusion by bypassing the exhaustive combination search.

Based on our embedding fusion benchmarking results (**Supplementary File 1**), two observations facilitated the development of a heuristic algorithm for optimal combination identification. First, the best performing single embedding was consistently included in the best performing combination. Second, contrary to our initial intuition that the best combination would consist only of the top-performing single embeddings, we discovered that in some cases, using a combination that included the worst single embedding yielded the most effective results. Thus, our heuristic needed to meet two requirements: return a combination that includes the best performing single embedding and consider combinations that include even the worst performing embeddings.

To meet these requirements, we adopted a greedy algorithm with an O(N) linear time complexity (**Fig. 2A**). For N embeddings, we start by assessing the performance of each embedding individually on a validation dataset. We then rank these embeddings from best to worst. We designate the best-performing embedding e_1 to be the first member of the optimal combination. We then evaluate the performance of the combination e_1+e_2 on the validation set. If this combination outperforms e_1 alone, we accept e_2 as another member of the optimal solution, and evaluate the performance of $(e_1e_2)+e_3$. If not, e_2 is excluded, and we check e_1+e_3, continuing this process through e_N (**Supplementary Fig. 1**). This approach ensures the best performing single embedding is always included and that even the worst performing single embeddings are considered.

This approach requires evaluating the validation performance N times for single embeddings plus N−1 times for the combined embeddings, requiring an overall O(N) time complexity. This marks a significant improvement from $O(2^N)$ exponential scaling. Given that the assessment of multiple pLM embeddings should be a standard procedure in studies involving pLM embeddings, the addition of N−1 runs to explore combined embeddings can be integrated easily with existing workflows without a significant increase in computational load. However, it is important to acknowledge that while this heuristic significantly streamlines the search process, it may only find a local optimum and may not reach the global optimum, which can only be

guaranteed with an exhaustive search. Based on our backtest of this heuristic on the six benchmark datasets (**Supplementary Tables 4-10**), we observed that the heuristic can often guide to a strong combination that is close to the best combination. As a heuristic, as we do not expect it to identify the best combination every time, we believe it has served its purpose well. We observe differences between the best embeddings on the validation dataset and the best embeddings on the test dataset, likely due to overfitting or differences in data distribution between the validation and test datasets. Even with single embeddings, the embedding with the best validation performance does not necessarily have the best test performance, indicating it is not a result of embedding fusion. This observation has also been made by Schmirler et al. (23) who ran evaluations on the same datasets used in this study.

Embedding fusion improves protein-protein interaction prediction and homologous sequence recovery

In the last stage of our study, we demonstrate a practical application of embedding fusion, alongside its optimal combination identification heuristic, on the challenging tasks assessed by Wu et al. in their ProteinCLIP study (13). Both tasks offer new insights on the broad applicability of embedding fusion. First, the protein-protein interaction prediction task diverges significantly from the benchmarks initially tested; it necessitates the learning of pairwise interaction relationships, contrasting with tasks that assess isolated properties such as fitness. Second, the homologous sequence recovery task, as defined by Wu et al., does not use a deep learning framework. This setting allows us to explore the effectiveness of embedding fusion within non-deep learning contexts as well. Lastly, these tasks provide additional opportunities to observe how well the heuristic can identify an optimal combination of embeddings by comparing the heuristic-identified combination with the exhaustive-search-identified true best combination.

Results on protein-protein interaction prediction

Addressing concerns regarding data leakage in various biological deep-learning studies, including protein-protein interaction (PPI) prediction, Bernett et al. (24) created a leakage-free human protein-protein interaction dataset, which we refer to as the Bernett dataset. Most recently, Wu et al. achieved state-of-the-art performance on the Bernett dataset using ProteinCLIP embeddings (13). As Wu et al. credit the function-centric representations learned by ProteinCLIP as the driving factor behind improvement on PPI prediction, we reasoned that applying embedding fusion with tpLMs can leverage the function-centric aspects of multiple tpLMs and further improve performance.

Before Wu et al., NaderiAlizadeh and Singh (25) and Sledzieski et al. (26) have also trained PPI classifiers using the Bernett dataset. Both works use simple multi-layer perceptron based models paired with ESM2 embeddings, allowing for additional comparisons with other independently reported baselines. For our embedding fusion experiments, we use the same simple classifier used by NaderiAlizadeh and Singh (25) with the same hyperparameters.

Using our heuristic, we identified the combination of ProteinCLIP, ProtST, and ProTrek to be the optimal combination based on the average AUPRC across five seeds on the validation set

(**Supplementary Table 11**). On the test set, we find that the selected combination of embeddings achieves a state-of-the-art AUPRC and accuracy (**Fig. 2B**). After assessing every possible combination, we find the true best combination based on the test set AUPRC is the combination of ProteinCLIP, ProtST, ProTrek, and ProteinDT (**Supplementary File 1**), which achieves even higher performance. As we find both the heuristic identified combination and the true best combination outperform any single tpLM embedding, we find embedding fusion successful in improving PPI prediction. While we did not identify the best combination, we quickly identified a combination that outperforms the best single embedding, highlighting the effectiveness of the heuristic.

Results on homologous sequence recovery

Another task used by Wu et al. is the homologous sequence recovery task. In this task, a query protein domain embedding is compared with the embeddings of all other sequences in a database to identify the most similar sequence, as defined by the cosine similarity between the two embeddings. If the domain associated with the most similar embedding belongs to the same homologous superfamily, it is considered a correct retrieval. Similarly to the protein-protein interaction task, ProteinCLIP excels at homologous sequence recovery, achieving a higher top-1 accuracy than the state-of-the-art MMseqs2. Again, we hypothesized that embedding fusion could produce function-dense representations able to retrieve homologous sequences with low sequence identity.

Using the heuristic, we identified the combination of ESM3, ProteinCLIP, ProTrek, and ProteinDT to be the optimal combination based on the top-1 accuracy (**Supplementary Table 12**). This combination achieves a top-1 accuracy of 81.6%, an improvement from the previously best 80.8% achieved using ProteinCLIP embeddings (**Fig. 2C**). While we do not observe a significant improvement in accuracy, after performing the exhaustive combination search, we find the heuristic was able to identify the true optimal combination, further affirming its effectiveness.

Discussion

Our study begins with a benchmarking of six state-of-the-art tpLMs against the large text-free ESM2 3B. We find that more often than not, incorporating function-related text is beneficial on various downstream tasks, including those involving mutational effect prediction and those involving diverse proteins such as melting temperature prediction. However, no tpLM outperformed ESM2 3B on AAV. This is likely due to the fact that performance on AAV “two-vs-many” has been shown to improve with model size even though the task is not well-aligned with pre-training (7), suggesting tpLMs could have outperformed ESM2 3B if they were larger. Even beyond the AAV dataset, we see not all tpLMs outperform ESM2 3B. It is possible that the benefits of text integration are still overshadowed by ESM2 3B’s larger parameter count. Future research can explore the application of text integration techniques to larger foundational pLMs, although we acknowledge this is significantly more computationally demanding. Another consideration is the exhaustion of the UniProt database, where most tpLMs find their text training data. Future work should explore additional sources of information that

have yet to be exploited. Based on the tpLM benchmarking results, if limited to choosing only one tpLM, we recommend ProTrek as it outperformed ESM2 3B most frequently–in four out of six benchmarks.
However, our findings also suggest embedding fusion should be applied whenever possible, as it increased performance for five benchmarks. We do observe one instance in which no combination of embeddings can outperform ProTrek in the subcellular location prediction benchmark. While unclear exactly what each model has learned given the black-box nature of language models, a possible explanation is that ProTrek has a much better understanding of the protein subcellular location compared to any other tpLM. Both ProtST and ProTrek use the UniProt “Subcellular Location” annotations as part of the text training data. In that regard, it is understandable that ProTrek and ProtST have the highest accuracy on the Location dataset. However, the combination of ProtST and ProTrek performs worse than ProTrek alone. Upon further investigation, we noticed while ProtST's text encoder is frozen, ProTrek’s text encoder is updated during pre-training. This increased text encoder flexibility likely allows ProTrek to develop a more accurate understanding of the relationship between protein sequences and their corresponding subcellular locations. Consequently, when ProTrek's embeddings are combined with those from ProtST, or any other tpLM, it may obscure ProTrek’s strong learned representation and decrease performance.

Lastly, embedding fusion allowed us to achieve state-of-the-art performances for predicting protein-protein interactions and identifying homologous proteins with low sequence identity. Furthermore, our heuristic was effective in quickly identifying an optimal combination of embeddings. We highlight that this study did not involve any specific hyperparameter tuning or adjustments to the model architecture, emphasizing the robustness and ease of integrating embedding fusion into existing workflows. While we could not do so due to the large computational cost, we believe that the performance gains associated with embedding fusion could have been amplified had we performed hyperparameter tuning.

We also acknowledge the limitations of our findings. The first limitation stems from the use of pLMs, in general. Recent works have highlighted the species bias present in the training data for pLMs as well as the potential data leakage between the pre-training data and the downstream task (27,28). As a result, performance on pLM embeddings on benchmark datasets may be over optimistic in regards to its ability to generalize to unseen protein sequences. While datasets used in this study have been partitioned to consider the sequence similarity between the training and test splits, we could not entirely eliminate the potential data leakage between the text training data and downstream tasks.

While fine-tuning and parameter-efficient fine-tuning (PEFT) of pLMs has been shown to be effective in improving performance on downstream tasks (23), our study focuses on embedding-based transfer learning. As embedding-based transfer learning remains the most broadly and commonly used application of pLMs, we highlight embedding fusion as an approach that can be directly applied to existing studies without requiring large computational resources or model adjustments. A future direction is exploring ways to combine the idea of embedding

fusion with fine-tuning techniques. In addition, although all tasks in this study employ average-pooled protein representations suitable for whole-protein related tasks, some tasks may require residue-level representations with shape [sequence length, dimension]. While such embeddings can still be concatenated, the resultant combined embeddings may become unmanageably large when combining more than three embeddings, posing challenges for practical application. Lastly, our experiments on embedding fusion only involve the concatenation of embeddings. While there are other ways to fuse embeddings, we find concatenation to be straightforward and effective.

In summary, we highlight embedding fusion as a simple yet robust strategy that can and should be applied in any computational proteomics work involving protein language model embeddings. We address the main drawback of embedding fusion, which is that the best embedding combination must be identified on a case-by-case basis. With our outlined heuristic, we transform embedding fusion from an interesting but largely impractical idea to a realistic strategy. While embedding fusion's success is not guaranteed, it comes with virtually no drawbacks and it is always worth attempting. Much like the studies mentioned above on pLM embedding aggregation and fine-tuning (23,25,26), this work fits into the greater field of protein language model exploitation. We believe there is much to be gained in further investigating how to utilize protein language models at their full potential–by doing so, we hope to not only improve performance but to also expand the utility of protein language models for the scientific community.

Methods

Protein language models

To select the tpLMs used in this study, we began with Pei et al.'s survey on biomolecular modeling with natural language (21). We filtered for models that only require an amino acid sequence to generate an embedding as we are primarily interested in exploring the capabilities and limitations of sequence-only approaches. We further filter for models that are intended to generate embeddings for downstream tasks, focusing on protein representation learning rather than focusing on building a natural language LLM for protein generation, such as ProtLLaMA (29) or ProtLLM (30). In addition to OntoProtein, ProtST, ProteinDT, and ProteinCLIP which are mentioned by Pei et al., we include ProTrek and ESM3 as they fit both criteria.

While data leakage between the tpLM training data text and the downstream tasks may exist, we do not retrain any pLM to reduce data leakage due to the time and computational requirements. For each tpLM, we briefly describe its text training data below. Additional details of the selected pLMs are shown in **Table 1**.

ESM2

ESM2 is the standard pLM we use to compare against the tpLMs. ESM2 is trained at six different scales, ranging from 8M parameters to 15B parameters. We selected the 3B parameter model to serve as the baseline pLM for comparison. While we considered using the smaller 150M or 650M parameter models so that all tested models are comparable in parameter size,

we opted to use the 3B parameter model for two reasons. First, we noticed only ProteinCLIP provides a direct comparison to ESM2 3B. While ProteinCLIP showed improvements over ESM2 3B, it remained unclear whether the same was true for the other pLMs. Second, and more importantly, although ESM2 3B is larger than most of the function-trained models we assess, we wanted to understand whether tpLMs have a practical use case–would a practitioner ever choose a smaller tpLM over ESM2 3B?

ProteinCLIP

The text data for training ProteinCLIP originates from the UniProt **"Function"** section.

There are six variations of ProteinCLIP, where the base model is one of ProtT5, ESM2 8M, 35M, 150M, 640M, and 3B. To select the ProteinCLIP model for the six benchmark tasks, we calculate the average accuracy of the six models on ProteinCLIP's reported tasks of protein-protein interaction prediction and homologous protein recovery. As ProteinCLIP trained on top of ProtT5 has the highest average accuracy, we use the ProtT5 variation for the six initial benchmark tasks. For our application of embedding fusion on the protein-protein interaction and homologous sequence recovery tasks we use the ProteinCLIP models with the best reported performance, ProtT5 and ESM2 3B respectively.

ProteinDT

The SwissProtCLAP text-protein pair dataset constructed by the authors of ProteinDT uses the SwissProt annotated subset of UniProt. From SwissProt, the **"Function"**, **"PTM/Processing"**, and **"Interaction"** sections are used.

ProteinDT offers two variations ProteinDT-InfoNCE and ProteinDT-EBM-NCE, differing in the loss function used during pre-training. We select ProteinDT-InfoNCE as it has a higher average performance across the benchmarks tested by the authors.

ProtST

ProtST uses text from the SwissProt annotated subset of UniProt. From SwissProt, the **"Function"**, **"Subcellular Location"**, and **"Sequence similarities"** sections (under "Family & Domains").

ProtST provides three options: ProtST-ProtBert, ProtST-ESM-1b, and ProtST-ESM2, differing in the base model used. Based on the performance on function annotation and the recommendation provided by the authors, we select ProtST-ESM-2 for this study.

ProTrek

ProTrek uses text from the UniProt database. The authors state that "**almost all subsections**" of the UniProt database are used.

ProTrek offers two pre-trained models with different parameter sizes (35M and 650M). We select the 650M model as it was highlighted by the authors in their comparison with other state-of-the-art pLMs.

OntoProtein
OntoProtein pairs **Gene Ontology** and **Gene Annotations** with protein sequence.

There is only one pre-trained OntoProtein model available.

ESM3
ESM3 uses residue-level text descriptions from InterPro annotations (annotation term names, associated GO terms per annotation, and ancestor GO terms).

There is only one pre-trained ESM3 model available.

Table 1. Parameter and embedding sizes of selected pre-trained models.

Model	Base pLM*	Parameter Size**	Embedding Size
ESM2	N/A	3B	2560
OntoProtein	ProtBERT	420M	1024
ProteinDT	ProtBERT	420M	1024
ProtST	ESM2	650M	512
ProteinCLIP	ESM2, ProtT5	3B, 1.2B	128
ProTrek	ESM2	640M	1024
ESM3	N/A	1.4B	1546

*Base pLM column is only applicable to models that are trained on top of pre-trained pLMs. ESM2 and ESM3 are trained from scratch.
**Parameter size refers to the base pLM parameter count.

Embedding generation

To generate the embeddings, we follow the instructions provided by the respective authors in their GitHub repositories. For four out of our seven tested pLMs, an N-amino acid long protein sequence input returns a [start token + N + end token, hidden dimension] shaped representation. We first remove the start and end tokens then mean-pool across the length axis to return a fixed-length vector for each protein. For ProtST, ProteinCLIP, and ProTrek, this step is not necessary as the model by default returns a fixed length vector.

Due to memory constraints, we set the maximum protein sequence length to be 5800AA. For sequences longer than 5800AA, we select the first 2400 residues and the last 2400 residues,

concatenate the residues to create a pseudo-protein sequence following Li et al. (7) who reasoned taking the N and C terminuses preserves biologically relevant signals.

Datasets

The six datasets used as benchmarks in this study were the same datasets used by Schmirler et al.'s study on fine-tuning pLMs (23). Given the diversity of tasks, we believed this was a good set of benchmarks for our study as well. We excluded the disorder (31) and secondary structure prediction (32,33) tasks because protein structure tasks have already been shown to be well aligned with current pLM pre-training (7). The two datasets used for applying embedding fusion were the same datasets used by Wu et al. (13). We briefly describe each dataset below. Additional details regarding the training/validation/test splits and the number of sequences in the datasets are shown in **Supplementary Table 13**.

GB1

The GB1 dataset (34) is a mutational dataset measuring the fitness (stability and binding affinity) of different mutants of the binding domain of Protein G. We use the 3-vs-rest split created by the FLIP benchmark (35) authors. The training and validation data consist of wild type, single, double and triple mutants, while the test data consist of all other mutants.

GFP

The GFP dataset (36) is a deep mutational scanning dataset that reports the fluorescence intensity of different green fluorescent protein mutants. The training, validation, and test splits are created by the TAPE benchmark (8) authors. The training and validation data are in a Hamming distance 3 neighborhood of the original sequence, while the test data is from the Hamming distance 4-15 neighborhood.

AAV

The AAV dataset (37) is a mutational screening dataset that measures the fitness of different adeno-associated virus 2 capsid protein mutants. We use the 2-vs-rest split created by the FLIP benchmark (35) authors. The training and validation data consists of wild type, single, and double mutants while the test data consists of all other mutants.

Location

The Location dataset (38,39) is a 10-class classification dataset that involves the prediction of a protein's subcellular location. The training and validation originates from DeepLoc (38) while the test set is Stärk et al.'s (39) "setHard" dataset, in which no sequences have more than 20% pairwise sequence identity from the DeepLoc training and validation data. The "setHard" test set provides a better evaluation of a pLM's learned representation on unseen data than the DeepLoc test set.

Meltome

The Meltome dataset (40) collects the melting temperatures of proteins from 13 species. We use the mixed split created by the FLIP benchmark (35) authors. After all sequences are

clustered, cluster representatives are selected using MMseqs2 at a 20% sequence identity threshold. Training data consists of sequences in 80% of clusters while the test data only consists of cluster representatives in the remaining 20% of clusters.

Stability

The Stability dataset (41) collects protease susceptibility measurements. We use the split created by the TAPE benchmark (8) authors. The training and validation data consists of sequences from four rounds of experimental tests while the test data consists of sequences in 1-Hamming distance neighborhoods around the promising proteins observed in the four experimental rounds.

Protein-Protein Interaction Prediction

The “gold-standard” human protein-protein interaction dataset created by Bernett et al. (24) is a binary classification dataset (interacting pair vs non-interacting pair). There is no overlap between training, validation, and test datasets. In addition, the sequence identity of sequences across sets are capped at 40%.

Homologous Sequence Recovery

The CATH dataset (42) consists of protein domains. Following Wu et al. (13) we use version 4.3.0 and use CATH S20, in which two sequences have a maximum sequence identity of 20% and 60% overlap. All proteins in non-singleton superfamilies are designated as query proteins. There is no training, validation, and test split for this dataset as it is not used for a machine learning task.

Models

Regressors and Classifier for the six benchmark datasets

We implemented a simple model based on Schmirler et al.’s work (23). All models begin with batch normalization of the input data, followed by a hidden layer, ReLU activation, dropout, and an output layer. The output dimension is 10 for the subcellular location classifier and 1 for all other regressors. The loss function is Cross Entropy Loss for the subcellular location classifier and Mean Squared Error Loss for all other regressors. The loss is optimized via Adam without a learning rate scheduler. The hidden dimension size is set to 32 and the dropout rate is set to 20%. The weights from the epoch with the lowest validation loss is used on the test set. These hyperparameters and training configurations are used by Schmirler et al. However, we scaled the batch size from 8 to 4096 to reduce training time. As we observed a drop in performance compared to Schmirler et al.’s results due to change in batch size, we scaled the original learning rate of 0.0001 to $0.0001 \times \sqrt{512}$ based on the guidance by Malladi et al. (43). Detailed descriptions of the hyperparameters are shown in **Supplementary Table 14**.

Protein-protein interaction classifier

We use the simple protein-protein interaction classifier used by NaderiAlizadeh and Singh (25). As the simple MLP used by Wu et al. (13) concatenates the projected protein representations before a final output layer, we prefer the NaderiAlizadeh classifier as it provides permutational

invariance with respect to the input order of the proteins. The protein-protein interaction classifier uses one projection layer followed by a ReLU activation. The binary prediction (0 or 1) is determined by rounding the cosine similarity between the two protein projections at the 0.5 threshold. The model uses Binary Cross Entropy as the loss function, which is optimized using the AdamW optimizer paired with a Cosine Annealing Warm Restarts learning rate scheduler that restarts every 10 epochs. The hidden dimension size is set to 1024. The weights from the epoch with the highest validation AUPRC is used on the test set. These hyperparameters are the set by NaderiAlizadeh and Singh (25). While we scaled the batch size from 32 to 4096, we kept the learning rate constant as no drop in performance was observed. Detailed descriptions of the hyperparameters are shown in **Supplementary Table 15**.

Homologous sequence recovery

We use the same method for homologous sequence as Wu et al. (13). Given a list of query proteins and a protein database, we calculate the cosine similarity between the query protein embedding and embedding of every protein in the database. We retrieve the protein with the highest similarity to the query, excluding itself, and check if the query protein and the retrieved protein belong to the same superfamily. If they belong to the same homologous superfamily, we consider that a successful retrieval. The reported accuracy is the number of successful retrievals divided by the number of queries made. There are no hyperparameters for this retrieval pipeline. Following Wu et al., we L2-normalize all embeddings before the retrieval pipeline.

Training and Implementation

All models are implemented in PyTorch v2.2.2 and trained on one NVIDIA RTX A6000 GPU with 48Gb memory. All deep-learning models are run five times using five different seeds (2,4,8,16, and 32). For homologous sequence recovery, we perform a single run as the results are deterministic.

To create a combined embedding, we concatenate the fixed length representations as a preprocessing step. Each protein language model is designated by an alphabetic character starting from the letter A (e.g., A: ESM2, B: ESM3, C: OntoProtein, D: ProteinCLIP, E: ProtST, F: ProTrek, G: ProteinDT). With six different embeddings available, we explore $2^6-1=63$ possible combinations—subtracting one to exclude the empty set. We do not consider the order of the embeddings, thus combinations are always sorted alphabetically by their one-letter code. For instance, the combination of embeddings G, D, and E would be ordered as {DEG} in the concatenated representation.

Data and Code Availability

The data and code necessary to reproduce the results from this study are available at https://github.com/Wang-lab-UCSD/Benchmarking-tpLMs.

Author Contributions

Young Su Ko: methodology, software, investigation, and writing (original draft). **Jonathan Parkinson:** software, validation, writing (review and editing). **Wei Wang:** supervision and writing (review and editing).

References

1. Brandes, N., Goldman, G., Wang, C. H., Ye, C. J. & Ntranos, V. Genome-wide prediction of disease variant effects with a deep protein language model. *Nat Genet* **55,** 1512–1522 (2023).
2. Lin, Z. *et al.* Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science* **379,** 1123–1130 (2023).
3. Hie, B. L. *et al.* Efficient evolution of human antibodies from general protein language models. *Nat Biotechnol* **42,** 275–283 (2024).
4. Madani, A. *et al.* Large language models generate functional protein sequences across diverse families. *Nat Biotechnol* **41,** 1099–1106 (2023).
5. Chen, B. *et al.* xTrimoPGLM: Unified 100B-Scale Pre-trained Transformer for Deciphering the Language of Protein. Preprint at http://arxiv.org/abs/2401.06199 (2024)
6. Zhang, Z. *et al.* Protein language models learn evolutionary statistics of interacting sequence motifs. Preprint at https://doi.org/10.1101/2024.01.30.577970 (2024)
7. Li, F.-Z., Amini, A. P., Yue, Y., Yang, K. K. & Lu, A. X. Feature Reuse and Scaling: Understanding Transfer Learning with Protein Language Models. Preprint at https://doi.org/10.1101/2024.02.05.578959 (2024)
8. Rao, R. *et al.* Evaluating Protein Transfer Learning with TAPE. Preprint at http://arxiv.org/abs/1906.08230 (2019)
9. Zhang, N. *et al.* OntoProtein: Protein Pretraining With Gene Ontology Embedding. Preprint at http://arxiv.org/abs/2201.11147 (2022)
10. Xu, M., Yuan, X., Miret, S. & Tang, J. ProtST: Multi-Modality Learning of Protein Sequences

and Biomedical Texts. Preprint at http://arxiv.org/abs/2301.12040 (2023)

11. Beltagy, I., Lo, K. & Cohan, A. SciBERT: A Pretrained Language Model for Scientific Text. Preprint at http://arxiv.org/abs/1903.10676 (2019)
12. A. Elnaggar *et al.* ProtTrans: Toward Understanding the Language of Life Through Self-Supervised Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **44,** 7112–7127 (2022).
13. Wu, K. E., Chang, H. & Zou, J. ProteinCLIP: enhancing protein language models with natural language. Preprint at https://doi.org/10.1101/2024.05.14.594226 (2024)
14. Su, J., Zhou, X., Zhang, X. & Yuan, F. ProTrek: Navigating the Protein Universe through Tri-Modal Contrastive Learning. Preprint at https://doi.org/10.1101/2024.05.30.596740 (2024)
15. Hayes, T. *et al.* Simulating 500 million years of evolution with a language model. Preprint at https://doi.org/10.1101/2024.07.01.600583 (2024)
16. Lester, B., Pressel, D., Hemmeter, A., Choudhury, S. R. & Bangalore, S. Multiple Word Embeddings for Increased Diversity of Representation. Preprint at http://arxiv.org/abs/2009.14394 (2020)
17. Badri, N., Kboubi, F. & Chaibi, A. H. Combining FastText and Glove Word Embedding for Offensive and Hate speech Text Detection. *Procedia Computer Science* **207,** 769–778 (2022).
18. Khan, M. A., Yadav, N., Jain, M. & Goyal, S. The Art of Embedding Fusion: Optimizing Hate Speech Detection. Preprint at http://arxiv.org/abs/2306.14939 (2023)
19. Chua, Z. M., Rajesh, A., Sinha, S. & Adams, P. D. PROTGOAT : Improved automated protein function predictions using Protein Language Models. Preprint at https://doi.org/10.1101/2024.04.01.587572 (2024)
20. Pokharel, S., Pratyush, P., Ismail, H. D., Ma, J. & Kc, D. B. Integrating Embeddings from

Multiple Protein Language Models to Improve Protein O-GlcNAc Site Prediction. *Int J Mol Sci* **24,** 16000 (2023).

21. Pei, Q. *et al.* Leveraging Biomolecule and Natural Language through Multi-Modal Learning: A Survey. Preprint at http://arxiv.org/abs/2403.01528 (2024)
22. Jiang, K. *et al.* Rapid protein evolution by few-shot learning with a protein language model. Preprint at https://doi.org/10.1101/2024.07.17.604015 (2024)
23. Schmirler, R., Heinzinger, M. & Rost, B. Fine-tuning protein language models boosts predictions across diverse tasks. Preprint at https://doi.org/10.1101/2023.12.13.571462 (2023)
24. Bernett, J., Blumenthal, D. B. & List, M. Cracking the black box of deep sequence-based protein–protein interaction prediction. *Briefings in Bioinformatics* **25,** bbae076 (2024).
25. NaderiAlizadeh, N. & Singh, R. Aggregating Residue-Level Protein Language Model Embeddings with Optimal Transport. Preprint at https://doi.org/10.1101/2024.01.29.577794 (2024)
26. Sledzieski, S. *et al.* Democratizing protein language models with parameter-efficient fine-tuning. *Proceedings of the National Academy of Sciences* **121,** e2405840121 (2024).
27. Ding, F. & Steinhardt, J. Protein language models are biased by unequal sequence sampling across the tree of life. Preprint at https://doi.org/10.1101/2024.03.07.584001 (2024)
28. Hermann, L., Fiedler, T., Nguyen, H. A., Nowicka, M. & Bartoszewicz, J. M. Beware of Data Leakage from Protein LLM Pretraining. Preprint at https://doi.org/10.1101/2024.07.23.604678 (2024)
29. Lv, L. *et al.* ProLLaMA: A Protein Language Model for Multi-Task Protein Language Processing. Preprint at http://arxiv.org/abs/2402.16445 (2024)
30. Zhuo, L. *et al.* ProtLLM: An Interleaved Protein-Language LLM with Protein-as-Word Pre-Training. Preprint at http://arxiv.org/abs/2403.07920 (2024)

31. Ilzhöfer, D., Heinzinger, M. & Rost, B. SETH predicts nuances of residue disorder from protein embeddings. *Front. Bioinform.* **2,** 1019597 (2022).

32. Klausen, M. S. *et al.* NetSurfP‐2.0: Improved prediction of protein structural features by integrated deep learning. *Proteins* **87,** 520–527 (2019).

33. Elnaggar, A. *et al.* ProtTrans: Towards Cracking the Language of Life's Code Through Self-Supervised Deep Learning and High Performance Computing. Preprint at http://arxiv.org/abs/2007.06225 (2021)

34. 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).

35. Dallago, C. *et al.* FLIP: Benchmark tasks in fitness landscape inference for proteins. Preprint at https://doi.org/10.1101/2021.11.09.467890 (2021)

36. Sarkisyan, K. S. *et al.* Local fitness landscape of the green fluorescent protein. *Nature* **533,** 397–401 (2016).

37. Bryant, D. H. *et al.* Deep diversification of an AAV capsid protein by machine learning. *Nat Biotechnol* **39,** 691–696 (2021).

38. 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 (2017).

39. Stärk, H., Dallago, C., Heinzinger, M. & Rost, B. Light attention predicts protein location from the language of life. *Bioinformatics Advances* **1,** vbab035 (2021).

40. Jarzab, A. *et al.* Meltome atlas—thermal proteome stability across the tree of life. *Nat Methods* **17,** 495–503 (2020).

41. Rocklin, G. J. *et al.* Global analysis of protein folding using massively parallel design, synthesis, and testing. *Science* **357,** 168–175 (2017).

42. Sillitoe, I. *et al.* CATH: expanding the horizons of structure-based functional annotations for

genome sequences. *Nucleic Acids Research* **47,** D280–D284 (2019).

43. Malladi, S., Lyu, K., Panigrahi, A. & Arora, S. On the SDEs and Scaling Rules for Adaptive Gradient Algorithms. Preprint at http://arxiv.org/abs/2205.10287 (2023)

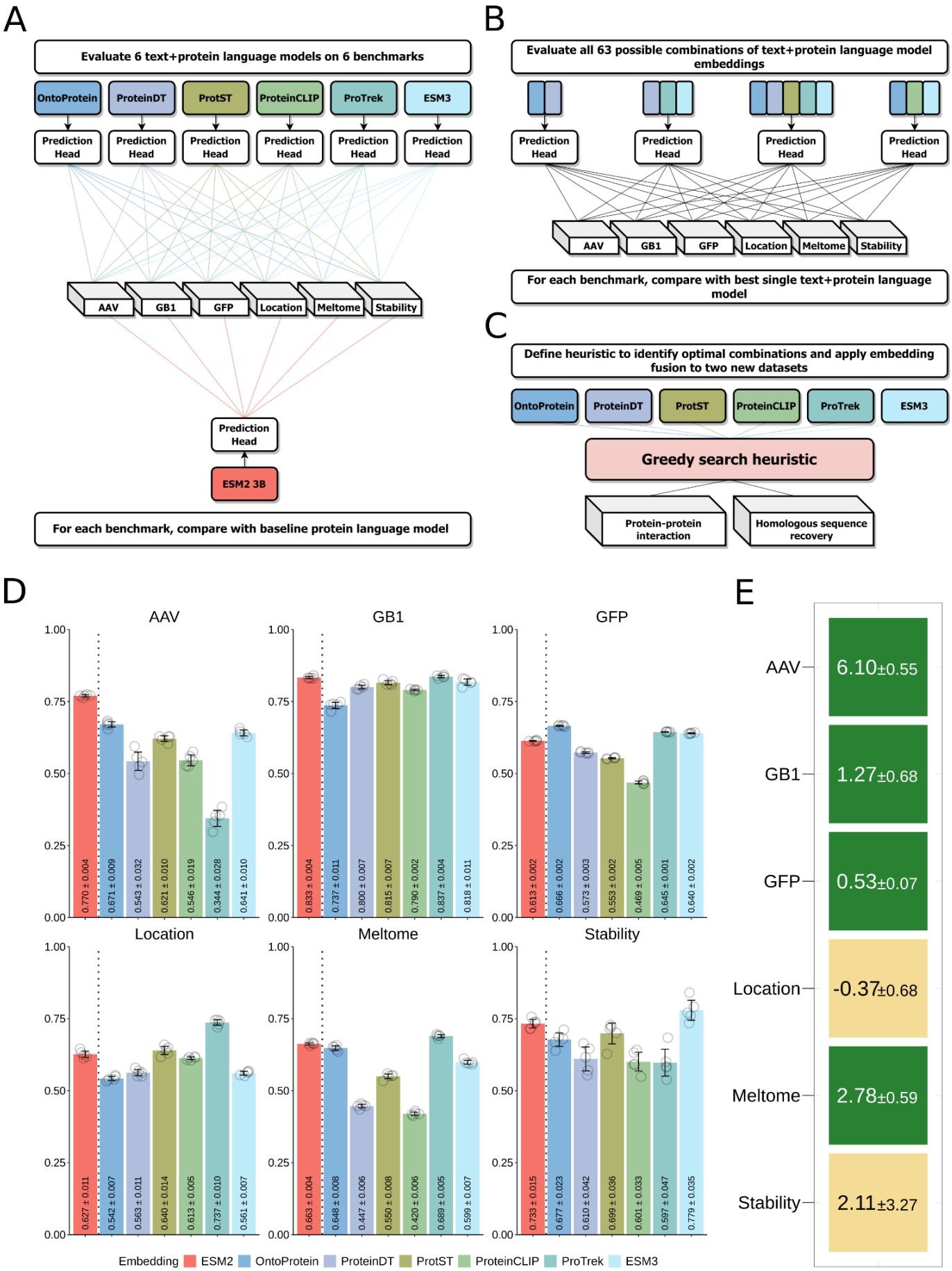
A
Evaluate 6 text+protein language models on 6 benchmarks
OntoProtein
ProteinDT
ProtST
ProteinCLIP
ProTrek
ESM3
Prediction Head
AAV
GB1
GFP
Location
Meltome
Stability
ESM2 3B
For each benchmark, compare with baseline protein language model
B
Evaluate all 63 possible combinations of text+protein language model embeddings
For each benchmark, compare with best single text+protein language model
C
Define heuristic to identify optimal combinations and apply embedding fusion to two new datasets
Greedy search heuristic
Protein-protein interaction
Homologous sequence recovery
D
AAV
GB1
GFP
Location
Meltome
Stability
0.770 ± 0.004
0.671 ± 0.009
0.543 ± 0.032
0.621 ± 0.010
0.546 ± 0.019
0.344 ± 0.028
0.641 ± 0.010
0.833 ± 0.004
0.737 ± 0.011
0.800 ± 0.007
0.815 ± 0.007
0.790 ± 0.002
0.837 ± 0.004
0.818 ± 0.011
0.613 ± 0.002
0.666 ± 0.002
0.573 ± 0.003
0.553 ± 0.002
0.469 ± 0.005
0.645 ± 0.001
0.640 ± 0.002
0.627 ± 0.011
0.542 ± 0.007
0.563 ± 0.011
0.640 ± 0.014
0.613 ± 0.005
0.737 ± 0.010
0.561 ± 0.007
0.663 ± 0.004
0.648 ± 0.008
0.447 ± 0.006
0.550 ± 0.008
0.420 ± 0.006
0.689 ± 0.005
0.599 ± 0.007
0.733 ± 0.015
0.677 ± 0.023
0.610 ± 0.042
0.699 ± 0.036
0.601 ± 0.033
0.597 ± 0.047
0.779 ± 0.035
Embedding ESM2 OntoProtein ProteinDT ProtST ProteinCLIP ProTrek ESM3
E
AAV 6.10±0.55
GB1 1.27±0.68
GFP 0.53±0.07
Location -0.37±0.68
Meltome 2.78±0.59
Stability 2.11±3.27

Figure 1. Study overview and benchmarking results. A. Six different text-integrated protein language models (OntoProtein, ProteinDT, ProtST, ProteinCLIP, ProTrek, and ESM3) are evaluated on six different datasets. Results are compared with ESM2 3B, a large text-free protein language model. **B.** We assess the effectiveness of embedding fusion. For each dataset, all 63 possible combinations of text-integrated protein language model embeddings are benchmarked. We compare the best combinations with the best single embeddings to see if embedding fusion results in an increase in performance. **C.** Based on the embedding fusion benchmarking results, we outline a heuristic to identify an optimal combination of embeddings quickly. We apply the heuristic on the protein-protein interaction and homologous sequence recovery tasks. **D.** Benchmarking results for text-integrated protein language models and ESM2 3B. 95% CIs are shown. **E.** Percent difference in performance gained by using embedding fusion compared to using a single text-integrated protein language model embedding.

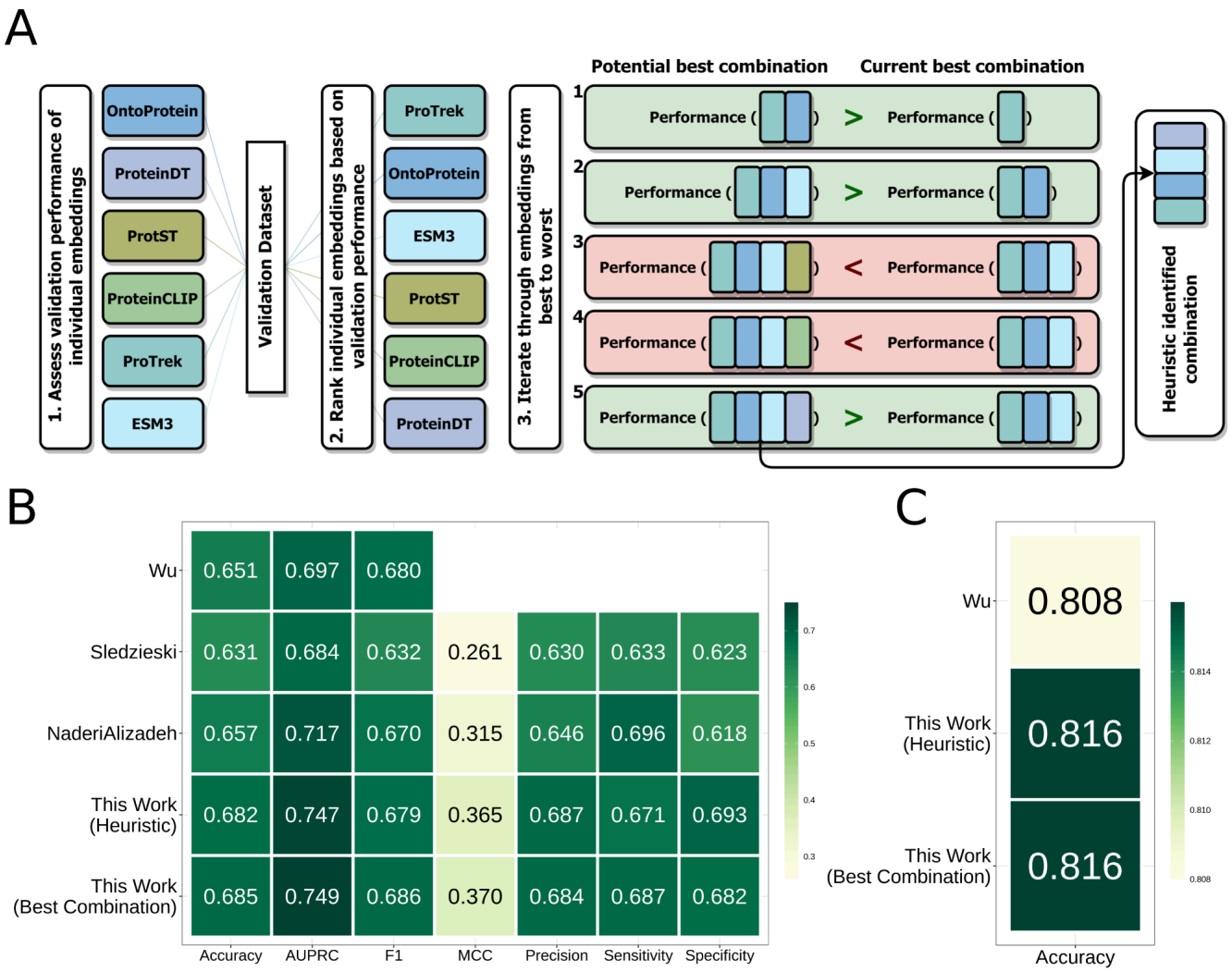

Figure 2. Heuristic framework and results on protein-protein interaction and homologous sequence recovery tasks. A. Protein language model embeddings are individually assessed, then ranked by highest validation performance. Then we iterate through the ranked list of embeddings, including an embedding as part of the optimal combination if the validation performance increases, and discarding if validation performance decreases. **B.** Results on the protein-protein interaction task. Heuristic refers to performance from the combination found by our heuristic framework, while the best combination is found by an exhaustive combination search. **C.** Results on the CATH S20 homologous sequence recovery task. Heuristic refers to performance from the combination found by our heuristic framework, while the best combination is found by an exhaustive combination search.