PLMFit: Benchmarking Transfer Learning with Protein Language Models for Protein Engineering

Thomas Bikias
Dep. of Biosystems Science and Engineering
ETH Zurich
Basel, CH 4056
tbikias@ethz.ch

Evangelos Stamkopoulos
Dep. of Biosystems Science and Engineering
ETH Zurich
Basel, CH 4056
estamkopoulo@ethz.ch

Sai T. Reddy
Dep. of Biosystems Science and Engineering
ETH Zurich
Basel, CH 4056
sai.reddy@ethz.ch

Abstract

Protein language models (PLMs) have emerged as a useful resource for protein engineering applications. Transfer learning (TL) leverages pre-trained parameters to extract features to train machine learning models or adjust the weights of PLMs for novel tasks via fine-tuning through back-propagation. TL methods have shown potential for enhancing protein predictions performance when paired with PLMs, however there is a notable lack of comparative analyses that benchmark TL methods applied to state-of-the-art PLMs, identify optimal strategies for transferring knowledge and determine the most suitable approach for specific tasks. Here, we report PLMFit, a benchmarking study that combines, three state-of-the-art PLMs (ESM2, ProGen2, ProteinBert), with three TL methods (feature extraction, low-rank adaptation, bottleneck adapters) for five protein engineering datasets. We conducted over 2900 experiments, altering PLM sizes and layers, TL hyperparameters and different training procedures. Our experiments reveal three key findings: (i) utilizing a fraction of PLM for transfer learning does not detrimentally impact performance, (ii) the choice between feature extraction and fine-tuning is primarily dictated by the amount and diversity of data and (iii) fine-tuning is most effective when generalization is necessary and only limited data is available. We provide PLMFit as an open-source software package, serving as a valuable resource for the scientific community to facilitate the feature extraction and fine-tuning of PLMs for various applications.

1 Introduction

Protein language models (PLMs) are becoming a valuable tool in computational biology with applications such as protein structure and function prediction, design and engineering. Similarly to large language models (e.g., ChatGPT [1], Llama [2]) that generate plausible sentences using human language, PLMs (e.g., ESM [3], ProGen [4], ProteinBert [5], Ankh [6], ProtTrans [7]) produce sequences of amino acids that are likely to exist in nature. Empowered by the transformers [8] architecture, they are trained on a large corpora of unlabeled natural proteins to produce sequences of amino acids with high likelihood of folding, expression and biological function. During this process, known as pre-training, multi-layered PLMs capture evolutionary [9] and structural [10]

AI for New Drug Modalities at NeurIPS 2024.

dependencies between amino acids by attempting to either reconstruct a corrupted sequence (i.e., masked language modeling [11]) or predict the next residue (i.e., token) given the previous as context (i.e., causal language modeling [12]). Acquired knowledge is stored in the weights of the different layers of PLMs and can transform the input sequence in an information rich representation (i.e., embeddings). Embeddings can be used as input features to train, typically shallow, ML models like artificial neural networks (ANNs) or convolutional neural networks (CNNs) to solve a wide variety of protein engineering tasks [13, 14, 15, 16, 17] as an alternative to encoding only the amino acid sequence information (e.g., one hot or categorical encoding) or including evolutionary information extracted from multiple sequence alignments (MSAs) [18].

Transfer learning (TL) leverages pre-trained parameters to train or adjust a different model to a novel task; TL can broadly be divided into two categories as the most prevalent techniques [19], feature extraction (FE) and fine-tuning (FT). In the context of PLMs, FE employs the retrieval of pre-trained weights from a PLM's layer which converts proteins' residues into evolutionary informed features [20] that can be used for arbitrary re-training of a model. On the contrary, FT includes the joint optimization of a PLM's (or PLM's fraction) weights with an untrained network (i.e., downstream head) using labeled data. As the size of foundational models scales pursuing an increase in generative and downstream performance (e.g., ESM-3 [21]), FT approaches face technical challenges. Arbitrary re-training of the architecture can be computationally infeasible because of their enormous size and can cause catastrophic forgetting of previously acquired knowledge [22]. Adopted from natural language processing (NLP), endeavors to mitigate these issues include parameter-efficient fine-tuning techniques (PEFT). PEFT aims to adapt pre-trained models to a new domain with minimal adjustments to the original parameters. These techniques focus on optimizing only a small subset of newly added parameters, rather than re-training the entire network, allowing lower resource consumption and maintaining or improving performance on the specific task. Prevalent practices include low-rank adaptation [23] (i.e., LoRA) and bottleneck adapter modules injection [19]. The former involves adding low-rank trainable matrices in parallel with the transformer layers, while the latter suggests the injection of small neural network modules in between each layer of the pre-trained model.

Several studies propose pairing of TL techniques with PLMs to extract meaningful representations of proteins [24, 25], mainly using the embeddings extracted from the last layer, while others investigate the effect of PEFT in protein engineering tasks [26, 27]. However, it is still unclear in which setups the exploitation of PLMs has a guaranteed benefit, as baseline models trained with one hot encoded (OHE) protein sequences can overperform PLM-based methods in relevant biology tasks [28]. Moreover, choosing the most appropriate TL method is not straightforward. Multiple factors require calibration to optimally retrieve the stored information (e.g., extraction layer, FT hyperparameters, etc.). Additionally, the amount, diversity, and quality of training data but, also, the access to hardware resources (i.e., memory and no. of GPUs) are crucial considerations that can dictate the TL approach. Recent publications either evaluate the effectiveness of PEFT methods applied to PLMs for addressing biology-related tasks [29], or attempt to identify which specific layers might be most beneficial for embeddings extraction [30]. However, layer-specific analysis of PLMs comparing simultaneously FE and PEFT methods, along with comparisons to baseline models and larger PLMs (> 5B parameters) is still missing.

Here, we report PLMFit, a comparative analysis that benchmarks TL methods applied on state-of-the-art PLMs for seven protein fitness and function prediction and classification tasks. Using publicly available datasets, we evaluate >2,900 TL setups by varying the following: (i) PLM architectures and sizes, (ii) layers of PLMs, (iii) FT hyperparameters and (iv) different training scenarios. We envision PLMFit to offer a practical guide of the optimal parameters to the scientific community to leverage pre-trained foundational models based on the nature of the task and the available resources. Finally, as an output from this study we provide an easy-to-use tool to seamlessly apply TL on proprietary data. All codes and datasets are available at https://github.com/LSSI-ETH/plmfit.

2 Results

2.1 Datasets used represent a broad range of protein engineering tasks

To establish benchmarks of TL techniques on different tasks, we utilized publicly available repositories including datasets corresponding to different types and level of complexity. Attempting to cover the most common use cases in protein engineering, we consider each task to be a different split

Table 1: Summary of the datasets, task types and splits used in the study. E_r: Enrichment ratio; T_o: Thermostability temperature; RBD: Receptor binding domain

Dataset name	Sequence length	Mutated region	Task type	Split	Training samples	Testing samples
AAV	734-749	561-588	Regression (E_r)	sampled	66,066	16,517
				one vs rest	1,170	81,413
GB1	265	V39, D40, G41, V54	Regression (E_r)	three vs rest	2,968	5,765
				one vs rest	29	8,704
Meltome	20-750	-	Regression (T_o)	mixed	24,817	3,134
RBD	201	2-201	Classification (bind/escape)	one vs rest	875	217,484
Trastuzumab	449	99-108	Classification (bind/escape)	one vs rest	174	36,386

within a dataset (i.e., how the training and test data are being separated) and we refer to a setup as the combination of a specific set of TL method's hyperparameters applied on a specific task. We classified datasets into two types: (i) those consisting of proteins within a number of mutations, i.e., edit distance (ED), relative to a wild-type (WT) sequence, where the distribution of residues remains relatively consistent, and (ii) datasets of diverse protein families with sequences exhibit a low degree of similarity. For the first category, we utilized four fitness regression (i.e., *AAV-sampled*, *AAV-one vs rest*, *GB1-three vs rest* and *GB1-one vs rest*) tasks parsed from the widely adopted FLIP repository [17] and two binding classification tasks (*RBD-one vs rest*, *Trastuzumab-one vs rest*) generated in antibody engineering studies [31, 32]. For the second category (i.e. diverse datasets), we used the *Meltome-mixed* split from the FLIP repository, which includes stability temperatures for a diverse range of protein sequences. We suggest that the complexity of each task is determined by both the quantity and diversity of the data available for training, as well as the nature of the testing data on which the prediction efficiency is evaluated. Throughout the manuscript, we refer to tasks such as *AAV-sampled* and *GB1-three vs rest* as simple due to the similarity in the distribution of the training and evaluation sets, consisting of protein sequences with varying edit distances from a wild type. In contrast, *-one vs rest* splits are described as more complex, as they involve limited training data -due to the single mutation nature of the dataset (i.e., $ED = 1$) - and their prediction capability is evaluated on variants with higher ED from the WT (i.e., $ED >= 2$). We consider the *Meltome-mixed* split to be a complex task due to the high variability in the sequences included. Details about the datasets and tasks used in this study can be found in chapter 3.3 and Table 1.

2.2 PLM layer used significantly impacts the effectiveness of transfer learning methods

We evaluated the performance of three TL approaches, FE, LoRA and adapters, across three different tasks (i.e., *AAV-sampled*, *GB1-three vs rest*, *Meltome-mixed*) considering five levels of layer depth (first layer only, 25%, 50%, 75%, full model) for different sizes of PLMs from three families (ESM2, ProGen2 and ProteinBERT) (Figure 1A-C). For the tasks *AAV-sampled* and *GB1-three vs rest*, we observed that performance plateaued when 25% of layers are used for all three TL setups, after which there are minimal gains or even drops in performance, suggesting that pre-trained parameters stored in the first quarter of a foundational model could be more suitable when used for tasks that include protein sequences with similar distribution (i.e. variants of a wild type) (Figure 1Ai-ii, Figure 1Bi-ii, Figure 1Ci-ii). As validated by earlier studies[25, 30], the last layer may not provide the optimal training features and in several setups even lead to substantial decrease in performance (Figure 1Ai,ii). This pattern is consistent across most TL configurations, with the exception of ProGen2-small and ProGen2-medium versions when adapters are applied for the *AAV-sampled* task (Figure 1Ci). In these instances, the last layers exhibited superior performance with a Spearman's rank correlation (ρ) that equals 0.86 for ProGen2-small and 0.82 for ProGen2-medium compared to using 25% of the full model (ρ = 0.59 and ρ = 0.60 respectively). For tasks involving diverse protein sequences like *Meltome-mixed*, incremental performance benefits are exhibited when deeper layers are targeted for TL (Figure 1Aiii-Ciii), indicating that the task's complexity and sequence variability benefits from richer and more comprehensive representations from the deeper layers of PLMs. Performance variance among PLMs, are relatively marginal, despite different architecture, model size and pre-training strategy. While larger models like ProGen2-xlarge and ESM2-15B perform slightly better in most setups, shallower models such as ProteinBERT and ProGen2-small achieve comparable results even in complex tasks.

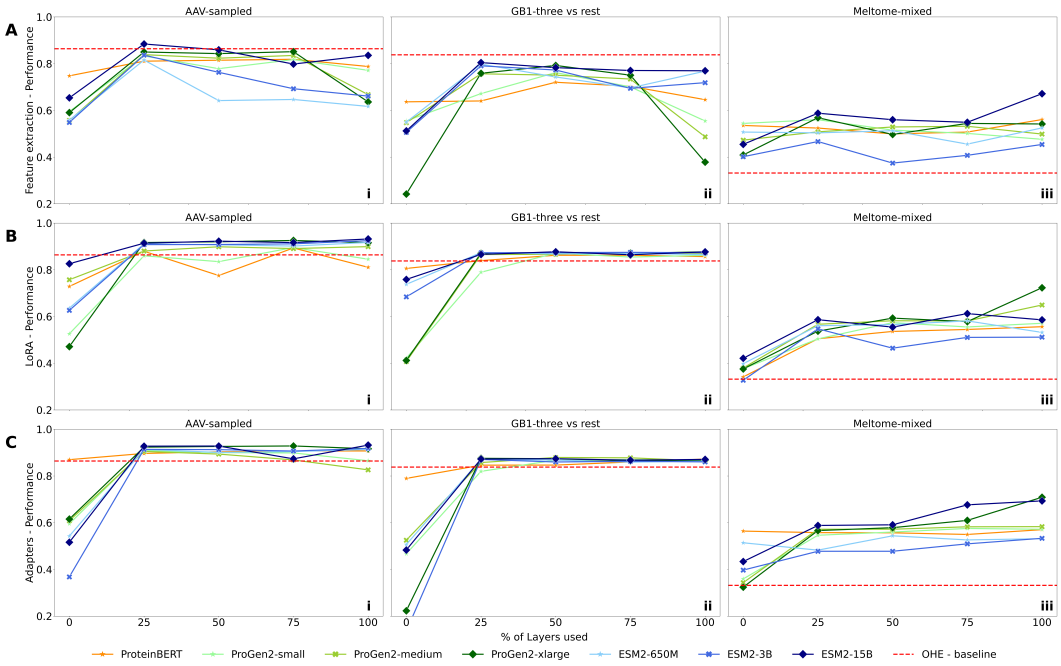

Figure 1: Performance analysis across, *AAV-sampled*, *GB1-three vs rest*, and *Meltome-mixed* tasks (columns i-iii) using different depth of PLMs' layers paired with TL methods, FE, LoRA and adapters (rows A-C). PLMs are differentiated by color, and performance curves are displayed for each task. For each subplot, x-axis shows the percentage of layers used - 0 (corresponds to using only the first layer of the model), 25%,50%,75%,100% (full model), and y-axis shows Spearman's correlation between predicted and ground truth values. The red dashed line represents a baseline model trained with the optimal hyperparameters using OHE sequences, (see chapter 3.5). PLM: Protein language models; TL: Transfer learning; OHE: One hot encoding, FE: Feature extraction; LoRA: Low rank adaptation.

2.3 Fine-tuning yields substantial performance gain in complex tasks

To evaluate the effectiveness of TL for five fitness prediction tasks (i.e., *AAV-sampled, AAV-one vs rest, GB1-three vs rest, GB1-one vs rest, Meltome-mixed*), we compared the performances of the best TL configurations for each PLM (Figure 2A). Configurations vary in regards to the fraction of PLM used (i.e., layer), training hyperparameters and downstream head (linear or single-layer neural network). In addition to FE, LoRA and adapters methods, we investigated the effect of FT only the last layer of the PLM, namely LoRA and adapters, as a computationally lighter alternative to the standard FT methods (see chapter 3.2). Overall, most TL techniques yielded superior results compared to baselines models trained with OHE of sequences. However, for the *AAV-sampled* and *GB1-three vs rest* tasks, the performance improvements were marginal, raising questions about the utility of PLMs and TL for these specific setups (Figure 2Ai, 3Aiii). These tasks are relatively simple because they involve sequences emerging from similar distributions (i.e., mutational variants of a starting protein sequence WT) for training and are validated on similarly structured data. Detailed results for each training setup can be found in the Supplementary materials' chapter Extended data.

Conversely, TL methods demonstrated the highest performance gains in diverse protein tasks (i.e., *Meltome-mixed*). This shows that PLMs excel in capturing distinct features across different protein families to accurately represent their amino acid sequences (Figure 2Av). Similarly, TL methods also significantly overperfromed in tasks where only single mutation variants were available during training, such as *AAV-one vs rest* and *GB1-one vs rest*, while sequences with higher ED were used for testing (Figure 2Aii, 3Aiv). Interestingly, for the same task, most FE models did not surpass the baseline model (Figure 2Aii), however, all FT methods appeared to recover and enhance performance, underscoring the importance of co-optimizing pre-trained PLM weights while incorporating task-specific knowledge. This effect is particularly evident in the *AAV-one vs rest* setup, where FE

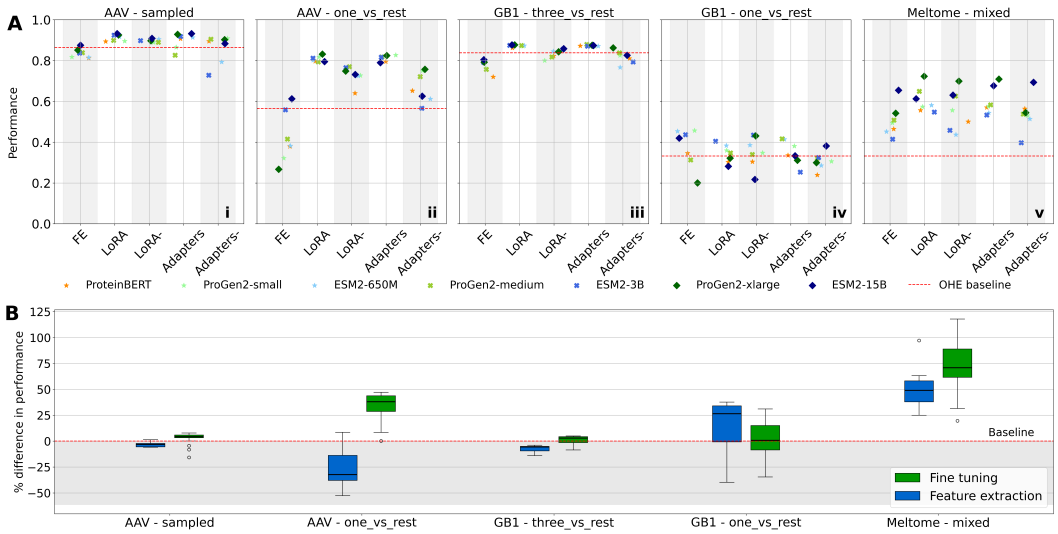

Figure 2: TL techniques performance (Spearman's correlation) comparison across five different tasks. (A) Spearman's correlation of the best performing PLM configuration in regards to layer, downstream head and pooling method used for each TL technique (x-axis), is being compared across each column (i-v): (i) *AAV-sampled*, (ii) *AAV-one vs rest*, (iii) *GB1-three vs rest*, (iv) *GB1-one vs rest*, and (v) *Meltome-mixed*. Different PLMs are used: ProteinBERT, ProGen2 (small, medium, xlarge), ESM2 (650M, 3B, 15B), with TL strategies including FE, LoRA, LoRA-, adapters, and adapters-. The red dashed line represents a baseline model trained with the optimal hyperparameters using OHE sequences, see Methods. (B) Percentage difference in performance relative to OHE baseline for FT (green) and FE (blue). Box plots display variability in performance gains across tasks and TL methods collectively for all PLMs and training configurations. PLM: Protein language models; TL: Transfer learning; OHE: One hot encoding; FE: Feature extraction; FT: Fine-tuning; LoRA: Low rank adaptation

techniques averaged a Spearman's rank correlation of 0.44, which is 21.5% below the baseline (ρ = 0.565), whereas FT models outperformed the baseline by an average 34% (average ρ = 0.75).

Figure 2B shows the distribution of FE and FT performance percentage changes compared to baseline models across the five tasks. Distribution in each box is calculated collectively for all PLMs and configurations for FE or FT methods and highlights the relationship between performance changes and TL category within different tasks. For the *AAV-sampled* and *GB1-three vs rest tasks*, both FE/FT approaches perform similarly and close to the baseline with median performance difference -3.2% / 3.95% and -5.64% / 2.89% respectively (Table S11). However, as task complexity increases, such as in the *Meltome-mixed*, FT yields substantial performance gains compared to FE. Specifically, FT can achieve an improvement of up to 117.82% over the baseline methods (Table S11). This is anticipated, as the increased number of parameters enhances the model's capacity to represent the diversity of proteins in this task. The TL configuration that performed best for each task is presented in Table 2.

2.4 Fine-tuning can generalize better to higher mutation variants when only labels for single mutations are available

Driven by the observation that fine-tuning PLMs can be particularly beneficial for *-one vs rest* splits, our investigation extended to examining the performance of the best configuration from each TL technique Table 2 across varying degrees of ED when trained only with single mutation variants. Figure 3 illustrates that TL-based models maintain more consistent performance levels as the ED increases compared to the respective baseline model which rapidly loses efficacy. Specifically, for the AAV and RBD datasets, performance slowly deteriorates at higher mutational levels, while the baseline models decline rapidly for the *AAV-one vs rest* (Figure 3A - red area) and no training was achievable for any OHE baseline model configuration for the *RBD-one vs rest task* (Figure 3B), likely due to the high sparsity introduced by OHE or the lack of capacity in logistic regression and single-layer neural networks architectures. Models fine-tuned with LoRA and adapters outperformed

Table 2: Scoreboard of the best transfer learning configuration results for each task.

Task	Score	Metric	Best configuration				
			PLM	TL-method	Layers used	Pooling	Downstream head
AAV - sampled	0.932	Spearman's correlation	ESM2-15B	Adapters	100%	Mean	Linear
AAV - one-vs-rest	0.831	Spearman's correlation	ProGen2-xlarge	LoRA	75%	CLS	Linear
GB1 - three-vs-rest	0.879	Spearman's correlation	ProGen2-medium	Adapters	50%	CLS	Linear
GB1 - one-vs-rest	0.457	Spearman's correlation	ProGen2-small	FE	75%	Mean	Linear
Meltome - mixed	0.723	Spearman's correlation	ProGen2-xlarge	LoRA	100%	Mean	Linear
RBD one-vs-rest	0.554	MCC	ProGen2-small	LoRA	50%	Mean	Linear
Trastuzumab one-vs-rest	0.390	MCC	ProGen2-small	LoRA-	50%	Mean	Linear

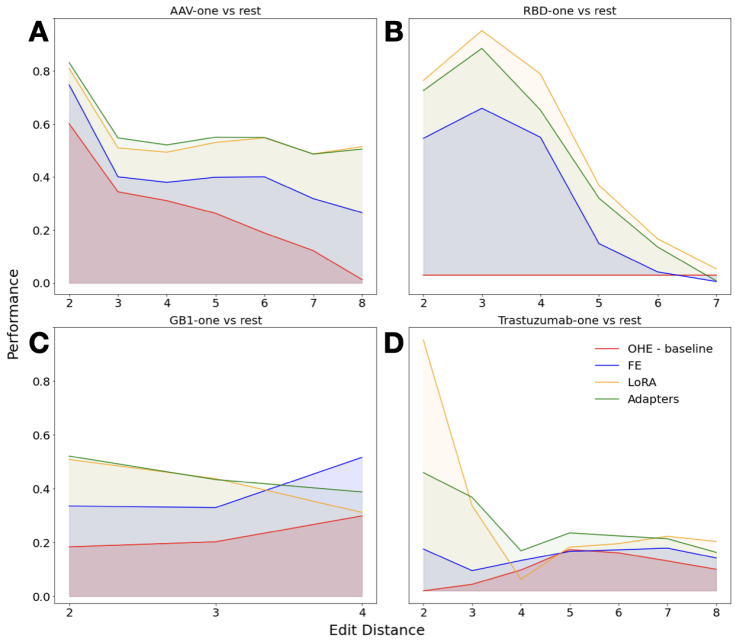

Figure 3: Performance trends relative to different edit distances for four different tasks: (A) *AAV-one vs rest*, (B) *RBD-one vs rest*, (C) *GB1-one vs rest*, and (D) *Trastuzumab-one vs rest*. The y-axis shows model performance (Spearman's correlation for GB1 and AAV dataset and MCC for RBD and Trastuzumab dataset), while the x-axis represents the ED from the reference sequence. Each line represents a different transfer learning strategy: FE (blue), LoRA (green), adapters (orange), and the OHE baseline (red). ED: Edit distance; FE: Feature extraction; LoRA: Low rank adaptation; MCC: Matthew's correlation coefficient; OHE: one hot encoding

both FE and baselines across all edit distances for the *GB1- one vs rest* and *Trastuzumab-one vs rest* tasks. Only exception was observed at $ED = 4$ for the GB1 dataset (Figure 3C, Figure 3D) where the FE model showed superior performance in higher edit distance; however, this deviation for the previous observation can likely be attributed to the relatively smaller training data size in the GB1 dataset ($n = 29$) (Table 1), raising concerns regarding the reliability of any trends. Similarly, for the Trastuzumab dataset, the limited number of sequences at lower edit distances Table 1)) compromises the reliability of predictions at higher edit distances, rendering the results less conclusive.

2.5 Practical guidelines for applying PLMFit

We compiled practical guidelines for the research community to effectively apply feature extraction and fine-tuning on PLMs, using the PLMFit platform. First, following the splitting of the dataset on training, validation, and testing sets, redundant (i.e., duplicates) and arbitrarily labeled (i.e.,

Table 3: Summary of protein language models used in this study

PLM	Type	No. of parameters	No. of layers	Embeddings Dim.
ProteinBERT	Masked LM	92 million	12	768
ProGen2-small	Causal LM	151 million	12	1024
ProGen2-medium	Causal LM	764 million	27	1536
ProGen2-xlarge	Causal LM	6.4 billion	32	4096
ESM2-650M	Masked LM	650 million	33	1280
ESM2-3B	Masked LM	3 billion	36	2560
ESM2-15B	Masked LM	15 billion	48	5120

same amino acid sequence with different label) sequences must be removed to prevent ambiguity during training. Depending on the amount of diversity required for the task of interest, data can be further clustered by sequence identity with protein clustering tools like MMSeq2[33] or kClust[34]. The main drivers of choosing TL approach and PLM are the diversity of training data, the level of accuracy required for the task and amount of accessible resources. Generally, tasks with small sequence variation between the training and testing sets, can be benefited by training a custom model ab initio. However, choosing the optimal architecture and tuning the hyperparameters may not be straightforward, despite theoretically being able to bring higher results. In that case, feature extraction e.g. ESM2-3B, ProGen-medium) using the 25% of a PLM with a linear downstream head could be sufficient and resource-efficient. Although fine-tuning a PLM can improve performance in this scenarios, the performance gains are often marginal.

For datasets characterized by single mutation variants, such as DMS experiments, custom models typically struggle to generalize to variants with higher edit distances. In these instances, using LoRA to fine-tune a fraction (e.g. 25-75%) of larger PLM (ESM2-15B, ProGen2-xlarge) is recommended to achieve better performance. Depending on the availability of GPUs, LoRA- technique (i.e. fine-tuning only the last layer) can serve as an effective compromise, offering adequate performance yield while mitigating the computational burden for this task.

In more complex tasks involving diverse protein sequences (e.g., *Meltome-mixed*), model performance tends to scale with size and amount of trainable parameters. Consequently, employing TL on PLMs for this type of tasks can improve performance, as the extensive knowledge embedded in these models from large datasets of natural proteins can be effectively leveraged. Particularly, applying LoRA on large-scale PLMs, like ESM2-15B or ProGen2-xlarge, is likely to be the most effective method. However, when GPU availability or inference speed is a limiting factor, fine-tuning smaller PLMs, such as ESM-3B or ProGen2-medium, offers a practical alternative that can provide satisfactory performance.

Deepspeed package with CPU offloading and mixed precision training is utilized in PLMFit to manage computational resources (Table S9). Stage 3 of Deepspeed is applied, with smaller reduce and all-gather bucket sizes for resource-constrained setups. These values can be adjusted for faster processing. PyTorch Lightning is used for easier integration with Deepspeed, providing a streamlined setup and cleaner code. Fine-tuning a PLM on a specific dataset using PLMFit is a streamlined process that can be executed with a single command.

3 Methods

3.1 Protein language models

TL techniques are applied to three state-of-the-art PLM families, two BERT-based (ESM2 and ProteinBERT) and one GPT-based (ProGen2), pre-trained with MLM and CLM objectives respectively. Different versions of these models are evaluated, covering a broad range of architecture size with their layers number spanning from 12 to 48, and their pre-trained parameters ranging from 92M to 15B respectively. Analytic overview of the PLMs assessed in this study is shown in Table 3.

3.2 Transfer Learning methods

As part of this study, both feature extraction and fine-tuning TL methods are investigated. We employ a layer pruning analysis assessing multiple fractions of the foundational models by extracting

embeddings from the first, the last and three intermediate layers corresponding to 25%, 50%, and 75% of the models' size. Two PEFT approaches were assessed to establish benchmarks, adapters and LoRA. Adapters are small architectures injected between the layers of a pre-trained PLM. For this study, adapters' architecture proposed by *Yang et. al*[27] is employed. LoRA decomposes the weight matrices of a pre-trained model into two low-rank matrices, significantly reducing the number of parameters to be trained. LoRA modules are applied on the pre-trained query, key and value matrices of the attention heads [8] in the different layers of the PLMs. Similarly to FE, the effect of adding FT-modules in different depths (first, last, intermediate; 25%, 50%, 75%) of PLMs is investigated. Additionally, to further decrease trainable parameters, we propose the addition of the respective modules only in the last layer of the foundational PLMs, namely LoRA- and adapters-.

3.3 Datasets and downstream tasks

Fitness prediction data were obtained from the widely used Fitness Landscape Inference for Proteins (FLIP) repository [17], which includes curated datasets mapping protein sequences to experimentally measured properties. Two datasets were used: Adeno-associated virus capsid (AAV) and GB1 domain of protein G (GB1), where each variant's fitness corresponds to its enrichment ratio [35, 36]. For AAV, we used the *-sampled* split, with random training/testing sequences, and the *-one vs rest* split, training on single mutation variants and testing on sequences with up to 39 mutations. For GB1, the *-one vs rest* and *-three vs rest* splits trained models with variants having up to three mutations, testing on sequences with up to four. For the thermostability prediction task, we used the Meltome dataset which includes diverse proteins clustered at 50% sequence similarity using MMSeq2 [33], and measures the maximum functional temperature (To). The *-mixed* split was used, with sequences from 13 species randomly selected for training. Binding classification tasks utilized two datasets [31, 32], one featuring the SARS-CoV-2 Omicron receptor-binding domain (RBD) screened for binding/escape interactions with human ACE2, and the other exploring mutations in the CDRH3 region of Trastuzumab for assessing HER2 binding or escape. Both used a *-one vs rest* split, training on single mutants ($ED = 1$) and testing on higher edit distances ($ED >= 2$). A summary of all datasets and splits is provided in Table 1.

3.4 Training and hyperparameter tuning

A large number of training setups (>2,900) was assessed in this study. For every training procedure Adam[37] optimizer is used with early stopping on best validation loss. All OHE baselines and FE-based models have been tuned for optimal hyperparameters (i.e. learning rate, batch size, weight decay) using bayesian optimization[38]. For LoRA, ranks of 4, 8, 16 and batch sizes of 2, 4, 6, 16, 32 were evaluated during hyperparameter tuning. Adapters modules parameter search space comprised of bottleneck dimensions of 16, 32, 64 and batch sizes of 4, 8, 16, 32. All methods implemented using PyTorch[39] coupled with the DeepSpeed (Table S9) package. Multiple hyperparameters have been assessed based on trial and error and existing literature. Pre-trained PLMs downloaded either from HuggingFace[40] or from their original repository and adjusted to allow high-throughput TL. Final training hyperparameters used for each TL setup are shown in Table S10. All codes and datasets are available at [Github will be shared after review for anonymization purposes].

3.5 Evaluation metrics and baselines

Performance of each TL-based model was compared to hyperparameter-tuned (Table S10) logistic regression and multi-layer perceptron (MLP) neural networks with one hidden layer (i.e., baselines), using one hot encoded (OHE) protein sequences as input. OHE is a method that represents each amino acid in a protein sequence as a binary vector, with all elements set to zero except for the position corresponding to that amino acid, which is set to one, capturing no biochemical properties or evolutionary relationships. Evaluation metrics used in this study vary, based on the nature of the task.; for models trained on regression tasks (i.e., enrichment ratio), we utilized Spearman's Rank correlation coefficient (ρ)[41] to assess the strength and direction of the monotonic relationship between predicted and actual values and for binary classification tasks (i.e, binding classification), we employed Matthew's Correlation Coefficient (MCC) as a metric [42], providing a balanced measure of the quality of binary classifications, taking into account true and false positives and negatives.

4 Discussion

In this study, we evaluated three TL approaches across five datasets, utilizing three families of PLMs. Each TL-based model was trained with varying parameters, including the number of layers used, pooling methods, downstream architecture, and training hyperparameters. Performance was compared against task-specific baselines trained with OHE sequences. The results indicate that TL can offer significant benefits in protein engineering tasks when an optimized configuration is being applied. FT, while improving performance across almost every task compared to FE, may not always be the optimal choice due to marginal performance gains relative to the computational cost it incurs. Nevertheless, FT proves particularly advantageous in limited training data scenarios and more challenging tasks, enabling models to generalize to unseen data by leveraging pre-training information. Conversely, when a sufficient amount of data is available, and the predictive task involves variants with mutations at positions similar to those in the training data (i.e. the distribution between the training and test sets remains relatively similar), the use of PLMs may be unnecessary, as a model trained ab-initio on the task-specific data could potentially offer comparable or even superior performance. Ultimately, the optimal use of pre-trained PLMs depends on the diversity and volume of training data, the specific nature of the task, and the computational resources available. Targeting specific layers within the foundational model not only enhances computational efficiency but can also boost performance, as embeddings from the final layer may not provide the most suitable representations for downstream tasks. Leveraging limited data or data including only single mutations to guide PLMs via TL can yield models that combine general knowledge acquired during pre-training (i.e. fitness and evolution) with task-specific data (e.g. experimental), making them capable of effectively generalizing to previously unseen protein sequences, typically in higher EDs. Such capabilities could be particularly valuable in leveraging Deep Mutational Scanning (DMS) experiments, which generate labeled libraries of single mutations across the entire or partial length of proteins. Fine-tuning pre-trained PLMs using these limited data can enable accurate predictions on novel combinatorial (i.e., $ED > 1$) libraries, without requiring extensive experimental workflows. We anticipate that PLMFit will serve as a valuable resource for the research community by offering a practical starting point for those seeking to leverage PLMs for various protein engineering tasks. Whether users aim to fine-tune pre-trained PLMs or extract embeddings from different layers, PLMFit can act as a tool to streamline these processes. Additionally, the platform can guide researchers in selecting appropriate parameters and configurations, enabling the practical application of PLMs in a range of biological tasks.

5 Limitations

While our evaluation provides insights at a great scale and depth, a more robust evaluation scheme, such as k-fold cross-validation, could offer a more comprehensive assessment of each transfer learning technique's performance. Additionally, we did not explore an extensive space of ranks for LoRA and more complex architectures for adapters, due to the high computational and time costs associated with these approaches. In the future, a broader hyperparameter space should be investigated to effectively determine the impact of each parameter. The datasets and tasks used in this study can carry inherent biases, which could influence the results. To validate the initial observations from this study, it is essential to include a more diverse range of data taking into consideration a broad range of protein families and tasks. Ultimately, TL-based models will need to transition from computational validation to real-world lab experiments to truly demonstrate their potential. We also encourage the community to contribute by uploading new datasets, tasks, transfer learning techniques, and additional pre-trained PLMs to help establish new benchmarks. Lastly, we intend to continuously update Table 2 with the highest performing combination of TL and PLM for different tasks to keep it relevant for ongoing research.

Acknowledgments and Disclosure of Funding

We would like to gratefully acknowledge ETH Zurich (Euler cluster) for providing computing resources and GPUS.
Funding: This research received no external funding.
Author contributions: Conceptualization, T.B., E.S., and S.T.R.; investigation, T.B., E.S.; software, T.B., E.S.; supervision, S.T.R.; writing- original draft T.B., E.S., and S.T.R.; Writing - review and editing: T.B., E.S., and S.T.R.

Competing Interests: The authors declare no competing interests.
Data and materials availability: All codes and datasets are available at https://github.com/LSSI-ETH/plmfit. All datasets can also be found in their original repo: AAV/GB1/Meltome (`https://github.com/J-SNACKKB/FLIP`), Trastuzumab (`https://github.com/dahjan/DMS_opt`) and RBD (`https://github.com/LSSI-ETH/Omicron_DML`)

References

[1] Partha Pratim Ray. ChatGPT: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope. *Internet of Things and Cyber-Physical Systems*, 3:121–154, January 2023.

[2] Hugo Touvron, Louis Martin, Kevin R Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, D Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, A Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M Kloumann, A Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *arXiv.org*, 2023.

[3] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023.

[4] Erik Nijkamp, Jeffrey A Ruffolo, Eli N Weinstein, Nikhil Naik, and Ali Madani. ProGen2: Exploring the boundaries of protein language models. *cels*, 14(11):968–978.e3, November 2023.

[5] Nadav Brandes, Dan Ofer, Yam Peleg, Nadav Rappoport, and Michal Linial. ProteinBERT: a universal deep-learning model of protein sequence and function. *Bioinformatics*, 38(8):2102–2110, April 2022.

[6] Ahmed Elnaggar, Hazem Essam, Wafaa Salah-Eldin, Walid Moustafa, Mohamed Elkerdawy, Charlotte Rochereau, and Burkhard Rost. Ankh : Optimized protein language model unlocks general-purpose modelling. *bioRxiv*, page 2023.01.16.524265, January 2023.

[7] Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, Debsindhu Bhowmik, and Burkhard Rost. ProtTrans: Toward understanding the language of life through self-supervised learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(10):7112–7127, October 2022.

[8] Ashish Vaswani, Noam M Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Adv. Neural Inf. Process. Syst.*, pages 5998–6008, June 2017.

[9] Brian L Hie, Varun R Shanker, Duo Xu, Theodora U J Bruun, Payton A Weidenbacher, Shaogeng Tang, Wesley Wu, John E Pak, and Peter S Kim. Efficient evolution of human antibodies from general protein language models. *Nat. Biotechnol.*, 42(2):275–283, April 2023.

[10] Orly Avraham, Tomer Tsaban, Ziv Ben-Aharon, Linoy Tsaban, and Ora Schueler-Furman. Protein language models can capture protein quaternary state. *BMC Bioinformatics*, 24(1):433, November 2023.

[11] Lucas Torroba Hennigen and Yoon Kim. Deriving language models from masked language models. *arXiv preprint arXiv:2305.15501*, 2023.

[12] Chu-Cheng Lin, Aaron Jaech, Xin Li, Matthew R Gormley, and Jason Eisner. Limitations of autoregressive models and their alternatives. *arXiv preprint arXiv:2010.11939*, 2020.

[13] Joshua Meier, Roshan Rao, Robert Verkuil, Jason Liu, Tom Sercu, and Alexander Rives. Language models enable zero-shot prediction of the effects of mutations on protein function. *bioRxiv*, page 2021.07.09.450648, November 2021.

[14] Karen Sargsyan and Carmay Lim. Using protein language models for protein interaction hot spot prediction with limited data. *BMC Bioinformatics*, 25(1):115, March 2024.

[15] Jeffrey A Ruffolo and Ali Madani. Designing proteins with language models. *Nat. Biotechnol.*, 42(2):200–202, February 2024.

[16] Roshan Rao, Nicholas Bhattacharya, Neil Thomas, Yan Duan, Xi Chen, John Canny, Pieter Abbeel, and Yun S Song. Evaluating protein transfer learning with TAPE. *Adv. Neural Inf. Process. Syst.*, 32:9689–9701, December 2019.

[17] Christian Dallago, Jody Mou, Kadina E Johnston, Bruce J Wittmann, Nicholas Bhattacharya, Samuel Goldman, Ali Madani, and Kevin K Yang. FLIP: Benchmark tasks in fitness landscape inference for proteins. *bioRxiv*, page 2021.11.09.467890, November 2021.

[18] B Rost and C Sander. Prediction of protein secondary structure at better than 70% accuracy. *J. Mol. Biol.*, 232(2):584–599, July 1993.

[19] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR, 2019.

[20] Chau Tran, Siddharth Khadkikar, and Aleksey Porollo. Survey of protein sequence embedding models. *Int. J. Mol. Sci.*, 24(4), February 2023.

[21] Thomas Hayes, Roshan Rao, Halil Akin, Nicholas James Sofroniew, Deniz Oktay, Zeming Lin, Robert Verkuil, Marius Wiggert, Rohil Badkundri, Irhum Shafkat, Jun Gong, Alexander Derry, Raul Santiago Molina, Neil Thomas, Yousuf Khan, Chetan Mishra, Carolyn Kim, Liam J Bartie, Salvatore Candido, and Alexander Rives. Simulating 500 million years of evolution with a language model.

[22] Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv [cs.CL]*, August 2023.

[23] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. *arXiv [cs.CL]*, June 2021.

[24] Chloe Hsu, Hunter Nisonoff, Clara Fannjiang, and Jennifer Listgarten. Combining evolutionary and assay-labelled data for protein fitness prediction. *bioRxiv*, page 2021.03.28.437402, March 2021.

[25] Francesca-Zhoufan Li, Ava P Amini, Yisong Yue, Kevin K Yang, and Alex X Lu. Feature reuse and scaling: Understanding transfer learning with protein language models. *bioRxiv*, page 2024.02.05.578959, February 2024.

[26] Samuel Sledzieski, Meghana Kshirsagar, Minkyung Baek, Rahul Dodhia, Juan Lavista Ferres, and Bonnie Berger. Democratizing protein language models with parameter-efficient fine-tuning. *Proceedings of the National Academy of Sciences*, 121(26):e2405840121, 2024.

[27] Wei Yang, Chun Liu, and Zheng Li. Lightweight fine-tuning a pretrained protein language model for protein secondary structure prediction. *https://papers.ssrn.com › ...https://papers.ssrn.com › ...*, April 2023.

[28] Bruce J Wittmann, Yisong Yue, and Frances H Arnold. Informed training set design enables efficient machine learning-assisted directed protein evolution. *Cell Syst*, 12(11):1026–1045.e7, November 2021.

[29] Robert Schmirler, Michael Heinzinger, and Burkhard Rost. Fine-tuning protein language models boosts predictions across diverse tasks. *Nat. Commun.*, 15(1):7407, August 2024.

[30] Lucrezia Valeriani, Diego Doimo, Francesca Cuturello, Alessandro Laio, Alessio Ansuini, and Alberto Cazzaniga. The geometry of hidden representations of large transformer models. *arXiv [cs.LG]*, February 2023.

[31] Lester Frei, Beichen Gao, Jiami Han, Joseph Michael Taft, Edward B Irvine, Cedric R Weber, Rachita Kumar, Benedikt Eisinger, and Sai T Reddy. Deep learning-guided selection of antibody therapies with enhanced resistance to current and prospective SARS-CoV-2 omicron variants. *bioRxiv*, pages 2023–2010, 2023.

[32] Derek M Mason, Simon Friedensohn, Cédric R Weber, Christian Jordi, Bastian Wagner, Simon M Meng, Roy A Ehling, Lucia Bonati, Jan Dahinden, Pablo Gainza, et al. Optimization of therapeutic antibodies by predicting antigen specificity from antibody sequence via deep learning. *Nature Biomedical Engineering*, 5(6):600–612, 2021.

[33] Martin Steinegger and Johannes Söding. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nat. Biotechnol.*, 35(11):1026–1028, November 2017.

[34] Maria Hauser, Christian E Mayer, and Johannes Söding. kclust: fast and sensitive clustering of large protein sequence databases. *BMC bioinformatics*, 14:1–12, 2013.

[35] Drew H Bryant, Ali Bashir, Sam Sinai, Nina K Jain, Pierce J Ogden, Patrick F Riley, George M Church, Lucy J Colwell, and Eric D Kelsic. Deep diversification of an AAV capsid protein by machine learning. *Nat. Biotechnol.*, 39(6):691–696, June 2021.

[36] Zachary Wu, S B Jennifer Kan, Russell D Lewis, Bruce J Wittmann, and Frances H Arnold. Machine learning-assisted directed protein evolution with combinatorial libraries. *Proc. Natl. Acad. Sci. U. S. A.*, 116(18):8852–8858, April 2019.

[37] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[38] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

[39] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[40] T Wolf. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[41] F Reading. Spearman rank correlation coefficient. *Concise Encycl Stat*, page 502, 2008.

[42] Davide Chicco and Giuseppe Jurman. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC Genomics*, 21:1–13, 1 2020.

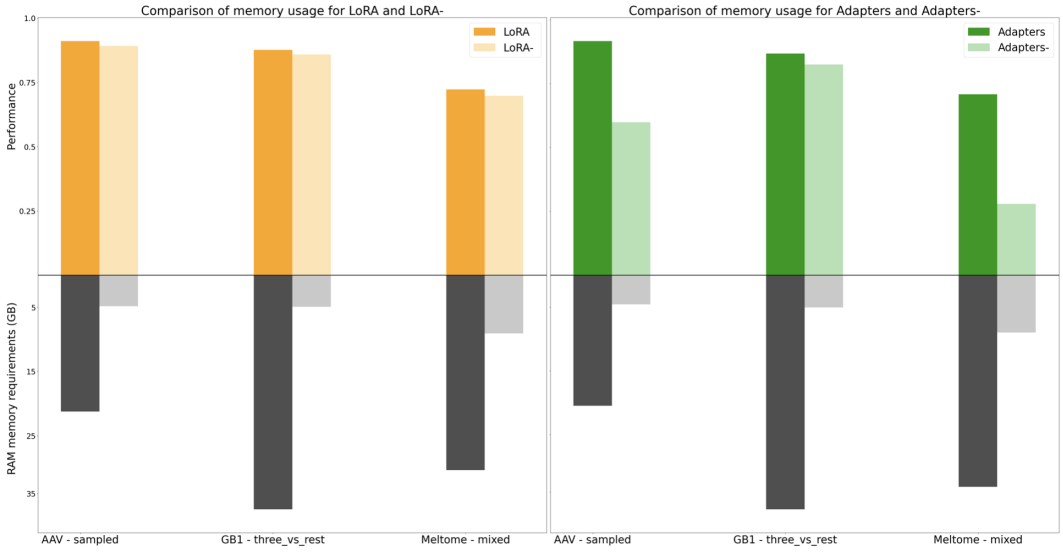

Figure S1: Performance and memory trade-off between LoRA/LoRA- and adapters/adapters- across three tasks. The orange bars (dark for LoRA and light for LoRA-) and the green bars (dark for adapters and light for adapters-) show the performance (Spearman's correlation) of the models on three tasks: *AAV-sampled*, *GB1-three vs rest*, and *Meltome-mixed*. The y-axis represents performance on a scale from 0 to 1, where LoRA consistently outperforms LoRA- across all datasets. The gray part of the bars (dark for default and light for - version) represents the GPU memory requirements in gigabytes (GB) for each approach. LoRA requires significantly more memory compared to LoRA-, illustrating a trade-off between performance and memory efficiency. LoRA: Low rank adaptation

A Supplementary Material

Memory requirements and performance trade-off when fine-tuning only the final layer

Our analysis demonstrated that fine-tuning techniques can yield superior results when adapting pre-trained PLMs for protein engineering tasks. Despite adopting PEFT methods, even when fine-tuning a small fraction of the model, the number of trainable parameters can still be large. This is due to the sheer size of models like ESM2-15B and ProGen2-xlarge, with 98 and 6.4 billion parameters respectively. In such models, updating even a very small subset requires significant computational resources. Motivated by these challenges, we investigated the performance of FT modules added only to the final layer of the entire PLM, namely, LoRA- and adapters-. Adopting this strategy, we reduced the number of trainable parameters by a fraction of the total layers number. Figure S1 illustrates a comparative analysis of performance (upper section) and memory usage (lower section) between LoRA, LoRA-, adapters, and adapters- across three tasks: *AAV-sampled*, *GB1-three vs rest*, and *Meltome-mixed*. However, both LoRA and adapters outperform their reduced counterparts, LoRA- and adapters-, the performance drops for *AAV-sampled* and *GB1-three vs rest* are marginal. Only *Meltome-mixed* exhibits a more notable decline when adapters- method is applied. Importantly, both LoRA- and adapters- have significantly lower memory requirements, without a drastic performance loss. GPU RAM memory ranges from 6.5-12.8 gigabytes (GB), compared to the standard application of these methods which require 28-52 GB Table S1. By reducing the memory requirements without significantly sacrificing performance, LoRA- and adapters- provide accessibility to larger PLMs democratizing advanced model FT and allowing a broader range of institutions and researchers to fine tune PLMs without the need for expensive cloud computing services or specialized infrastructure. PLMs, training parameters and hardware resources used for LoRA- and adapters- for the comparative analysis and for the entirety of setups are shown in Table S1 and Tables S3, S5 and S6 respectively.

Table S1: Comparative analysis demonstrating the performance and memory trade-off between LoRA/LoRA- and adapters/adapters- across three tasks. Best performing setups are selected for each of the three largest pre-trained PLMs assessed in this study, where each PLM is assigned one of the three tasks; *AAV-sampled*, *GB1-three vs rest*, and *Meltome-mixed*. Performance is measured using Spearman's rank correlation coefficient. PLM: Protein Language Model; LoRA: Low rank adaptation

Tasks	PLM	Method	Performance	GPU RAM requirements (GB)	GPU Used
AAV - sampled	ESM-3B	LoRA	0.911	28	NVIDIA A100 40GB (x4)
		LoRA-	0.891	6.4	NVIDIA Quadro RTX 6000 24G (x4)
		Adapters	0.917	29	NVIDIA A100 80GB (x4)
		Adapters-	0.599	6.5	NVIDIA Quadro RTX 6000 24G (x4)
GB1 - three vs rest	ESM2-15B	LoRA	0.876	48	NVIDIA A100 80GB (x4)
		LoRA-	0.858	6.5	NVIDIA Quadro RTX 6000 24G (x2)
		Adapters	0.868	52	NVIDIA A100 80GB (x4)
		Adapters-	0.825*	7.2	NVIDIA Quadro RTX 6000 24G (x2)
Meltome - mixed	ProGen2-xlarge	LoRA	0.723	40	NVIDIA A100 80GB (x4)
		LoRA-	0.698	12	NVIDIA Quadro RTX 6000 24G (x4)
		Adapters	0.708	47	NVIDIA A100 80GB (x4)
		Adapters-	0.279	12.8	NVIDIA Quadro RTX 6000 24G (x4)

Downstream heads architectures

Outputs from different PLMs' encoder layers are used as representations of protein sequence. The original decoder has been discarded and replaced with the task specific downstream head using these embeddings as input features for training. Transformer-based encoder outputs are 2-d matrices ($V_{local}\,\epsilon\mathbb{R}^{sequence\,length\times embedding\,dimension}$) where each residue (i.e, token) is described by a 1-d numerical vector ($V_{global}\,\epsilon\mathbb{R}^{embedding\,dimension}$). Prior to inputting these representations into the downstream architecture, it is necessary to transition from local (i.e. token-wise) to global (i.e. sequence-wise) representations, thereby transforming the entire sequence into a feature vector. To achieve this, multiple reduction approaches can be applied. Within the scope of this study, two reduction techniques were assessed, mean- and classification token- pooling. Specifically, for BERT-based PLMs (ESM2, ProteinBERT), classification token-pooling involves gathering the embeddings from the first token prepended in every sequence, also called CLS token, and for GPT-based PLMs (ProGen2), this token involves selecting the embeddings of the last token, a special token appended in the end of each sequence called EOS. By respectively averaging the elements towards the sequence length dimension for each position through the embedding dimension or selecting the embeddings of a token that holds information about the whole sequence, 2-d matrices (V_{local}) are transformed to a 1-d vector (V_{global}). Leveraging global representations of protein sequences, deep learning-based architectures are trained to address specific tasks. We evaluated two shallow architectures, logistic regression and a two-layer multi-layer perceptron (MLP). Our focus was on highlighting the information encapsulated in the PLMs' embeddings, thus we utilized architectures that do not require extensive optimization as downstream models.

Hardware resources

High-performance computing clusters were used for all experiments, with hardware configurations varying based on dataset size and transfer learning techniques. Multiple Nvidia GPUs (GeForce RTX 2080 Ti, RTX 3090, RTX 4090, TITAN RTX, Quadro RTX 6000, Tesla A100) were utilized for inference and backpropagation, with 1 to 4 GPUs used in parallel to accelerate training. Tables S2 to S8 list the detailed resources used for each setup. Different pooling techniques require the same amount of resources and are therefore combined.

Extended data

We present an extended representation of the results, including Table S11 which provides statistical summaries of the box plots in Figure 1B and heatmaps in figs. S2 to S8, displaying the result metrics for all 2,940 TL setups evaluated.

Table S2: Resources used for each TL setup on *AAV-one vs rest* task.

PLM + Layer	Embeddings Extraction		Feature Extraction		LoRA		LoRA-		Adapters		Adapters-	
	GPUs	CPU RAM	GPUs	CPU RAM	GPUs	CPU RAM	GPUs	CPU RAM	GPUs	CPU RAM	GPUs	CPU RAM
ProteinBERT - 1st	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB
25%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB
50%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB
75%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB
All/Last	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB
ProGen2-small - 1st	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB
25%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB
50%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB
75%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB
All/Last	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB
ESM2-650M - 1st	1 x RTX 4090	12GB	1 x RTX 4090	14GB	1 x RTX 3090	18GB	1 x RTX 3090	18GB	1 x RTX 3090	18GB	1 x RTX 3090	18GB
25%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	1 x RTX 3090	18GB	1 x RTX 3090	18GB	1 x RTX 3090	18GB	1 x RTX 3090	18GB
50%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x RTX 3090	18GB	2 x RTX 4090	18GB	2 x RTX 4090	18GB	2 x RTX 4090	18GB
75%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x TITAN RTX	18GB	2 x Quadro RTX 6000	18GB	2 x Quadro RTX 6000	18GB	2 x Quadro RTX 6000	18GB
All/Last	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x TITAN RTX	18GB	2 x Quadro RTX 6000	18GB	2 x RTX 4090	18GB	2 x Quadro RTX 6000	18GB
ProGen2-medium - 1st	1 x RTX 4090	12GB	1 x RTX 4090	14GB	1 x RTX 3090	18GB	1 x RTX 2080 Ti	18GB	1 x RTX 3090	18GB	1 x RTX 2080 Ti	18GB
25%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	1 x RTX 3090	18GB	1 x RTX 2080 Ti	18GB	1 x RTX 3090	18GB	1 x RTX 2080 Ti	18GB
50%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x RTX 3090	18GB	2 x RTX 2080 Ti	18GB	2 x RTX 3090	18GB	2 x RTX 2080 Ti	18GB
75%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x TITAN RTX	18GB	2 x RTX 4090	18GB	2 x TITAN RTX	18GB	2 x RTX 4090	18GB
All/Last	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x TITAN RTX	18GB	2 x RTX 4090	18GB	2 x TITAN RTX	18GB	2 x RTX 4090	18GB
ESM2-3B - 1st	1 x RTX 4090	30GB	1 x RTX 4090	14GB	1 x RTX 3090	18GB	1 x RTX 3090	18GB	1 x RTX 2080 Ti	18GB	1 x RTX 3090	18GB
25%	1 x RTX 4090	30GB	1 x RTX 4090	14GB	1 x RTX 3090	18GB	1 x RTX 4090	18GB	1 x RTX 3090	18GB	1 x RTX 4090	18GB
50%	1 x RTX 4090	30GB	1 x RTX 4090	14GB	2 x RTX 3090	18GB	1 x Quadro RTX 6000	18GB	1 x RTX 3090	18GB	1 x Quadro RTX 6000	18GB
75%	1 x RTX 4090	30GB	1 x RTX 4090	14GB	2 x TITAN RTX	18GB	1 x Quadro RTX 6000	18GB	2 x A100 (40 GiB)	18GB	1 x Quadro RTX 6000	18GB
All/Last	1 x RTX 4090	30GB	1 x RTX 4090	14GB	2 x TITAN RTX	18GB	1 x A100 (40 GiB)	18GB	3 x A100 (40 GiB)	18GB	1 x A100 (40 GiB)	18GB
ProGen2-xlarge - 1st	1 x RTX 4090	60GB	1 x RTX 4090	14GB	1 x RTX 2080 Ti	40GB	1 x RTX 2080 Ti	40GB	1 x RTX 2080 Ti	40GB	1 x RTX 2080 Ti	40GB
25%	1 x RTX 4090	60GB	1 x RTX 4090	14GB	1 x RTX 3090	40GB	1 x RTX 3090	40GB	1 x RTX 3090	40GB	1 x RTX 3090	40GB
50%	1 x RTX 4090	60GB	1 x RTX 4090	14GB	2 x RTX 3090	40GB	2 x RTX 4090	40GB	1 x RTX 3090	40GB	2 x RTX 4090	40GB
75%	1 x RTX 4090	60GB	1 x RTX 4090	14GB	2 x A100 (40 GiB)	40GB	2 x RTX 4090	40GB	2 x A100 (40 GiB)	40GB	2 x RTX 4090	40GB
All/Last	1 x A100 (80 GiB)	60GB	1 x RTX 4090	14GB	2 x A100 (80 GiB)	40GB	3 x TITAN RTX	40GB	3 x A100 (40 GiB)	40GB	3 x TITAN RTX	40GB
ESM2-15B - 1st	1 x RTX 4090	80GB	1 x RTX 4090	14GB	1 x RTX 2080 Ti	70GB	1 x RTX 3090	70GB	1 x RTX 2080 Ti	75GB	1 x RTX 3090	75GB
25%	1 x RTX 4090	80GB	1 x RTX 4090	14GB	1 x RTX 3090	70GB	1 x RTX 4090	70GB	2 x RTX 3090	75GB	2 x RTX 4090	75GB
50%	1 x RTX 4090	80GB	1 x RTX 4090	14GB	2 x A100 (40 GiB)	70GB	2 x Quadro RTX 6000	70GB	3 x A100 (40 GiB)	75GB	2 x Quadro RTX 6000	75GB
75%	1 x A100 (80 GiB)	80GB	1 x RTX 4090	14GB	3 x A100 (80 GiB)	70GB	2 x Quadro RTX 6000	70GB	3 x A100 (80 GiB)	75GB	2 x Quadro RTX 6000	75GB
All/Last	1 x A100 (80 GiB)	80GB	1 x RTX 4090	14GB	3 x A100 (80 GiB)	70GB	2 x A100 (40 GiB)	70GB	4 x A100 (80 GiB)	75GB	2 x A100 (80 GiB)	75GB

Table S3: Resources used for each TL setup on *AAV-sampled* task.

PLM + Layer	Embeddings Extraction		Feature Extraction		LoRA		LoRA-		Adapters		Adapters-	
	GPUs	CPU RAM	GPUs	CPU RAM	GPUs	CPU RAM	GPUs	CPU RAM	GPUs	CPU RAM	GPUs	CPU RAM
ProteinBERT - 1st	1 x RTX 4090	12GB	1 x RTX 4090	10GB	2 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	2 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB
25%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	2 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	2 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB
50%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	3 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	3 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB
75%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	3 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	3 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB
All/Last	1 x RTX 4090	12GB	1 x RTX 4090	10GB	3 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	3 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB
ProGen2-small - 1st	1 x RTX 4090	12GB	1 x RTX 4090	10GB	2 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	2 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB
25%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	2 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	2 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB
50%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	3 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	3 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB
75%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	3 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	3 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB
All/Last	1 x RTX 4090	12GB	1 x RTX 4090	10GB	3 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	3 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB
ESM2-650M - 1st	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x RTX 4090	12GB	1 x RTX 3090	12GB	2 x RTX 4090	18GB	1 x RTX 3090	18GB
25%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x RTX 4090	12GB	1 x RTX 3090	12GB	2 x RTX 4090	18GB	1 x RTX 3090	18GB
50%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	3 x RTX 4090	12GB	2 x RTX 4090	12GB	3 x RTX 4090	18GB	2 x RTX 4090	18GB
75%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	4 x RTX 4090	12GB	2 x Quadro RTX 6000	12GB	4 x RTX 4090	18GB	2 x Quadro RTX 6000	18GB
All/Last	1 x RTX 4090	12GB	1 x RTX 4090	14GB	4 x Quadro RTX 6000	12GB	2 x Quadro RTX 6000	12GB	4 x Quadro RTX 6000	18GB	2 x Quadro RTX 6000	18GB
ProGen2-medium - 1st	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x RTX 4090	12GB	1 x RTX 3090	12GB	2 x RTX 4090	18GB	1 x RTX 3090	18GB
25%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x RTX 4090	12GB	1 x RTX 3090	12GB	2 x RTX 4090	18GB	1 x RTX 3090	18GB
50%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	3 x RTX 4090	12GB	2 x RTX 4090	12GB	3 x RTX 4090	18GB	2 x RTX 4090	18GB
75%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	4 x RTX 4090	12GB	2 x RTX 4090	12GB	4 x RTX 4090	18GB	2 x RTX 4090	18GB
All/Last	1 x RTX 4090	12GB	1 x RTX 4090	14GB	4 x Quadro RTX 6000	12GB	2 x RTX 4090	12GB	4 x Quadro RTX 6000	18GB	2 x RTX 4090	18GB
ESM2-3B - 1st	1 x RTX 4090	30GB	1 x RTX 4090	14GB	2 x TITAN RTX	40GB	2 x RTX 3090	40GB	2 x TITAN RTX	18GB	2 x RTX 3090	18GB
25%	1 x RTX 4090	30GB	1 x RTX 4090	14GB	2 x TITAN RTX	40GB	3 x RTX 4090	40GB	2 x TITAN RTX	18GB	3 x RTX 4090	18GB
50%	1 x RTX 4090	30GB	1 x RTX 4090	14GB	3 x RTX 4090	40GB	4 x TITAN RTX	40GB	4 x RTX 4090	18GB	4 x TITAN RTX	18GB
75%	1 x RTX 4090	30GB	1 x RTX 4090	14GB	3 x A100 (40 GiB)	40GB	4 x TITAN RTX	40GB	3 x A100 (40 GiB)	18GB	4 x TITAN RTX	18GB
All/Last	1 x RTX 4090	30GB	1 x RTX 4090	14GB	4 x A100 (40 GiB)	40GB	4 x Quadro RTX 6000	40GB	4 x A100 (80 GiB)	18GB	4 x Quadro RTX 6000	18GB
ProGen2-xlarge - 1st	1 x RTX 4090	60GB	1 x RTX 4090	14GB	2 x TITAN RTX	40GB	2 x RTX 3090	40GB	2 x TITAN RTX	40GB	2 x RTX 3090	40GB
25%	1 x RTX 4090	60GB	1 x RTX 4090	14GB	2 x TITAN RTX	40GB	3 x RTX 4090	40GB	2 x TITAN RTX	40GB	3 x RTX 4090	40GB
50%	1 x RTX 4090	60GB	1 x RTX 4090	14GB	4 x RTX 4090	40GB	4 x TITAN RTX	40GB	4 x RTX 4090	40GB	4 x TITAN RTX	40GB
75%	1 x RTX 4090	60GB	1 x RTX 4090	14GB	4 x A100 (40 GiB)	40GB	4 x TITAN RTX	40GB	4 x A100 (40 GiB)	40GB	4 x TITAN RTX	40GB
All/Last	1 x A100 (80 GiB)	60GB	1 x RTX 4090	14GB	4 x A100 (40 GiB)	40GB	4 x Quadro RTX 6000	40GB	4 x A100 (40 GiB)	40GB	4 x Quadro RTX 6000	40GB
ESM2-15B - 1st	1 x RTX 4090	80GB	1 x RTX 4090	14GB	2 x TITAN RTX	70GB	1 x RTX 3090	70GB	2 x TITAN RTX	75GB	2 x RTX 3090	75GB
25%	1 x RTX 4090	80GB	1 x RTX 4090	14GB	3 x TITAN RTX	70GB	2 x Quadro RTX 6000	70GB	3 x TITAN RTX	75GB	2 x TITAN RTX	75GB
50%	1 x RTX 4090	80GB	1 x RTX 4090	14GB	3 x A100 (40 GiB)	70GB	3 x Quadro RTX 6000	70GB	3 x A100 (40 GiB)	75GB	3 x TITAN RTX	75GB
75%	1 x A100 (80 GiB)	80GB	1 x RTX 4090	14GB	3 x A100 (80 GiB)	70GB	4 x Quadro RTX 6000	70GB	4 x A100 (80 GiB)	75GB	4 x A100 (80 GiB)	75GB
All/Last	1 x A100 (80 GiB)	80GB	1 x RTX 4090	14GB	4 x A100 (80 GiB)	70GB	4 x A100 (80 GiB)	70GB	4 x A100 (80 GiB)	75GB	4 x Quadro RTX 6000	75GB

Table S4: Resources used for each TL setup on *GB1-one vs rest* task.

PLM + Layer	Embeddings Extraction		Feature Extraction		LoRA		LoRA-		Adapters		Adapters-	
	GPUs	CPU RAM	GPUs	CPU RAM	GPUs	CPU RAM	GPUs	CPU RAM	GPUs	CPU RAM	GPUs	CPU RAM
ProteinBERT - 1st	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 4090	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 4090	8GB	1 x RTX 2080 Ti	8GB
25%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 4090	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 4090	8GB	1 x RTX 2080 Ti	8GB
50%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 4090	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 4090	8GB	1 x RTX 2080 Ti	8GB
75%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 4090	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 4090	8GB	1 x RTX 2080 Ti	8GB
All/Last	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 4090	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 4090	8GB	1 x RTX 2080 Ti	8GB
ProGen2-small - 1st	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 4090	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 4090	12GB	1 x RTX 2080 Ti	12GB
25%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 4090	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 4090	12GB	1 x RTX 2080 Ti	12GB
50%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 4090	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 4090	12GB	1 x RTX 2080 Ti	12GB
75%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 4090	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 4090	12GB	1 x RTX 2080 Ti	12GB
All/Last	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 4090	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 4090	12GB	1 x RTX 2080 Ti	12GB
ESM2-650M - 1st	1 x RTX 4090	12GB	1 x RTX 4090	14GB	1 x RTX 3090	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 4090	18GB	1 x RTX 2080 Ti	18GB
25%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	1 x RTX 3090	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 4090	18GB	1 x RTX 2080 Ti	18GB
50%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x RTX 4090	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 4090	18GB	1 x RTX 2080 Ti	18GB
75%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x Quadro RTX 6000	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 4090	18GB	1 x RTX 2080 Ti	18GB
All/Last	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x Quadro RTX 6000	12GB	2 x RTX 3090	12GB	2 x RTX 3090	18GB	2 x RTX 3090	18GB
ProGen2-medium - 1st	1 x RTX 4090	12GB	1 x RTX 4090	14GB	1 x RTX 4090	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 4090	18GB	1 x RTX 2080 Ti	18GB
25%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	1 x RTX 4090	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 4090	18GB	1 x RTX 2080 Ti	18GB
50%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 4090	18GB	1 x RTX 2080 Ti	18GB
75%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x TITAN RTX	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 4090	18GB	1 x RTX 2080 Ti	18GB
All/Last	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x TITAN RTX	12GB	2 x RTX 3090	12GB	2 x RTX 3090	18GB	2 x RTX 3090	18GB
ESM2-3B - 1st	1 x RTX 4090	30GB	1 x RTX 4090	14GB	1 x RTX 3090	40GB	1 x RTX 2080 Ti	40GB	1 x RTX 4090	18GB	1 x RTX 2080 Ti	18GB
25%	1 x RTX 4090	30GB	1 x RTX 4090	14GB	1 x RTX 4090	40GB	1 x RTX 2080 Ti	40GB	1 x RTX 4090	18GB	1 x RTX 2080 Ti	18GB
50%	1 x RTX 4090	30GB	1 x RTX 4090	14GB	1 x Quadro RTX 6000	40GB	1 x RTX 2080 Ti	40GB	1 x RTX 4090	18GB	1 x RTX 2080 Ti	18GB
75%	1 x RTX 4090	30GB	1 x RTX 4090	14GB	1 x Quadro RTX 6000	40GB	2 x RTX 3090	40GB	2 x TITAN RTX	18GB	2 x RTX 3090	18GB
All/Last	1 x RTX 4090	30GB	1 x RTX 4090	14GB	1 x A100 (40 GiB)	40GB	2 x TITAN RTX	40GB	2 x RTX 3090	18GB	2 x TITAN RTX	18GB
ProGen2-xlarge - 1st	1 x RTX 4090	60GB	1 x RTX 4090	14GB	1 x RTX 2080 Ti	40GB	1 x RTX 2080 Ti	40GB	1 x RTX 2080 Ti	40GB	1 x RTX 2080 Ti	40GB
25%	1 x RTX 4090	60GB	1 x RTX 4090	14GB	1 x RTX 2080 Ti	40GB	1 x RTX 2080 Ti	40GB	1 x RTX 2080 Ti	40GB	1 x RTX 2080 Ti	40GB
50%	1 x RTX 4090	60GB	1 x RTX 4090	14GB	1 x RTX 2080 Ti	40GB	1 x RTX 2080 Ti	40GB	1 x RTX 3090	40GB	1 x RTX 2080 Ti	40GB
75%	1 x RTX 4090	60GB	1 x RTX 4090	14GB	1 x RTX 2080 Ti	40GB	2 x RTX 3090	40GB	2 x TITAN RTX	40GB	2 x RTX 3090	40GB
All/Last	1 x A100 (80 GiB)	60GB	1 x RTX 4090	14GB	2 x TITAN RTX	40GB	2 x TITAN RTX	40GB	2 x RTX 3090	40GB	2 x TITAN RTX	40GB
ESM2-15B - 1st	1 x RTX 4090	80GB	1 x RTX 4090	14GB	1 x RTX 3090	70GB	1 x RTX 2080 Ti	70GB	1 x RTX 2080 Ti	75GB	1 x RTX 2080 Ti	75GB
25%	1 x RTX 4090	80GB	1 x RTX 4090	14GB	1 x RTX 4090	70GB	1 x RTX 2080 Ti	70GB	1 x RTX 2080 Ti	75GB	1 x RTX 2080 Ti	75GB
50%	1 x RTX 4090	80GB	1 x RTX 4090	14GB	1 x Quadro RTX 6000	70GB	1 x RTX 4090	70GB	1 x RTX 3090	75GB	2 x RTX 4090	75GB
75%	1 x A100 (80 GiB)	80GB	1 x RTX 4090	14GB	1 x Quadro RTX 6000	70GB	2 x TITAN RTX	70GB	2 x A100 (40 GiB)	75GB	2 x TITAN RTX	75GB
All/Last	1 x A100 (80 GiB)	80GB	1 x RTX 4090	14GB	1 x A100 (40 GiB)	70GB	2 x A100 (40 GiB)	70GB	2 x A100 (80 GiB)	75GB	2 x A100 (80 GiB)	75GB

Table S5: Resources used for each TL setup on *GB1-three vs rest* task.

PLM + Layer	Embeddings Extraction		Feature Extraction		LoRA		LoRA-		Adapters		Adapters-	
	GPUs	CPU RAM	GPUs	CPU RAM	GPUs	CPU RAM	GPUs	CPU RAM	GPUs	CPU RAM	GPUs	CPU RAM
ProteinBERT - 1st	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB
25%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB
50%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB
75%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB
All/Last	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB
ProGen2-small - 1st	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB
25%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB
50%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB
75%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB
All/Last	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB
ESM2-650M - 1st	1 x RTX 4090	12GB	1 x RTX 4090	14GB	1 x RTX 4090	12GB	1 x RTX 3090	12GB	1 x RTX 3090	18GB	1 x RTX 3090	18GB
25%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	1 x RTX 4090	12GB	1 x RTX 3090	12GB	1 x RTX 4090	18GB	1 x RTX 3090	18GB
50%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x RTX 4090	12GB	1 x RTX 4090	12GB	2 x RTX 4090	18GB	1 x RTX 4090	18GB
75%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x Quadro RTX 6000	12GB	1 x RTX 4090	12GB	2 x Quadro RTX 6000	18GB	1 x RTX 4090	18GB
All/Last	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x Quadro RTX 6000	12GB	1 x RTX 4090	12GB	2 x Quadro RTX 6000	18GB	1 x RTX 4090	18GB
ProGen2-medium - 1st	1 x RTX 4090	12GB	1 x RTX 4090	14GB	1 x RTX 3090	12GB	1 x RTX 3090	12GB	1 x RTX 3090	18GB	1 x RTX 3090	18GB
25%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	1 x RTX 3090	12GB	1 x RTX 3090	12GB	1 x RTX 3090	18GB	1 x RTX 3090	18GB
50%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x RTX 4090	12GB	1 x RTX 4090	12GB	2 x RTX 4090	18GB	1 x RTX 4090	18GB
75%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x Quadro RTX 6000	12GB	1 x RTX 4090	12GB	2 x Quadro RTX 6000	18GB	1 x RTX 4090	18GB
All/Last	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x Quadro RTX 6000	12GB	1 x RTX 4090	12GB	2 x Quadro RTX 6000	18GB	1 x RTX 4090	18GB
ESM2-3B - 1st	1 x RTX 4090	30GB	1 x RTX 4090	14GB	1 x RTX 4090	40GB	1 x RTX 3090	40GB	1 x RTX 3090	18GB	1 x RTX 3090	18GB
25%	1 x RTX 4090	30GB	1 x RTX 4090	14GB	1 x RTX 4090	40GB	1 x RTX 4090	40GB	1 x RTX 4090	18GB	1 x RTX 4090	18GB
50%	1 x RTX 4090	30GB	1 x RTX 4090	14GB	1 x Quadro RTX 6000	40GB	1 x Quadro RTX 6000	40GB	4 x Quadro RTX 6000	18GB	2 x Quadro RTX 6000	18GB
75%	1 x RTX 4090	30GB	1 x RTX 4090	14GB	1 x Quadro RTX 6000	40GB	1 x Quadro RTX 6000	40GB	4 x Quadro RTX 6000	18GB	2 x Quadro RTX 6000	18GB
All/Last	1 x RTX 4090	30GB	1 x RTX 4090	14GB	1 x A100 (40 GiB)	40GB	2 x Quadro RTX 6000	40GB	4 x Quadro RTX 6000	18GB	2 x Quadro RTX 6000	18GB
ProGen2-xlarge - 1st	1 x RTX 4090	60GB	1 x RTX 4090	14GB	1 x RTX 3090	40GB	1 x RTX 3090	40GB	1 x RTX 3090	40GB	1 x RTX 3090	40GB
25%	1 x RTX 4090	60GB	1 x RTX 4090	14GB	1 x RTX 4090	40GB	1 x RTX 4090	40GB	1 x RTX 4090	40GB	1 x RTX 4090	40GB
50%	1 x RTX 4090	60GB	1 x RTX 4090	14GB	1 x Quadro RTX 6000	40GB	1 x Quadro RTX 6000	40GB	4 x Quadro RTX 6000	40GB	2 x Quadro RTX 6000	40GB
75%	1 x RTX 4090	60GB	1 x RTX 4090	14GB	1 x Quadro RTX 6000	40GB	1 x Quadro RTX 6000	40GB	4 x Quadro RTX 6000	40GB	2 x Quadro RTX 6000	40GB
All/Last	1 x A100 (80 GiB)	60GB	1 x RTX 4090	14GB	1 x Quadro RTX 6000	40GB	2 x Quadro RTX 6000	40GB	4 x Quadro RTX 6000	40GB	2 x Quadro RTX 6000	40GB
ESM2-15B - 1st	1 x RTX 4090	80GB	1 x RTX 4090	14GB	1 x RTX 3090	70GB	1 x RTX 3090	70GB	1 x RTX 3090	75GB	1 x RTX 3090	75GB
25%	1 x RTX 4090	80GB	1 x RTX 4090	14GB	2 x RTX 4090	70GB	2 x RTX 4090	70GB	2 x RTX 4090	75GB	1 x RTX 4090	75GB
50%	1 x RTX 4090	80GB	1 x RTX 4090	14GB	3 x A100 (40 GiB)	70GB	2 x RTX 4090	70GB	4 x A100 (40 GiB)	75GB	2 x RTX 4090	75GB
75%	1 x A100 (80 GiB)	80GB	1 x RTX 4090	14GB	3 x A100 (80 GiB)	70GB	2 x Quadro RTX 6000	70GB	4 x A100 (80 GiB)	75GB	2 x Quadro RTX 6000	75GB
All/Last	1 x A100 (80 GiB)	80GB	1 x RTX 4090	14GB	4 x A100 (80 GiB)	70GB	2 x Quadro RTX 6000	70GB	4 x A100 (80 GiB)	75GB	2 x Quadro RTX 6000	75GB

Table S6: Resources used for each TL setup on *Meltome-mixed* task.

PLM + Layer	Embeddings Extraction		Feature Extraction		LoRA		LoRA-		Adapters		Adapters-	
	GPUs	CPU RAM	GPUs	CPU RAM	GPUs	CPU RAM	GPUs	CPU RAM	GPUs	CPU RAM	GPUs	CPU RAM
ProteinBERT - 1st	1 x RTX 4090	12GB	1 x RTX 4090	10GB	2 x RTX 2080 Ti	12GB	2 x RTX 3090	12GB	2 x RTX 3090	8GB	2 x RTX 3090	8GB
25%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	2 x RTX 2080 Ti	12GB	2 x RTX 3090	12GB	2 x RTX 3090	8GB	2 x RTX 3090	8GB
50%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	3 x RTX 2080 Ti	12GB	3 x RTX 2080 Ti	12GB	3 x RTX 4090	8GB	3 x RTX 2080 Ti	8GB
75%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	3 x RTX 3090	12GB	3 x RTX 3090	12GB	3 x RTX 4090	8GB	3 x RTX 3090	8GB
All/Last	1 x RTX 4090	12GB	1 x RTX 4090	10GB	3 x RTX 3090	12GB	3 x RTX 3090	12GB	3 x Quadro RTX 6000	8GB	3 x RTX 3090	8GB
ProGen2-small - 1st	1 x RTX 4090	12GB	1 x RTX 4090	10GB	2 x RTX 2080 Ti	12GB	2 x RTX 2080 Ti	12GB	2 x RTX 3090	12GB	2 x RTX 2080 Ti	12GB
25%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	2 x RTX 2080 Ti	12GB	2 x RTX 2080 Ti	12GB	2 x RTX 3090	12GB	2 x RTX 2080 Ti	12GB
50%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	3 x RTX 2080 Ti	12GB	3 x RTX 2080 Ti	12GB	3 x TITAN RTX	12GB	3 x RTX 2080 Ti	12GB
75%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	3 x RTX 3090	12GB	3 x RTX 3090	12GB	3 x RTX 4090	12GB	3 x RTX 3090	12GB
All/Last	1 x RTX 4090	12GB	1 x RTX 4090	10GB	3 x RTX 3090	12GB	3 x RTX 3090	12GB	3 x RTX 4090	12GB	3 x RTX 3090	12GB
ESM2-650M - 1st	1 x RTX 4090	12GB	1 x RTX 4090	14GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB	2 x RTX 4090	18GB	1 x RTX 4090	18GB
25%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB	2 x RTX 4090	18GB	1 x RTX 4090	18GB
50%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x RTX 4090	12GB	2 x RTX 4090	12GB	3 x RTX 4090	18GB	2 x RTX 4090	18GB
75%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x RTX 4090	12GB	2 x RTX 4090	12GB	4 x RTX 4090	18GB	2 x RTX 4090	18GB
All/Last	1 x RTX 4090	12GB	1 x RTX 4090	14GB	4 x Quadro RTX 6000	12GB	2 x TITAN RTX	12GB	4 x Quadro RTX 6000	18GB	2 x TITAN RTX	18GB
ProGen2-medium - 1st	1 x RTX 4090	12GB	1 x RTX 4090	14GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB	2 x RTX 4090	18GB	1 x RTX 4090	18GB
25%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB	2 x RTX 4090	18GB	1 x RTX 4090	18GB
50%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x RTX 4090	12GB	2 x RTX 4090	12GB	3 x RTX 4090	18GB	2 x RTX 4090	18GB
75%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x RTX 4090	12GB	2 x RTX 4090	12GB	4 x RTX 4090	18GB	2 x RTX 4090	18GB
All/Last	1 x RTX 4090	12GB	1 x RTX 4090	14GB	4 x Quadro RTX 6000	12GB	2 x TITAN RTX	12GB	4 x Quadro RTX 6000	18GB	2 x TITAN RTX	18GB
ESM2-3B - 1st	1 x RTX 4090	30GB	1 x RTX 4090	14GB	2 x TITAN RTX	40GB	1 x RTX 4090	40GB	2 x TITAN RTX	18GB	1 x RTX 4090	18GB
25%	1 x RTX 4090	30GB	1 x RTX 4090	14GB	2 x TITAN RTX	40GB	2 x TITAN RTX	40GB	2 x TITAN RTX	18GB	2 x TITAN RTX	18GB
50%	1 x RTX 4090	30GB	1 x RTX 4090	14GB	4 x TITAN RTX	40GB	4 x TITAN RTX	40GB	4 x TITAN RTX	18GB	4 x TITAN RTX	18GB
75%	1 x RTX 4090	30GB	1 x RTX 4090	14GB	3 x A100 (40 GiB)	40GB	4 x TITAN RTX	40GB	3 x A100 (40 GiB)	18GB	4 x TITAN RTX	18GB
All/Last	1 x RTX 4090	30GB	1 x RTX 4090	14GB	4 x A100 (80 GiB)	40GB	4 x Quadro RTX 6000	40GB	3 x A100 (40 GiB)	18GB	4 x Quadro RTX 6000	18GB
ProGen2-xlarge - 1st	1 x RTX 4090	60GB	1 x RTX 4090	14GB	1 x RTX 4090	40GB	1 x RTX 4090	40GB	2 x TITAN RTX	40GB	1 x RTX 4090	40GB
25%	1 x RTX 4090	60GB	1 x RTX 4090	14GB	2 x TITAN RTX	40GB	2 x TITAN RTX	40GB	2 x TITAN RTX	40GB	2 x TITAN RTX	40GB
50%	1 x RTX 4090	60GB	1 x RTX 4090	14GB	4 x TITAN RTX	40GB	4 x TITAN RTX	40GB	4 x TITAN RTX	40GB	4 x TITAN RTX	40GB
75%	1 x RTX 4090	60GB	1 x RTX 4090	14GB	4 x A100 (40 GiB)	40GB	4 x TITAN RTX	40GB	3 x A100 (40 GiB)	40GB	4 x TITAN RTX	40GB
All/Last	1 x A100 (80 GiB)	60GB	1 x RTX 4090	14GB	4 x A100 (80 GiB)	40GB	4 x Quadro RTX 6000	40GB	4 x A100 (80 GiB)	40GB	4 x Quadro RTX 6000	40GB
ESM2-15B - 1st	1 x RTX 4090	80GB	1 x RTX 4090	14GB	2 x TITAN RTX	70GB	1 x RTX 4090	70GB	2 x TITAN RTX	75GB	1 x RTX 4090	75GB
25%	1 x RTX 4090	80GB	1 x RTX 4090	14GB	2 x TITAN RTX	70GB	2 x TITAN RTX	70GB	3 x TITAN RTX	75GB	4 x TITAN RTX	75GB
50%	1 x RTX 4090	80GB	1 x RTX 4090	14GB	4 x A100 (40 GiB)	70GB	4 x TITAN RTX	70GB	4 x A100 (40 GiB)	75GB	4 x TITAN RTX	75GB
75%	1 x A100 (80 GiB)	80GB	1 x RTX 4090	14GB	4 x A100 (80 GiB)	70GB	4 x TITAN RTX	70GB	4 x A100 (80 GiB)	75GB	4 x TITAN RTX	75GB
All/Last	1 x A100 (80 GiB)	80GB	1 x RTX 4090	14GB	4 x A100 (80 GiB)	70GB	4 x TITAN RTX	70GB	4 x A100 (80 GiB)	75GB	4 x A100 (80 GiB)	75GB

Table S7: Resources used for each TL setup on *Trastuzumab-one vs rest* task.

PLM + Layer	Embeddings Extraction		Feature Extraction		LoRA		LoRA-		Adapters		Adapters-	
	GPUs	CPU RAM	GPUs	CPU RAM	GPUs	CPU RAM	GPUs	CPU RAM	GPUs	CPU RAM	GPUs	CPU RAM
ProteinBERT - 1st	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB
25%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB
50%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB
75%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB
All/Last	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB	1 x RTX 4090	8GB	1 x RTX 4090	8GB
ProGen2-small - 1st	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB
25%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB
50%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB
75%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB
All/Last	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB
ESM2-650M - 1st	1 x RTX 4090	12GB	1 x RTX 4090	14GB	1 x RTX 3090	12GB	1 x RTX 3090	12GB	1 x RTX 3090	18GB	1 x RTX 3090	18GB
25%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	1 x RTX 3090	12GB	1 x RTX 3090	12GB	1 x RTX 3090	18GB	1 x RTX 3090	18GB
50%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x RTX 4090	12GB	2 x RTX 4090	12GB	2 x RTX 4090	18GB	2 x RTX 4090	18GB
75%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x Quadro RTX 6000	12GB	2 x Quadro RTX 6000	12GB	2 x Quadro RTX 6000	18GB	2 x Quadro RTX 6000	18GB
All/Last	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x Quadro RTX 6000	12GB	2 x Quadro RTX 6000	12GB	2 x Quadro RTX 6000	18GB	2 x Quadro RTX 6000	18GB
ProGen2-medium - 1st	1 x RTX 4090	12GB	1 x RTX 4090	14GB	1 x RTX 3090	12GB	1 x RTX 4090	12GB	1 x RTX 3090	18GB	1 x RTX 4090	18GB
25%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	1 x RTX 3090	12GB	1 x RTX 3090	12GB	1 x RTX 3090	18GB	1 x RTX 3090	18GB
50%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x RTX 4090	12GB	2 x RTX 4090	12GB	2 x RTX 4090	18GB	2 x RTX 4090	18GB
75%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x Quadro RTX 6000	12GB	2 x Quadro RTX 6000	12GB	2 x Quadro RTX 6000	18GB	2 x Quadro RTX 6000	18GB
All/Last	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x Quadro RTX 6000	12GB	2 x Quadro RTX 6000	12GB	2 x Quadro RTX 6000	18GB	2 x Quadro RTX 6000	18GB
ESM2-3B - 1st	1 x RTX 4090	30GB	1 x RTX 4090	14GB	1 x RTX 3090	40GB	1 x RTX 3090	40GB	1 x RTX 3090	18GB	1 x RTX 3090	18GB
25%	1 x RTX 4090	30GB	1 x RTX 4090	14GB	1 x RTX 4090	40GB	1 x RTX 4090	40GB	1 x RTX 4090	18GB	1 x RTX 4090	18GB
50%	1 x RTX 4090	30GB	1 x RTX 4090	14GB	1 x Quadro RTX 6000	40GB	1 x Quadro RTX 6000	40GB	1 x Quadro RTX 6000	18GB	1 x Quadro RTX 6000	18GB
75%	1 x RTX 4090	30GB	1 x RTX 4090	14GB	1 x Quadro RTX 6000	40GB	1 x Quadro RTX 6000	40GB	1 x Quadro RTX 6000	18GB	1 x Quadro RTX 6000	18GB
All/Last	1 x RTX 4090	30GB	1 x RTX 4090	14GB	1 x A100 (40 GiB)	40GB	1 x Quadro RTX 6000	40GB	1 x A100 (40 GiB)	18GB	1 x Quadro RTX 6000	18GB
ProGen2-xlarge - 1st	1 x RTX 4090	60GB	1 x RTX 4090	14GB	1 x RTX 2080 Ti	40GB	1 x RTX 2080 Ti	40GB	1 x RTX 2080 Ti	40GB	1 x RTX 2080 Ti	40GB
25%	1 x RTX 4090	60GB	1 x RTX 4090	14GB	1 x RTX 3090	40GB	1 x RTX 3090	40GB	1 x RTX 3090	40GB	1 x RTX 3090	40GB
50%	1 x RTX 4090	60GB	1 x RTX 4090	14GB	2 x RTX 3090	40GB	2 x RTX 3090	40GB	2 x RTX 3090	40GB	1 x RTX 3090	40GB
75%	1 x RTX 4090	60GB	1 x RTX 4090	14GB	2 x A100 (40 GiB)	40GB	2 x RTX 3090	40GB	2 x A100 (40 GiB)	40GB	1 x RTX 3090	40GB
All/Last	1 x A100 (80 GiB)	60GB	1 x RTX 4090	14GB	2 x A100 (80 GiB)	40GB	2 x Quadro RTX 6000	40GB	2 x A100 (80 GiB)	40GB	1 x Quadro RTX 6000	40GB
ESM2-15B - 1st	1 x RTX 4090	80GB	1 x RTX 4090	14GB	1 x RTX 2080 Ti	70GB	1 x RTX 2080 Ti	70GB	1 x RTX 2080 Ti	75GB	1 x RTX 2080 Ti	75GB
25%	1 x RTX 4090	80GB	1 x RTX 4090	14GB	2 x RTX 3090	70GB	2 x RTX 3090	70GB	2 x RTX 3090	75GB	2 x RTX 3090	75GB
50%	1 x RTX 4090	80GB	1 x RTX 4090	14GB	2 x A100 (40 GiB)	70GB	2 x RTX 3090	70GB	2 x A100 (40 GiB)	75GB	2 x RTX 3090	75GB
75%	1 x A100 (80 GiB)	80GB	1 x RTX 4090	14GB	3 x A100 (80 GiB)	70GB	2 x Quadro RTX 6000	70GB	3 x A100 (80 GiB)	75GB	2 x RTX 4090	75GB
All/Last	1 x A100 (80 GiB)	80GB	1 x RTX 4090	14GB	3 x A100 (80 GiB)	70GB	3 x Quadro RTX 6000	70GB	3 x A100 (80 GiB)	75GB	2 x A100 (80 GiB)	75GB

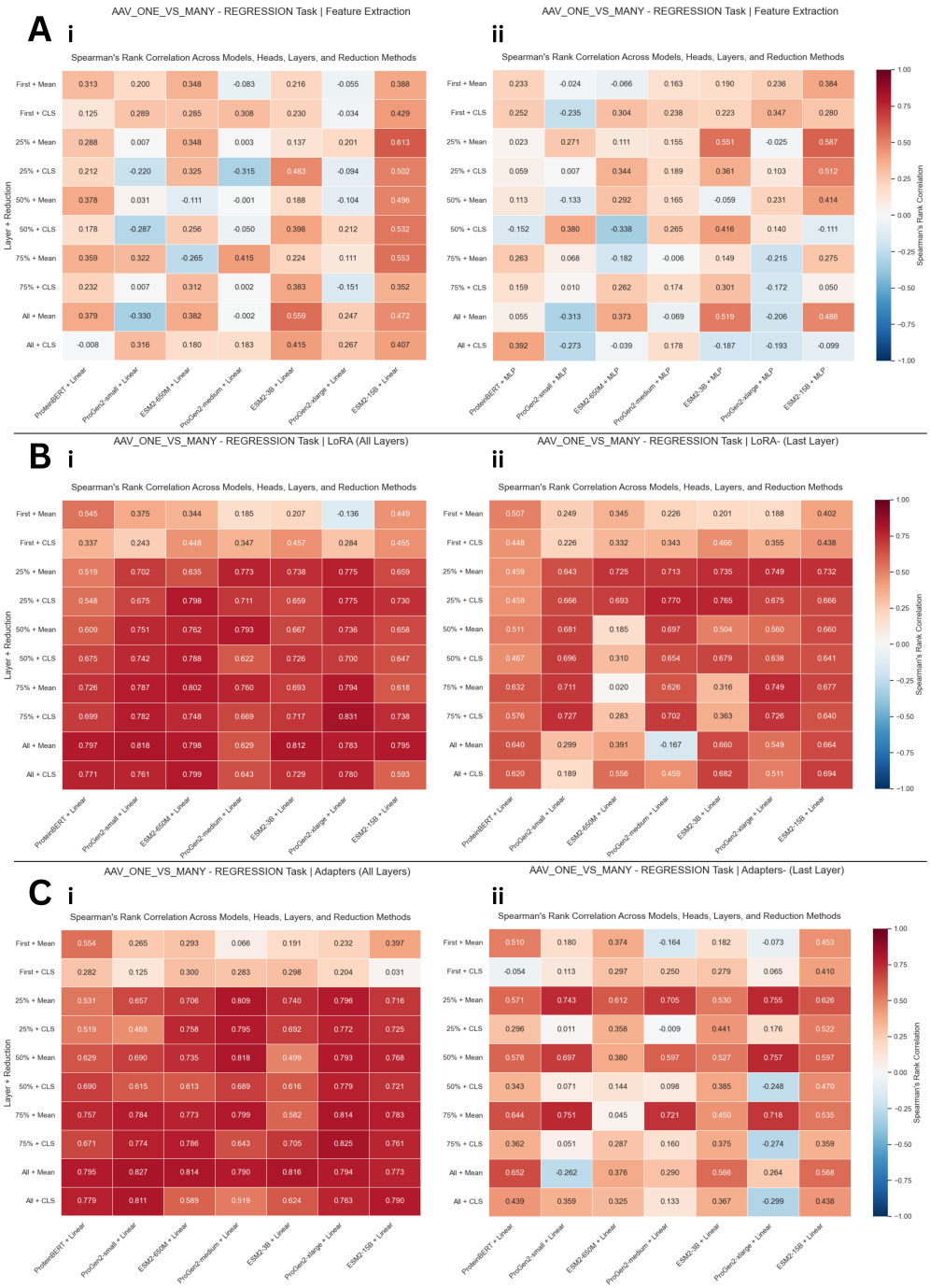

Figure S2: Detailed results per TL method for *AAV-one vs rest* task. Spearman's rank correlation is used as a performance metric for each setup, with x-axis showing the PLM and the head used and the y-axis representing the layers used and the pooling method employed; mean stands for mean pooling and CLS for pooling the classification token for BERT-based PLMs (ESM2, ProteinBERT) and the EOS token for GPT-based PLMs (ProGen2). (A) Feature extraction detailed results using (i) a linear downstream head and (ii) a MLP with one hidden layer as a downstream head. (B) LoRA detailed results when (i) applying LoRA to all layers of PLMs and (ii) applying LoRA to the last layer of PLMs. (C) Adapters detailed results when (i) applying adapters to all layers of PLMs and (ii) applying Adapters to the last layer of PLMs. TL: Transfer Learning; PLM: Protein Language Model; LoRA: Low Rank Adaptation

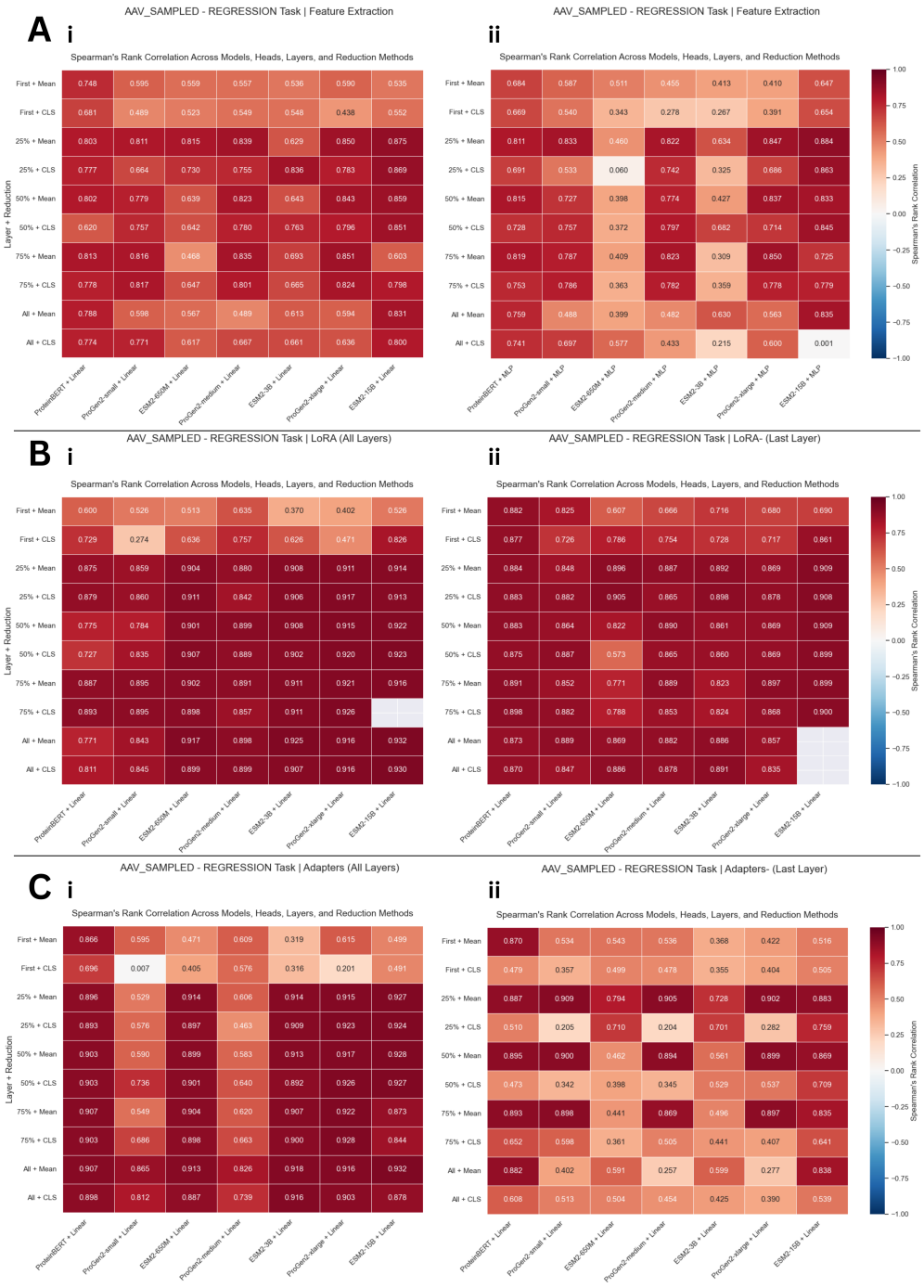

Figure S3: Detailed results per TL method for *AAV-sampled* task. Spearman's rank correlation is used as a performance metric for each setup, with x-axis showing the PLM and the head used and the y-axis representing the layers used and the pooling method employed; mean stands for mean pooling and CLS for pooling the classification token for BERT-based PLMs (ESM2, ProteinBERT) and the EOS token for GPT-based PLMs (ProGen2). (A) Feature extraction detailed results using (i) a linear downstream head and (ii) a MLP with one hidden layer as a downstream head. (B) LoRA detailed results when (i) applying LoRA to all layers of PLMs and (ii) applying LoRA to the last layer of PLMs. (C) Adapters detailed results when (i) applying adapters to all layers of PLMs and (ii) applying adapters to the last layer of PLMs. Empty cells represent work in progress, due to the computational burden these setups bear. TL: Transfer Learning; PLM: Protein Language Model; LoRA: Low Rank Adaptation

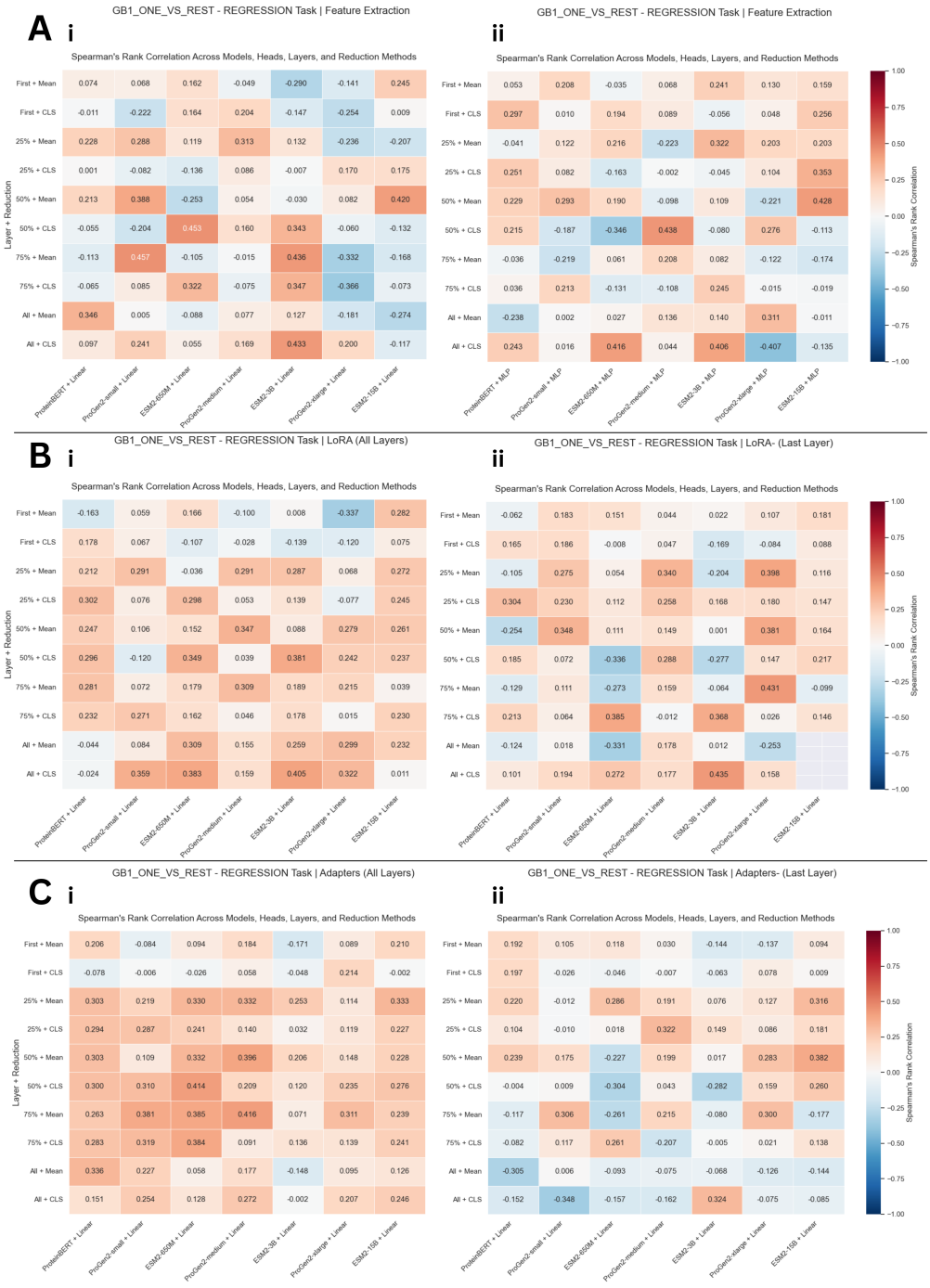

Figure S4: Detailed results per TL method for *GB1-one vs rest* task. Spearman's rank correlation is used as a performance metric for each setup, with x-axis showing the PLM and the head used and the y-axis representing the layers used and the pooling method employed; mean stands for mean pooling and CLS for pooling the classification token for BERT-based PLMs (ESM2, ProteinBERT) and the EOS token for GPT-based PLMs (ProGen2). (A) Feature extraction detailed results using (i) a linear downstream head and (ii) a MLP with one hidden layer as a downstream head. (B) LoRA detailed results when (i) applying LoRA to all layers of PLMs and (ii) applying LoRA to the last layer of PLMs. (C) Adapters detailed results when (i) applying adapters to all layers of PLMs and (ii) applying adapters to the last layer of PLMs. Empty cells represent work in progress, due to the computational burden these setups bear. TL: Transfer Learning; PLM: Protein Language Model; LoRA: Low Rank Adaptation

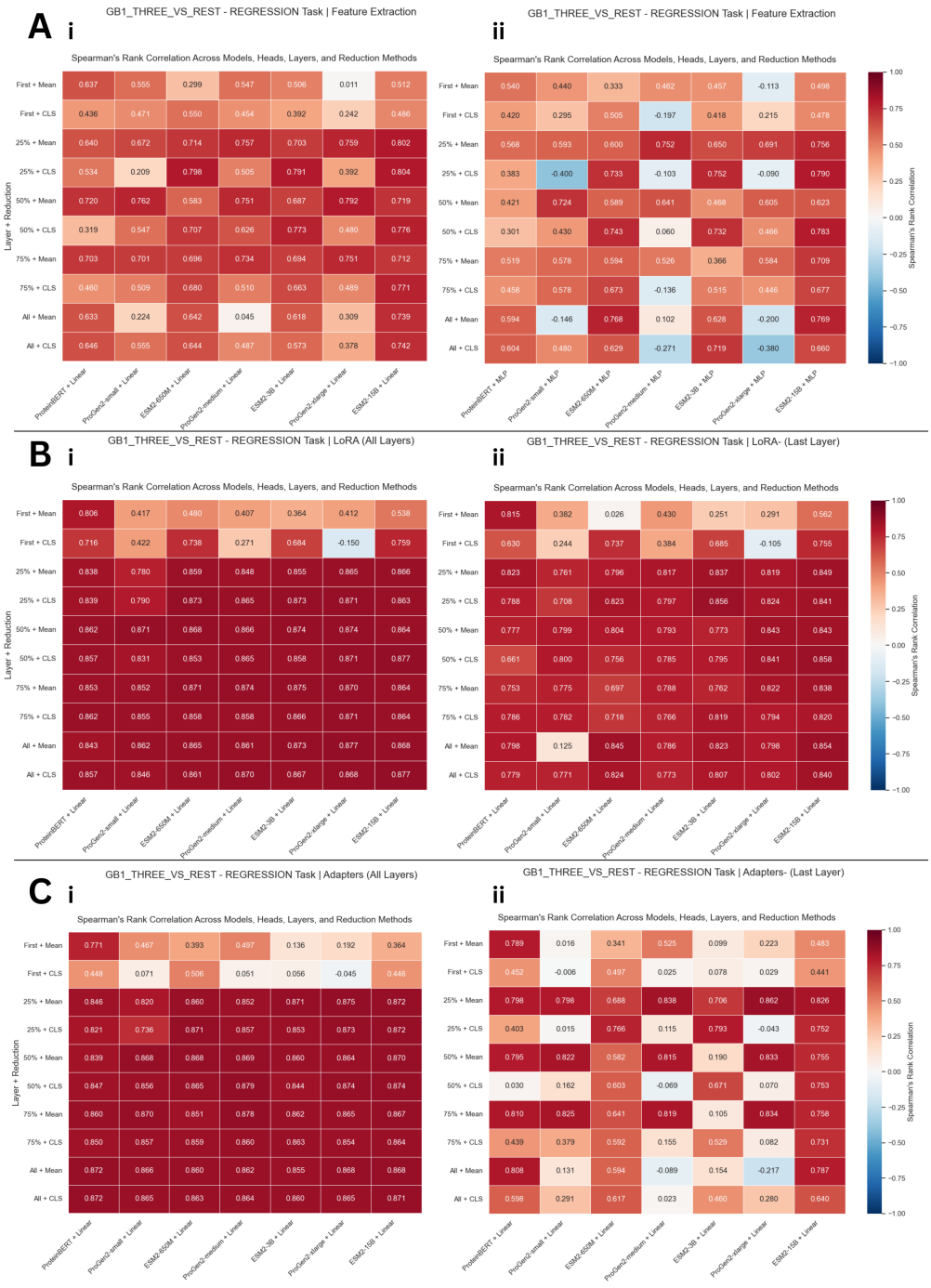

Figure S5: Detailed results per TL method for *GB1-three vs rest* task. Spearman's rank correlation is used as a performance metric for each setup, with x-axis showing the PLM and the head used and the y-axis representing the layers used and the pooling method employed; mean stands for mean pooling and CLS for pooling the classification token for BERT-based PLMs (ESM2, ProteinBERT) and the EOS token for GPT-based PLMs (ProGen2). (A) Feature extraction detailed results using (i) a linear downstream head and (ii) a MLP with one hidden layer as a downstream head. (B) LoRA detailed results when (i) applying LoRA to all layers of PLMs and (ii) applying LoRA to the last layer of PLMs. (C) Adapters detailed results when (i) applying adapters to all layers of PLMs and (ii) applying adapters to the last layer of PLMs. Empty cells represent work in progress, due to the computational burden these setups bear. TL: Transfer Learning; PLM: Protein Language Model; LoRA: Low Rank Adaptation

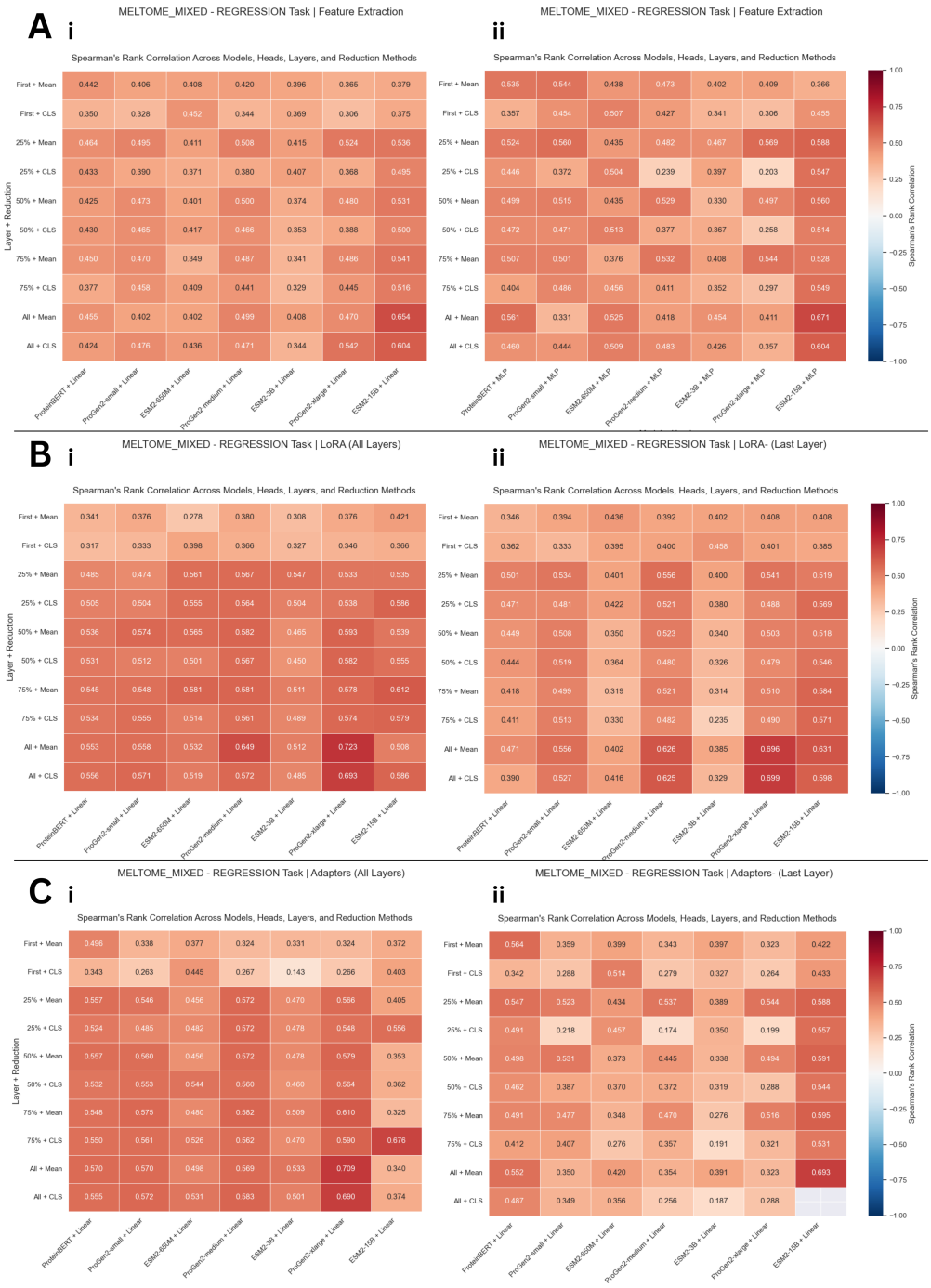

Figure S6: Detailed results per TL method for *Meltome-mixed* task. Spearman's rank correlation is used as a performance metric for each setup, with x-axis showing the PLM and the head used and the y-axis representing the layers used and the pooling method employed; mean stands for mean pooling and CLS for pooling the classification token for BERT-based PLMs (ESM2, ProteinBERT) and the EOS token for GPT-based PLMs (ProGen2). (A) Feature extraction detailed results using (i) a linear downstream head and (ii) a MLP with one hidden layer as a downstream head. (B) LoRA detailed results when (i) applying LoRA to all layers of PLMs and (ii) applying LoRA to the last layer of PLMs. (C) Adapters detailed results when (i) applying adapters to all layers of PLMs and (ii) applying adapters to the last layer of PLMs. Empty cells represent work in progress, due to the computational burden these setups bear. TL: Transfer Learning; PLM: Protein Language Model; LoRA: Low Rank Adaptation

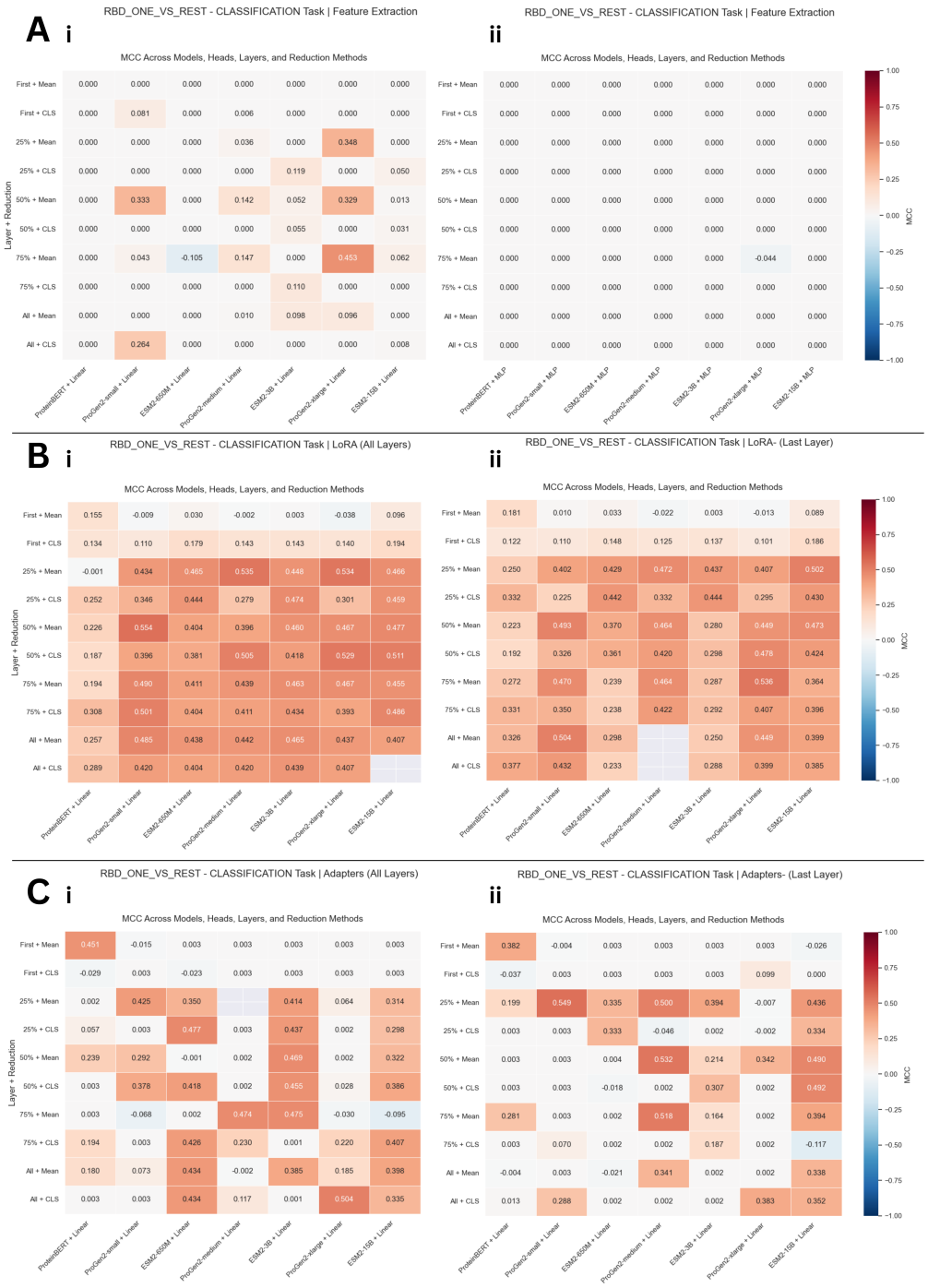

Figure S7: Detailed results per TL method for *RBD-one vs rest* task. MCC is used as a performance metric for each setup, with x-axis showing the PLM and the head used and the y-axis representing the layers used and the pooling method employed; mean stands for mean pooling and CLS for pooling the classification token for BERT-based PLMs (ESM2, ProteinBERT) and the EOS token for GPT-based PLMs (ProGen2). (A) Feature extraction detailed results using (i) a linear downstream head and (ii) a MLP with one hidden layer as a downstream head. (B) LoRA detailed results when (i) applying LoRA to all layers of PLMs and (ii) applying LoRA to the last layer of PLMs. (C) Adapters detailed results when (i) applying adapters to all layers of PLMs and (ii) applying adapters to the last layer of PLMs. Empty cells represent work in progress, due to the computational burden these setups bear. TL: Transfer Learning; PLM: Protein Language Model; LoRA: Low Rank Adaptation; MCC: Matthew's correlation coefficient

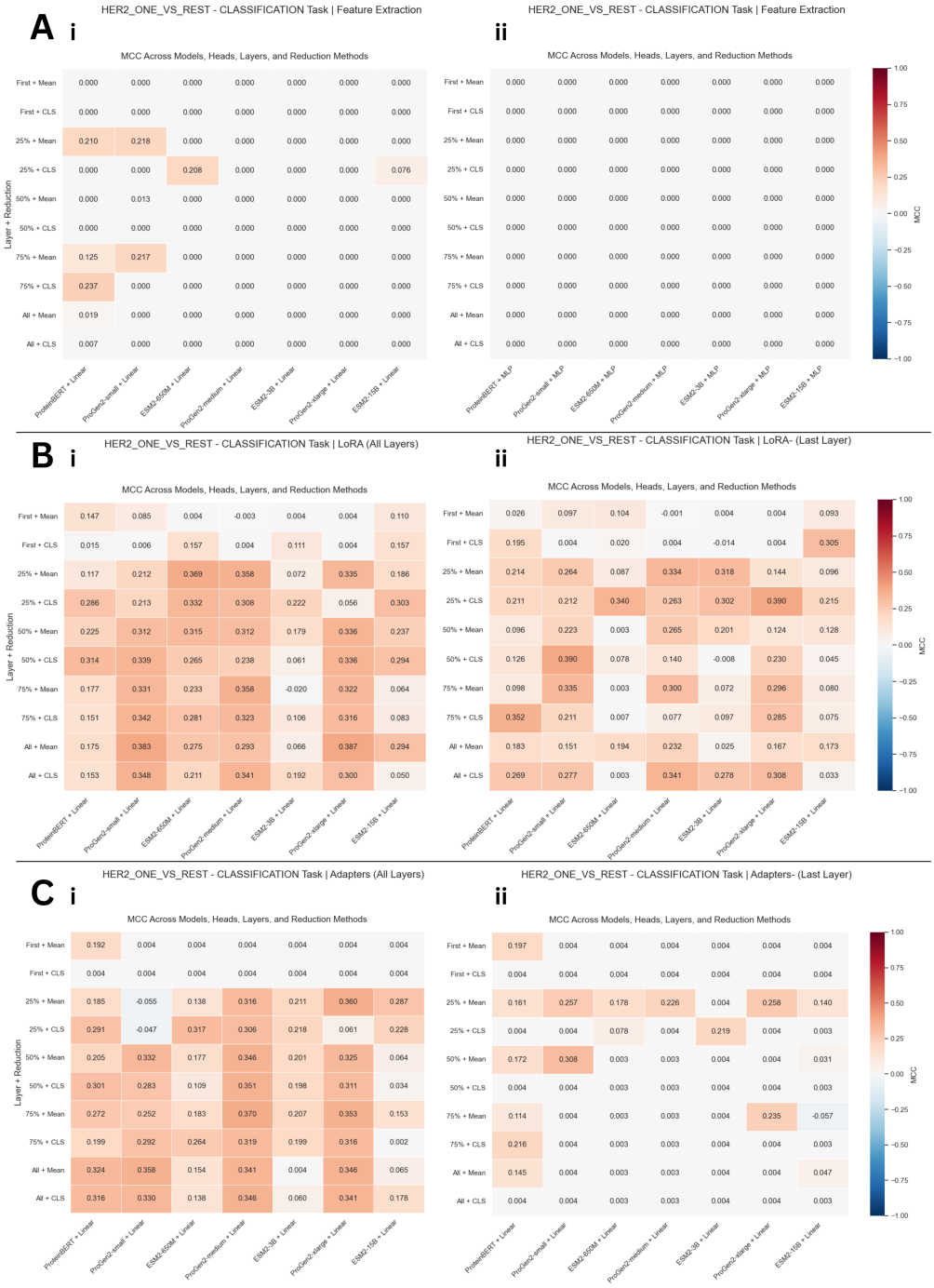

Figure S8: Detailed results per TL method for *Trastuzumab (HER2)-one vs rest* task. MCC is used as a performance metric for each setup, with x-axis showing the PLM and the head used and the y-axis representing the layers used and the pooling method employed; mean stands for mean pooling and CLS for pooling the classification token for BERT-based PLMs (ESM2, ProteinBERT) and the EOS token for GPT-based PLMs (ProGen2). (A) Feature extraction detailed results using (i) a linear downstream head and (ii) a MLP with one hidden layer as a downstream head. (B) LoRA detailed results when (i) applying LoRA to all layers of PLMs and (ii) applying LoRA to the last layer of PLMs. (C) Adapters detailed results when (i) applying adapters to all layers of PLMs and (ii) applying adapters to the last layer of PLMs. TL: Transfer Learning; PLM: Protein Language Model; LoRA: Low Rank Adaptation; MCC: Matthew's correlation coefficient

Table S8: Resources used for each TL setup on *RBD-one vs rest* task.

PLM + Layer	Embeddings Extraction		Feature Extraction		LoRA		LoRA-		Adapters		Adapters-	
	GPUs	CPU RAM	GPUs	CPU RAM	GPUs	CPU RAM	GPUs	CPU RAM	GPUs	CPU RAM	GPUs	CPU RAM
ProteinBERT - 1st	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB
25%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB
50%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 2080 Ti	8GB	1 x RTX 2080 Ti	8GB
75%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 4090	12GB	1 x RTX 2080 Ti	12GB	1 x RTX 4090	8GB	1 x RTX 2080 Ti	8GB
All/Last	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB	1 x RTX 4090	8GB	1 x RTX 4090	8GB
ProGen2-small - 1st	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB
25%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB
50%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB
75%	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB
All/Last	1 x RTX 4090	12GB	1 x RTX 4090	10GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB	1 x RTX 4090	12GB
ESM2-650M - 1st	1 x RTX 4090	12GB	1 x RTX 4090	14GB	1 x RTX 3090	12GB	1 x RTX 3090	12GB	1 x RTX 3090	18GB	1 x RTX 3090	18GB
25%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	1 x RTX 3090	12GB	1 x RTX 3090	12GB	1 x RTX 3090	18GB	1 x RTX 3090	18GB
50%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x RTX 4090	12GB	2 x RTX 4090	12GB	2 x RTX 4090	18GB	2 x RTX 4090	18GB
75%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x Quadro RTX 6000	12GB	2 x Quadro RTX 6000	12GB	2 x Quadro RTX 6000	18GB	2 x Quadro RTX 6000	18GB
All/Last	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x Quadro RTX 6000	12GB	2 x Quadro RTX 6000	12GB	2 x Quadro RTX 6000	18GB	2 x Quadro RTX 6000	18GB
ProGen2-medium - 1st	1 x RTX 4090	12GB	1 x RTX 4090	14GB	1 x RTX 3090	12GB	1 x RTX 3090	12GB	1 x RTX 3090	18GB	1 x RTX 3090	18GB
25%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	1 x RTX 3090	12GB	1 x RTX 3090	12GB	1 x RTX 3090	18GB	1 x RTX 3090	18GB
50%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x RTX 4090	12GB	2 x RTX 4090	12GB	2 x RTX 4090	18GB	2 x RTX 4090	18GB
75%	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x Quadro RTX 6000	12GB	2 x RTX 4090	12GB	2 x Quadro RTX 6000	18GB	2 x RTX 4090	18GB
All/Last	1 x RTX 4090	12GB	1 x RTX 4090	14GB	2 x Quadro RTX 6000	12GB	2 x Quadro RTX 6000	12GB	2 x Quadro RTX 6000	18GB	2 x Quadro RTX 6000	18GB
ESM2-3B - 1st	1 x RTX 4090	30GB	1 x RTX 4090	14GB	1 x RTX 3090	40GB	1 x RTX 3090	40GB	1 x RTX 3090	18GB	1 x RTX 3090	18GB
25%	1 x RTX 4090	30GB	1 x RTX 4090	14GB	2 x RTX 4090	40GB	2 x RTX 4090	40GB	2 x RTX 4090	18GB	2 x RTX 4090	18GB
50%	1 x RTX 4090	30GB	1 x RTX 4090	14GB	2 x Quadro RTX 6000	40GB	2 x Quadro RTX 6000	40GB	2 x Quadro RTX 6000	18GB	2 x Quadro RTX 6000	18GB
75%	1 x RTX 4090	30GB	1 x RTX 4090	14GB	3 x Quadro RTX 6000	40GB	3 x Quadro RTX 6000	40GB	3 x Quadro RTX 6000	18GB	2 x Quadro RTX 6000	18GB
All/Last	1 x RTX 4090	30GB	1 x RTX 4090	14GB	3 x A100 (40 GiB)	40GB	3 x Quadro RTX 6000	40GB	3 x A100 (40 GiB)	18GB	2 x Quadro RTX 6000	18GB
ProGen2-xlarge - 1st	1 x RTX 4090	60GB	1 x RTX 4090	14GB	1 x RTX 2080 Ti	40GB	1 x RTX 2080 Ti	40GB	1 x RTX 2080 Ti	40GB	1 x RTX 2080 Ti	40GB
25%	1 x RTX 4090	60GB	1 x RTX 4090	14GB	2 x RTX 3090	40GB	2 x RTX 3090	40GB	2 x RTX 3090	40GB	2 x RTX 3090	40GB
50%	1 x RTX 4090	60GB	1 x RTX 4090	14GB	2 x RTX 3090	40GB	2 x RTX 3090	40GB	2 x RTX 3090	40GB	2 x RTX 3090	40GB
75%	1 x RTX 4090	60GB	1 x RTX 4090	14GB	2 x A100 (40 GiB)	40GB	2 x RTX 3090	40GB	2 x Quadro RTX 6000	40GB	2 x RTX 3090	40GB
All/Last	1 x A100 (80 GiB)	60GB	1 x RTX 4090	14GB	3 x A100 (80 GiB)	40GB	3 x Quadro RTX 6000	40GB	3 x Quadro RTX 6000	40GB	2 x Quadro RTX 6000	40GB
ESM2-15B - 1st	1 x RTX 4090	80GB	1 x RTX 4090	14GB	1 x RTX 2080 Ti	70GB	1 x RTX 2080 Ti	70GB	1 x RTX 2080 Ti	75GB	1 x RTX 2080 Ti	75GB
25%	1 x RTX 4090	80GB	1 x RTX 4090	14GB	2 x RTX 3090	70GB	2 x RTX 3090	70GB	3 x RTX 3090	75GB	1 x RTX 3090	75GB
50%	1 x RTX 4090	80GB	1 x RTX 4090	14GB	2 x A100 (40 GiB)	70GB	2 x Quadro RTX 6000	70GB	3 x Quadro RTX 6000	75GB	2 x Quadro RTX 6000	75GB
75%	1 x A100 (80 GiB)	80GB	1 x RTX 4090	14GB	3 x A100 (80 GiB)	70GB	2 x Quadro RTX 6000	70GB	3 x A100 (80 GiB)	75GB	2 x Quadro RTX 6000	75GB
All/Last	1 x A100 (80 GiB)	80GB	1 x RTX 4090	14GB	4 x A100 (80 GiB)	70GB	2 x A100 (40 GiB)	70GB	3 x A100 (80 GiB)	75GB	2 x A100 (80 GiB)	75GB

Table S9: Parameters used for the DeepSpeed package.

Parameter	Value
Stage	3
Parameter Offload	Yes
Optimizer Offload	Yes
Offload Device	CPU
Sub Group Size	1e12
Overlap Comms.	Yes
Allgather Bucket Size	2e8
Reduce Bucket Size	2e8

Table S10: Hyperparameters used for all setups and for each task examined in the PLMFit study.

	GB1 one-vs-rest	GB1 - three vs rest	AAV - one vs rest	AAV - sampled	Meltome - mixed	RBD - one-vs-rest	HER2 - one-vs-rest
Feature extraction / One hot encoding							
Hyperparameters Tuned	Learning rate, weight decay, batch size, hidden dimension (MLP)						
Learning rate space	1e-2 - 1e-6						
Weight decay space	1e-1 - 1e-6						
Batch size space	8 - 128						
Hidden dimension space (MLP)	64 - 2048						
Total trials Linear/MLP	100/500						
Initial random points	20						
Loss function	MSE					BCE	
Optimizer	Adam						
Epochs	200						
Patience (early stopping)	30						
LoRA							
Modules Applied	Q, K, V						
LoRA rank	8						
LoRA alpha	16						
LoRA dropout	0.1						
Loss function	MSE					BCE	
Optimizer	Adam (DeepSpeedCPU)						
Batch size	4						
Epochs	200	30	30	3	10	150	150
Patience (early stopping)	100	5	-	1	5	-	-
Learning rate	1e-4						
Weight decay	1e-2						
LoRA-							
Modules Applied	Q, K, V						
LoRA rank	8						
LoRA alpha	16						
LoRA dropout	0.1						
Loss function	MSE					BCE	
Optimizer	Adam (DeepSpeedCPU)						
Batch size	4						
Epochs	200	30	30	10	15	150	150
Patience (early stopping)	100	5	-	5	5	-	-
Learning rate	1e-4						
Weight decay	1e-2						
Adapters							
Modules Applied	After FFN						
Bottleneck dimension	32						
Scaling	Learned						
Adapter dropout	0.1						
Loss function	MSE					BCE	
Optimizer	Adam (DeepSpeedCPU)						
Batch size	4						
Epochs	200	30	30	3	10	150	150
Patience (early stopping)	100	5	-	1	5	-	-
Learning rate	1e-4						
Weight decay	1e-4						
Adapters-							
Modules Applied	After FFN						
Bottleneck dimension	32						
Scaling	Learned						
Adapter dropout	0.1						
Loss function	MSE					BCE	
Optimizer	Adam (DeepSpeedCPU)						
Batch size	4						
Epochs	200	30	30	10	15	150	150
Patience (early stopping)	100	5	-	5	5	-	-
Learning rate	1e-4						
Weight decay	1e-4						

Table S11: Statistical summaries (quarter1, quarter3, median and max) of the box plots depicted in Figure 2B.

Task	Box plot stats	FE	FT
AAV - sampled	Q1	-5.62%	-0.39%
	Q3	-1.48%	7.45%
	Median	-3.2%	3.95%
	Max	2.32%	7.45%
AAV - one_vs_rest	Q1	-38.61%	8.38%
	Q3	8.45%	47.17%
	Median	-30.54%	38.03%
	Max	8.45%	47.17%
GB1 - three_vs_rest	Q1	-14.05%	-8.56%
	Q3	-4%	4.9%
	Median	-5.65%	2.89%
	Max	-4%	4.9%
GB1 - one_vs_rest	Q1	-6.41%	-34.65%
	Q3	37.67%	30.98%
	Median	31.46%	0.74%
	Max	37.67%	30.98%
Meltome - mixed	Q1	58.12%	31.47%
	Q3	71.27%	117.83%
	Median	68.71%	70.79%
	Max	102.22%	117.83%

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Both "Abstract" and "Introduction" sections are organized by paragraphs rationally ordered presenting in an easy to follow way the scope of the paper. Both sections end by referring the overall contribution of the results of this study to the scientific fields of machine learning and protein engineering. Observed results and future goals are clearly separated in the "Introduction" and "Discussion" sections.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: A "Limitations" section is provided to address the limitations of this paper.

   Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: The presented work is a benchmarking analysis that draws conclusion based on the observations of computational experiments. Theoretical results are not provided as part of this study.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides detailed experimental setups (alongside with the datasets and code) to reproduce all the experiments. Detailed tables with parameters of packages and architectures are provided in the supplementary material section to ensure reproducibility.

   Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes] ,

   Justification: All codes and datasets are provided in a github repository as stated in sections "Introduction" and "Acknowledgements and Discoluse of Funding"

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.

- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer:[Yes]

   Justification: Training and validation splits, along with extensive tables with all hyperparameters, modules, optimizer and hardware used for each experiment in this study are clearly explained and presented with tables in the "Supplementary Material" section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: To address the core question of when and how to apply fine-tuning, feature extraction, or neither, we present an ensemble of results as bar plots (Figure 2b). Error plots for statistical significance for every experiment conducted are not included in this study due to the scale and computational cost of evaluating over 2,900 experimental setups, some involving models with billions of parameters. Running each experiment multiple times to estimate variance would be computationally infeasible considering that some experiments (e.g. fine-tuning ESM2 with 15 billion parameters) could require 1 week to complete. Instead, detailed heatmaps for all experiments, including variations with different pooling methods, are provided in the extended data section of the supplementary materials, offering a comprehensive sanity check of the validity of our comparative analysis.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Detailed tables with the computational resources used in this study are provided in the "Supplementary Materials" section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Paper was written considering all the aspects of NeurIPS Code of Ethics. This study does not involve human subjects.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.

- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Publicly available datasets are used for this benchmarking study. All references and citations are provided via GitHub in in the "Introduction" and "Acknowledgments and Disclosure of Funding".

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: An open source platform has been developed as part of this study. Proper documentation and implementation details are provided at `https://github.com/LSSI-ETH/plmfit`.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification:[NA]

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification:[NA]

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.