# Optimizing protein language models with Sentence Transformers

**Istvan Redl** [*]
Peptone Ltd.
istvan@peptone.io

**Fabio Airoldi**
Peptone Ltd.
fabio@peptone.io

**Sandro Bottaro**
Peptone Ltd.
sandro@peptone.io

**Albert Chung**
Peptone Ltd.
albert@peptone.io

**Oliver Dutton**
Peptone Ltd.
oliver@peptone.io

**Carlo Fisicaro**
Peptone Ltd.
carlo@peptone.io

**Patrik Foerch**
Peptone Ltd.
patrik@peptone.io

**Louie Henderson**
Peptone Ltd.
louie@peptone.io

**Falk Hoffmann**
Peptone Ltd.
falk@peptone.io

**Michele Invernizzi**
Peptone Ltd.
michele@peptone.io

**Benjamin M J Owens**
Peptone Ltd.
ben@peptone.io

**Stefano Ruschetta**
Peptone Ltd.
stefano@peptone.io

**Kamil Tamiola**
Peptone Ltd.
kamil@peptone.io

## Abstract

Protein language models (pLMs) have appeared in a wide range of *in-silico* protein engineering tasks and have shown impressive results. However the ways they are applied remain mostly standardised. Here, we introduce a set of finetuning techniques based on *Sentence Transformers* (STs) integrated with a novel data augmentation procedure and show how it can offer new state-of-the-art performance. Despite having initially been developed in classic NLP space, STs hold a natural appeal in pLM related applications, largely due their use of sequence pairs and triplets in the process. We demonstrate this conceptual approach in two different settings that frequently occur in this domain; a *residue* and also a *sequence* level prediction task. Apart from showing how these tools can extract more and higher quality information from pLMs, we discuss the main differences between their applications in NLP and in the protein spaces. We conclude by discussing the related challenges and provide a comprehensive outlook on potential applications.

## 1 Introduction

Sequence based methods have long adopted techniques from classical natural language processing (NLP) e.g. [1] used an LSTM based approach to predict structural similarities between proteins, and [21] was one of the first papers that used transformers on various downstream tasks, including secondary structure, contact map, fluorescence and stability predictions. In recent years a whole suite of transformer based models appeared [25, 22, 13, 7, 12, 10, 18, 19, 17], and have shown impressive performance in numerous in-silico protein engineering tasks. Fine tuning, e.g. [26, 30] or using frozen representations, e.g. [28], from these protein Language Models (pLMs) are standard ways to use them in various downstream tasks.

[*]Peptone Ltd, 370 Grays Inn Road, London WC1X 8BB, UK (https://peptone.io/); Correspondence to: Istvan Redl: istvan@peptone.io

Here we show how Sentence Transformers [24], sometimes also called Sentence-BERT (S-BERT), a collection of powerful techniques that have been widely used in information retrieval and semantic searches in the classical NLP space, can be applied to pLMs as well. STs push the representations of two semantically similar sequences closer together in the embedding space, and further apart for dissimilar sequences. Each sequence is embedded by the same encoder via a Siamese network, and the embeddings are compared with a contrastive learning loss such as cosine distance.

Our contributions:

- As to the best of our knowledge, these techniques have not been used in protein space thus far, we give an overview of SentenceTransformers that have been applied extensively in NLP and are behind some of the top embedding models, see [15] for an extensive list of models and references;
- We demonstrate through two examples how STs can be applied to in-silico protein engineering tasks and how they can achieve state-of-the-art performance. Apart from a more straightforward *sequence level* prediction task (solubility), we also show how this approach can be used in *residue level* prediction (disorder). Code is made available at `https://github.com/PeptoneLtd/contrastive-finetuning-plms`;
- We also discuss challenges, applications and areas in protein space, where these tools can potentially contribute. Given that in NLP these techniques have been in development since the end of 2010's, we are not aiming to cover all potential applications with experimental evidence, but we do hope to encourage our community to explore these tools.

## 2 Background

Given that our proposed approach is based on a technique that is - to the best of our knowledge - new in the pLM space, we provide a brief description of the main tools and concepts related to STs in Appendix A.1.

Supervised contrastive learning has shown marked improvements over more traditional supervised methods in classical NLP [16], [8] and computer vision [2] as well. The idea of contrastive learning has appeared in the protein space in the past, even related to pLMs, see [9]. However, the authors of [9] used static embeddings from a pLM as inputs to a feed-forward neural network (FNN), which was optimised on a triplet loss, similar to 2 in Appendix A.1. In other words, only the weights of the FNN were optimised during training and the backpropagation left the pLM weights unchanged. This amounts to an approach depicted on the left hand side of Figure 1, where only the prediction head was trained.

Recently some ideas from S-BERT [24] appeared in [32], where they used the 3 types ([`CLS`], `mean`, `max`) of pooled embeddings concatenated into one as part of an intriguing explainability study related to embeddings produced by pLMs. However, the nature of the work in [32] is very different to what we present here.

We mention another interesting work [27] in the direction of contrastive learning and pLM, where the authors designed a method called *contrastive coembedding* to embed proteins and drugs into a shared latent space. On a high level, this was a building block of a sequence based method that predicts drug-target interaction.

## 3 Overview of our Approach

In this section, we present our approach using STs in two different settings; *residue* and *sequence* level prediction tasks. While there are clear differences, which we explain in detail, some of the key building blocks are the same.

We draw inspiration from an approach in NLP called SETFIT [31] (**Se**ntence **T**ransformer **Fi**ne**t**uning). The key idea is to train a BiEncoder (BE) on pairs of sequences. In all cases we will instantiate the BE by using one of the *pre-trained* `ESM-2` models [12], which were trained with masked language modelling objective on UniRef sequence databases. In the training set, we use positive and negative pairs, i.e. where two sequences are either both from the same or different class. More details on how these classes are defined will be given below in the task specific sections. Training objective will

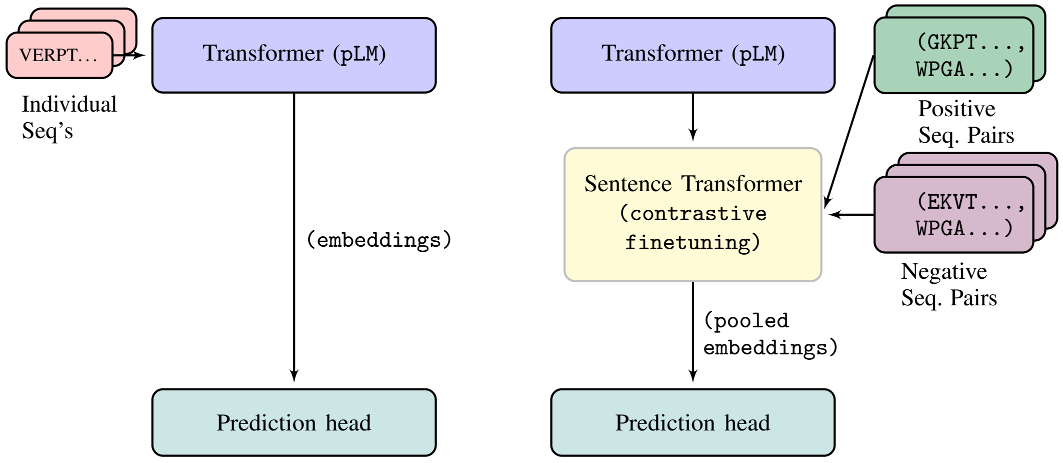

Figure 1: Schematic overview of finetuning pLMs using Sentence Transformers (on the **right**) compared to a more traditional approach (on the **left**), where either the frozen embeddings of a pre-trained model are used coupled with a prediction head or the pLM is finetuned 'end-to-end' using the prediction head, i.e. the weights of the pLMs are also updated.

mainly be cosine similarity loss,

$$\cos(u_a, u_b) = \frac{u_a \cdot u_b}{\|u_a\|\|u_b\|}, \tag{1}$$

where $u_a$ denotes the pooled (e.g. mean) embedding of sequence $a$, i.e. this is a vector of size $d_{model}$, in the numerator $\cdot$ is the dot product and $\|\cdot\|$ stands for the Euclidean norm. Cosine similarity is compared to the pair label, which is set to $1$ if the sequences of the pairs belong to the same class, or $0$ otherwise. During the training the pooled embeddings of sequences will be pushed closer or further apart, depending on whether they belong to the same class or not. In the second step we use the fine-tuned BiEncoder to extract the embeddings of the sequences in the training set, which will be the input of the prediction head. A schematic view of our approach together with a more traditional one is depicted in Figure 1.

The crucial difference between the original SETFIT and our setting is that in the former approach, the fine-tuning phase uses a ST that has already been trained on a large number of example pairs, e.g. `sentence-transformers/all-roberta-large-v1` was trained on 1B pairs, for details and resources see `https://huggingface.co/sentence-transformers/all-roberta-large-v1`. This allows for the original SETFIT approach to perform actual few shot learning and achieve great results. In our case, we will have to use more data from the training set during the fine-tuning step, because there is no pLM based pre-trained ST available that was trained on large number of protein pairs.

### 3.1 Residue level task

Since STs output pooled - typically `mean` - embeddings, the approach outlined above does not lend itself naturally to a setting where amino acid or token level prediction is required. However we can still take advantage of STs by simply fine-tuning a small pLM on sequence level labels and concatenate this sequence level embedding with each residue level representation. In other words, for a given sequence we augment each amino acid level representation extracted from a larger pLM (e.g. `esm2_t33_650M_UR50D`) by attaching the *same* sequence level embedding, which is the output of a ST finetuned on the same training dataset. Figure 2 in Appendix A.3 shows an example how a finetuned ST alters the embeddings.

### 3.2 Sequence level task

Using a ST when a sequence level quantity has to be predicted is fairly straightforward and this is what we already outlined in the beginning of this section. We discuss the details on the choices regarding the training of the BE and also the prediction head in Appendix A.3 below.

Table 1: ST-avg benchmark on CheZoD

| | Regression | | | Classification | |
|---|---|---|---|---|---|
| Predictor | $r^2$ | MAE | $\texttt{corr}_{\texttt{Spearm.}}$ | AU-PRC | APS |
| ST-avg | **.435**±0.012 | **2.982**±0.031 | **.734**±0.003 | **.920**±0.003 | **.960**±0.003 |
| SETH* | – | – | .72±0.01 | .91±0.01 | – |
| ST-CV | .388±0.024 | 3.090±0.075 | .714±0.007 | .914±0.002 | .958±0.002 |
| ADOPT-esm2-CV | .309±0.005 | 3.273±0.013 | .700±0.001 | .907±0.001 | .954±0.000 |

## 4 Evaluation

Below, we present the experimental results. Details on the datasets, model architectures and training settings are given in Appendix A.2 and A.3.

### 4.1 Results on disorder predictions

**CheZoD benchmark with cross validation** To get a better understanding of the behaviour of the proposed approach we carried out a 10-fold cross validation (CV) on the CheZoD training set, where in each fold we ran 200 random seeds, and picked the model that had the highest correlation between predicted and actual z-scores on the left out fold. We tested the 10 top ranked models on the CheZoD test set, this is referred to as ST-CV in Table 1. As a baseline, we ran a predictor whose prediction head was set the same as that of ST-CV, but only the residue level representations were used from esm2_t33_650M_UR50D. Given that this simple approach is very similar to ADOPT [23], we referred to it as ADOPT-esm2-CV. Results on SETH were taken as reported in [11], hence the asterisk * mark and the 2 decimal display, see Table 1.

We highlight some key observations from these benchmarks. The ST based approach typically generalises better, sometimes significantly, than the other approaches which rely on frozen embeddings from pre-trained pLMs, which can include more advanced prediction heads, e.g. SETH that utilises a two layer CNN. This better generalisation can be seen through different lenses. In comparison of ST -CV with ADOPT-esm2-CV, the former has significantly higher $r^2$ (0.388 vs. 0.309), and also higher average correlation (0.714 vs. 0.700). To make a fair assessment between ST -avg and SETH, we can disregard the clear gap observed in the CheZoD benchmark and focus only on their performances on CAID, where ST -avg is better in terms of AU-PRC (0.935 vs. 0.930) and f1 (0.832 vs. 0.830), and almost on par in terms of APS (0.892 vs. 0.893), see Table 4 in Appendix A.4. The difference is more pronounced when compared to ST -top. It is important to add that these models can typically be found "*in the tail*", i.e. among the top 5 or even higher quantiles of finetuned STs, and identifying them comes at an increased computational cost. We will further comment in the Discussion section how this can potentially be reduced.

### 4.2 Results on solubility prediction

The improvement in AU-PRC (area under the precision-recall curve) between ST-esm2_t30 and NetSolP-esm2_t33 on the PSI dataset may seem minor, 0.768 vs. 0.757, however, the pLM behind the former is considerably smaller than that of the latter, see Table 2; esm2_t30 is a 150M parameter model with embedding size of 640, whereas esm2_t33 has 650M parameters and an embedding size 1280. The improvement ST-esm2_t30 offers is not only more pronounced in terms of accuracy, but also in APS (average precision score) compared to any of the current state-of-the-art, e.g. ESM-MSA-P or NetSolP, where 'P' at the end indicates that these models used pre-trained embeddings and only the prediction heads were finetuned, see [30]. For a fair comparison, we evaluated the 5-fold runs of the ST -based models and NetSolP approaches on an independent validation set taken from [30]. Among those, ST-esm2_t30 retains its marginal lead and shows better generalisation results, see Table 3 in Appendix A.3.2. Figure 3 shows a visual comparison of the two approaches in terms of accuracy and model sizes, see Appendix A.3.2.

Table 2: NetSol benchmark on the PSI dataset - solubility prediction

| Predictor | Num. of params (m) | Metrics | | | |
|---|---|---|---|---|---|
| | | AU-PRC | APS | recall | accuracy |
| DeepSol S2 | - | .67 ±0.04 | .81 ±0.05 | – | .54 ±0.02 |
| ESM-MSA-P | - | .75 ±0.03 | .76 ±0.02 | – | .71 ±0.03 |
| NetSolP | - | .73 ±0.02 | .74 ±0.03 | – | .70 ±0.02 |
| `ST-esm2_t12` | 35 | .745 ±0.038 | .843 ±0.034 | .711 ±0.033 | .711 ±0.033 |
| `ST-esm2_t30` | 150 | .768 ±0.039 | .857 ±0.033 | **.738** ±0.039 | **.738** ±0.039 |
| `ST-esm2_t33` | 650 | **.771** ±0.028 | **.864** ±0.026 | .733 ±0.020 | .733 ±0.020 |
| `NetSolP-esm2_t12` | 35 | .734 ±0.037 | .838 ±0.033 | .699 ±0.029 | .699 ±0.029 |
| `NetSolP-esm2_t33` | 650 | .757 ±0.027 | .849 ±0.027 | .694 ±0.043 | .694 ±0.043 |

## 5 Discussion

We presented a pLM finetuning approach based on Sentence Transformers integrated in a novel data augmentation procedure and showed through two representative examples how these versatile tools achieve state-of-the-art performance. To remain transparent around the challenges regarding the application of this approach poses, and to outline potential future research directions, we collected the following points.

**Random seed** That the performance of finetuning STs, and in general BERT like models, depends so much on random seeds, is a known issue in classic NLP, see e.g. [29]. Ultimately this can also be viewed as a *subset selection* problem. Nonetheless, seed optimization methods that can offer partial solutions exist. One such example is to monitor the ST's performance on a separate development set during the training, and apply early stopping in such a way that only the best of, say, 5 models gets trained all the way to the end. The authors of [29] reported strong correlation between performances of the fully trained model and its version taken from the early phase of the training. This could potentially work in the protein domain when only the embeddings from a trained ST are used as inputs to a prediction head, e.g. solubility prediction. However, when those are combined or concatenated with other inputs, this approach will likely break down.

**Pre-training pLM with contrastive learning at scale** The SETFIT approach we presented above can be viewed as a contrastive pre-training with limited data, and part of the reason for the performance gains seem marginal is because we do not use STs that were pre-trained large scale with contrastive objectives. In classic NLP now it is a standard practice to first pre-train a language model (LM) with (masked) language modelling objective and follow it in a second stage with a contrastive pre-training, see [16]. While this may be more challenging in the protein domain, we hypothesize that this can still be achievable through investing in careful data collection and data set design. Even some of the latest research papers apply the same standard "single stage" pre-training approach, see e.g. [6].

**pLM and ST size** We showed that there is plenty of value to be extracted from smaller pLMs. However, one of the most commonly used mid-sized `esm` models, `esm2_t33_650M_UR50D` currently would require either a larger 80GB GPU or a sharded ST, as it runs into out of memory issues on a single NVIDIA DGX A100 40GB GPU. Currently we are working on releasing an implementation of ST, which would allow this kind of training, i.e. on limited resources.

**Decoder-only models and ST** There are already some promising work along sentence transformers applied to GPT-style models, see e.g. [14]. It would be interesting to test whether some of those tools can be lifted to the protein domain as well.

**Search tasks** There are many use cases of S-BERT models in semantic search and various retrieval tasks in classic NLP. Those types of search tasks are also known in the protein domain. Hence, it will be interesting to see, how these tools could also help in those situations and not just in prediction tasks demonstrated here.

## Acknowledgements

The first author would like to thank Acyr Locatelli for insightful discussions.

## References

[1] T. Bepler and B. Berger. Learning protein sequence embeddings using information from structure. *arXiv preprint arXiv:1902.08661*, 2019.

[2] M. F. Chen, D. Y. Fu, D. Adila, M. Zhang, F. Sala, K. Fatahalian, and C. Ré. Shoring up the foundations: Fusing model embeddings and weak supervision. *arXiv preprint arXiv:2203.13270*, 2022.

[3] A. D. Conte, M. Mehdiabadi, A. Bouhraoua, A. Miguel Monzon, S. C. E. Tosatto, and D. Piovesan. Critical assessment of protein intrinsic disorder prediction (caid) - results of round 2. *Proteins: Structure, Function, and Bioinformatics*, 2023.

[4] R. Dass, F. A. A. Mulder, and J. T. Nielsen. ODiNPred: comprehensive prediction of protein order and disorder. *Scientific Reports*, 10(1):14780, Sept. 2020.

[5] A. Del Conte, A. Bouhraoua, M. Mehdiabadi, D. Clementel, A. M. Monzon, C. predictors, S. C. E. Tosatto, and D. Piovesan. CAID prediction portal: a comprehensive service for predicting intrinsic disorder and binding regions in proteins. *Nucleic Acids Research*, 51(W1):W62–W69, 05 2023.

[6] A. Elnaggar, H. Essam, W. Salah-Eldin, W. Moustafa, M. Elkerdawy, C. Rochereau, and B. Rost. Ankh : Optimized protein language model unlocks general-purpose modelling. *bioRxiv*, 2023.

[7] A. Elnaggar, M. Heinzinger, C. Dallago, G. Rehawi, W. Yu, L. Jones, T. Gibbs, T. Feher, C. Angerer, M. Steinegger, D. Bhowmik, and B. Rost. Prottrans: Towards cracking the language of lifes code through self-supervised deep learning and high performance computing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021.

[8] D. Y. Fu, M. F. Chen, M. Zhang, K. Fatahalian, and C. Ré. The details matter: Preventing class collapse in supervised contrastive learning. *Computer Sciences & Mathematics Forum*, 3(1), 2022.

[9] M. Heinzinger, M. Littmann, I. Sillitoe, N. Bordin, C. Orengo, and B. Rost. Contrastive learning on protein embeddings enlightens midnight zone. *NAR Genomics and Bioinformatics*, 4(2):lqac043, 06 2022.

[10] D. Hesslow, N. Zanichelli, P. Notin, I. Poli, and D. Marks. Rita: a study on scaling up generative protein sequence models. *arXiv preprint arXiv:2205.05789*, 2022.

[11] D. Ilzhöfer, M. Heinzinger, and B. Rost. Seth predicts nuances of residue disorder from protein embeddings. *Frontiers in Bioinformatics*, 2, 2022.

[12] Z. Lin, H. Akin, R. Rao, B. Hie, Z. Zhu, W. Lu, N. Smetanin, A. dos Santos Costa, M. Fazel-Zarandi, T. Sercu, S. Candido, et al. Language models of protein sequences at the scale of evolution enable accurate structure prediction. *bioRxiv*, 2022.

[13] J. Meier, R. Rao, R. Verkuil, J. Liu, T. Sercu, and A. Rives. Language models enable zero-shot prediction of the effects of mutations on protein function. *bioRxiv*, 2021.

[14] N. Muennighoff. Sgpt: Gpt sentence embeddings for semantic search. *arXiv preprint arXiv: 2202.08904*, 2022.

[15] N. Muennighoff, N. Tazi, L. Magne, and N. Reimers. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*, 2023.

[16] A. Neelakantan, T. Xu, R. Puri, A. Radford, J. M. Han, J. Tworek, Q. Yuan, N. Tezak, J. W. Kim, C. Hallacy, J. Heidecke, P. Shyam, B. Power, T. E. Nekoul, G. Sastry, G. Krueger, D. Schnurr, F. P. Such, K. Hsu, M. Thompson, T. Khan, T. Sherbakov, J. Jang, P. Welinder, and L. Weng. Text and code embeddings by contrastive pre-training, 2022.

[17] E. Nijkamp, J. Ruffolo, E. N. Weinstein, N. Naik, and A. Madani. Progen2: Exploring the boundaries of protein language models. *arXiv preprint arXiv:2206.13517*, 2022.

[18] P. Notin, M. Dias, J. Frazer, J. M. Hurtado, A. N. Gomez, D. Marks, and Y. Gal. Tranception: Protein fitness prediction with autoregressive transformers and inference-time retrieval. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 16990–17017. PMLR, 17–23 Jul 2022.

[19] P. Notin, L. V. Niekerk, A. W. Kollasch, D. Ritter, Y. Gal, and D. S. Marks. Trancepteve: Combining family-specific and family-agnostic models of protein sequences for improved fitness prediction. *bioRxiv*, 2022.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[21] R. Rao, N. Bhattacharya, N. Thomas, Y. Duan, X. Chen, J. F. Canny, P. Abbeel, and Y. S. Song. Evaluating protein transfer learning with tape. In *NeurIPS*, pages 9686–9698, 2019.

[22] R. M. Rao, J. Meier, T. Sercu, S. Ovchinnikov, and A. Rives. Transformer protein language models are unsupervised structure learners. *bioRxiv*, 2020.

[23] I. Redl, C. Fisicaro, O. Dutton, F. Hoffmann, L. Henderson, B. J. Owens, M. Heberling, E. Paci, and K. Tamiola. ADOPT: intrinsic protein disorder prediction through deep bidirectional transformers. *NAR Genomics and Bioinformatics*, 5(2):lqad041, 05 2023.

[24] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[25] A. Rives, J. Meier, T. Sercu, S. Goyal, Z. Lin, J. Liu, D. Guo, M. Ott, C. L. Zitnick, J. Ma, and R. Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *PNAS*, 2019.

[26] J. A. Ruffolo, J. J. Gray, and J. Sulam. Deciphering antibody affinity maturation with language models and weakly supervised learning. *arXiv preprint arXiv:2112.07782*, 2021.

[27] R. Singh, S. Sledzieski, B. Bryson, L. Cowen, and B. Berger. Contrastive learning in protein language space predicts interactions between drugs and protein targets. *Proceedings of the National Academy of Sciences*, 120(24):e2220778120, 2023.

[28] H. Stärk, C. Dallago, M. Heinzinger, and B. Rost. Light attention predicts protein location from the language of life. *Bioinformatics Advances*, 1(1):vbab035, 11 2021.

[29] N. Thakur, N. Reimers, J. Daxenberger, and I. Gurevych. Augmented sbert: Data augmentation method for improving bi-encoders for pairwise sentence scoring tasks. *arXiv preprint arXiv: 2010.08240*, 2021.

[30] V. Thumuluri, H.-M. Martiny, J. J. Almagro Armenteros, J. Salomon, H. Nielsen, and A. R. Johansen. NetSolP: predicting protein solubility in Escherichia coli using language models. *Bioinformatics*, 38(4):941–946, 11 2021.

[31] L. Tunstall, N. Reimers, U. E. S. Jo, L. Bates, D. Korat, M. Wasserblat, and O. Pereg. Efficient few-shot learning without prompts. *arXiv preprint arXiv: 2209.11055*, 2022.

[32] M. Wenzel, E. Grüner, and N. Strodthoff. Insights into the inner workings of transformer models for protein function prediction. *arXiv preprint arXiv: 2309.03631*, 2023.

## A Appendix

### A.1 A brief overview of Sentence Transformers

Sentence Transformers offer two main tools, BiEncoders (BE) and Cross-Encoders (CE). BEs, or S-BERTs , use a pooling operation, typically `[CLS]`, `mean` or `max`, on the embeddings extracted by a BERT model. In case of an input sequence pair, $a$ and $b$, their pooled embeddings $u_a$ and $u_b$ are compared using e.g. cosine similarity, as per equation 1, which is optimised using a mean-squared error loss. Apart from this, the authors of [24] introduced and experimented with two other objective functions. We briefly mention the triplet objective function, as that can be equally meaningful for pLMs, albeit we are not using it in this paper. In this instance, the input is a sequence triplet, $a$, called an *anchor*, $p$, a *positive* and $n$, a *negative* sequence, with their corresponding embeddings $u_a, u_p, u_n$, and the following objective function is used:

$$\max(\|u_a - u_p\| - \|u_a - u_n\| + \epsilon, 0), \tag{2}$$

where $\|\cdot\|$ is some distance metric, typically Euclidean, and $\epsilon$ can also be fine tuned, but originally set to a default value, e.g. 1. The idea behind this objective is to bring the embedding of the positive sequence closer to the embedding of an anchor, and at the same time push the embedding of the negative sequence further away from it. In a pLM setting one could take solubility with a binary sequence label and choose an anchor sequence of 1. In this case, the positive sequence could be another sequence that is from the same class, and a negative that is labelled $0$.

Cross-Encoders, on the other hand, take an input sequence pair concatenated and return a similarity score between $0$ and $1$; a value closer to $1$ indicates higher similarity. The main difference between CEs and BEs is that while BEs can output sequence embeddings, CEs can not and they only score sequence pairs. This in turn also affects their respective applications.

Giving a comprehensive review of these tools is out of the scope of this paper, we refer to [24] as a starting point and in general `https://www.sbert.net/` is a great source.

### A.2 Datasets

#### A.2.1 Disorder prediction

Disorder prediction has been in the forefront of computational biology for decades. The Critical Assessment of protein Intrinsic Disorder (CAID) prediction portal currently has around 30 predictors benchmarked on the CAID2 dataset that contains 348 proteins. We used the Disorder-PDB reference set, see [5, 3], as an independent validation set. Residues of each protein have to be classified as ordered or disordered, and some residues are excluded from the performance measurements. In total $130,877$ residues require predictions, which make up $45.6\%$ of all residues of the $348$ sequences.

Many predictors, e.g. SETH [11] and ADOPT [23] have been developed on the CheZoD dataset [4] that contain z-scores, which are based on experimental Nuclear Magnetic Resonance (NMR) chemical shifts. SETH in particular is one of the few state-of-the-art methods in the CAID benchmark that have submitted valid predictions for all 348 sequences.

We retrieved the CheZoD training and test datasets from ADOPT [23], as their training dataset is slightly more restrictive than that of SETH, and contains only 1159 sequences.

The CheZoD test set contains 117 sequences. Both predictors, SETH and ADOPT use this and in the latter case, sequence similarities between test and train sequences were no more than $20\%$, see [23].

#### A.2.2 Solubility prediction

NetSolP [30] is a transformer based, current state-of-the-art solubility predictor. In [30], the authors have tested many pLMs, including ProtT5 [7], and `esm-12, esm-1b` and `esm-msa` from the first generation of `esm` models [25, 22, 13]. Moreover two types of training strategies were tested: One strategy, when the weights of the transformer were also finetuned, i.e. 'end-to-end', and another one, when only the weights of the (linear) prediction head were optimised and the underlying weights of the pLM were kept frozen.

The PSI dataset was one of few sets used in [30] in a 5-fold cross validation. It contains $12,216$ protein sequences, of which $\sim 66\%$ are soluble; for more details see Section 2.3 in [30]. The task is a binary classification to predict solubility.

As a separate validation, we tested our approach on the Price dataset for solubility, described in Section 2.4 in [30]. This set contains 1323 sequences and does not share more than $25\%$ pairwise sequence identity with the entire PSI dataset, see Section 3.1 in [30].

### A.3 ST based models - training details

Accompanying sample code containing data and some example scripts is available at `https://github.com/PeptoneLtd/contrastive-finetuning-plms`.

#### A.3.1 Disorder predictor

To specify the details of our approach described on a high level in the overview section, we propose to use a residue level representation from `esm2_t33_650M_UR50D` [12] loading from `huggingface` with 8bit precision. To augment the residue level embeddings, we retrieve sequence level embeddings from a fine-tuned ST. For this, we are using the smallest *pre-trained* `esm2_t6_8M_UR50D` to instantiate a BiEncoder with a default `mean` pooling using the `SentenceTransformer` package, see [24].

We bucket the sequences in the training set into $4$ classes and use them as labels of the samples used in the fine-tuning process. We created these labels based on mean and standard deviation of z-scores, which are calculated for each sequence. Based on the median standard deviation of z-scores we first split the set into two and further partition each group into two based on the median value of mean z-scores of that given group. While this may seem an arbitrary choice, the reason for this was to capture an additional level of granularity beyond the mean disorder, while also ensuring that the training set contains equally sized classes. The classes can be naturally interpreted as *Highly disordered*, *Partially disordered*, *Structurally Flexible* and *Structured proteins*. We experimented with using only two classes based on z-scores, however we found that the 4 way split worked better. Similarly, adding more granularity by creating more classes did not yield better results. This is likely because of the relatively limited amount of sequence level data.

Based on a random seed, we sampled $256$ sequences from each class and randomly created positive and negative pairs, where the former (latter) meant sequences have the same (different) class labels. We iterated this sampling process $5$ times to get a training set per seed. Positive pairs were labelled with $1$ and negative pairs with $0$. For optimisation, we used cosine similarity loss, see equation 1. We used $4$ epochs, a batch size of $24$ and $10\%$ of the total training iterations were used as warm up steps. These training hyperparameters were chosen based on simplified grid search with repeated $30$ seeds.

**Prediction head for regression** To keep the setting simple, we reverted to using an ELASTICNET from scikit-learn [20] with parameters $\alpha = 1 \cdot 10^{-4}$, $\texttt{max\_iter} = 5 \cdot 10^5$ and $\texttt{tol} = 5 \cdot 10^{-5}$. One could additionally optimise the prediction head by finding better hyperparameters or other models, however we decided to omit this and kept this routine fixed in this work.

**Prediction head for classification** Given that the CAID dataset uses binary labels, we fitted a classification head on the CheZoD training dataset, where residues with z-score less (or greater) than $8$ were labeled with $1$ (or $0$) and considered disordered (or ordered). Similarly as above, we decided to use a simple SGDClassifier from scikit-learn [20] with the parameters $\alpha = 1 \cdot 10^{-4}$, $\texttt{max\_iter} = 5 \cdot 10^5$, $\texttt{tol} = 5 \cdot 10^{-5}$, $\texttt{penalty} = \texttt{elasticnet}$, $\texttt{loss} = \texttt{log\_loss}$ and $\texttt{random\_state} = 72$.

**Experimental details** In the first experiment, we used the entire CheZoD training set with no separate validation set to fit our model and tested it on the CheZoD test set. We ran 200 random seeds, each resulting in a model, and we ranked the models based on Spearmann correlations on the test set. We averaged the top 5 models and labelled it as `ST-avg`. One end-to-end run included creating the training set for the BE, model fitting, retrieving the embeddings for the test set and fitting two prediction heads (one regression and one classification). Nevertheless, the computational workload was still relatively low as it took on average around 5 minutes on a single NVIDIA DGX A100 40GB

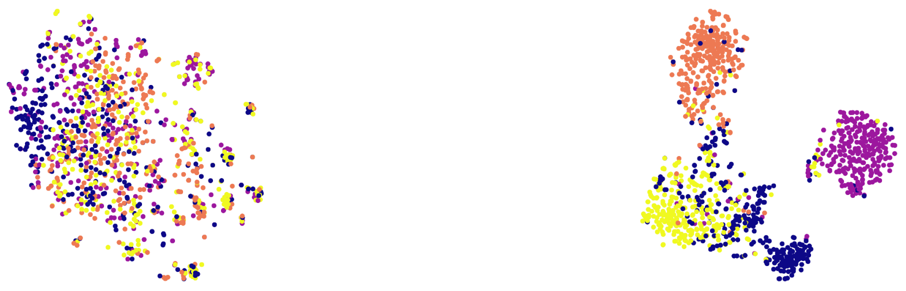

Figure 2: Embeddings of the sequences in the CheZoD training set extracted from a plain pre-trained `esm2_t6_8M_UR50D` (on the **left**) and from a finetuned BiEncoder (on the **right**) that was initialised by the same `esm` model. The four colours represent the disorder classes described in the Experiments section. We see that the embeddings from the BE are more aligned and clustered according to the classes. We used t-SNE from scikit-learn [20] to generate the 2d representations.

GPU. Note that we cached the residue level embeddings from `esm2_t33_650M_UR50D`. We observed that after 100 seed runs typically $2 - 3$ instances were above $0.73$ correlation already.

### A.3.2 Solubility predictor

STs can be applied more straightforwardly. We experiment with three different settings, one where a BiEncoder is initialised with `esm2_t12_35M_UR50D` (‘`esm2_t12`’), a second one with `esm2_t30_150M_UR50D` (‘`esm2_t30`’), and the third one with `esm2_t33_650M_UR50D` (‘`esm2_t33`’). All settings use `mean` pooling. Similarly as before, given a random seed, we sample $N_{\texttt{training}} = 512$ sequences from each - soluble and insoluble - class. We iterate through each protein in this set and select randomly a positive (i.e. sequences of the same class) and a negative pair (different class). We repeat this iteration $N_{\texttt{seq-pairs}} = 5$ times. We run each model training for `num_epoch` $= 3$ epochs with $10\%$ of total iterations as warm up steps and optimise cosine similarity loss. Batch sizes are set to $16$ for the smaller `esm2_t12`, to $6$ for `esm2_t30`, and to $6$ for the largest `esm2_t33`. As for the other hyperparameters, choices were made based on a simplified grid search with repeated $30$ seeds. In all cases, the default learning rate of $2 \cdot 10^{-5}$ was used. Note that using the standard Sentence Transformer package (`https://github.com/UKPLab/sentence-transformers`), we could fit the largest model (‘`esm2_t33`’) only on NVIDIA DGX A100 80GB GPUs.

**Classification head** Once the embeddings from the finetuned STs are retrieved, we fit a simple SGDClassifier using scikit-learn [20] and we keep the same parameters as before for the disorder classification, i.e. $\alpha = 1 \cdot 10^{-4}$, `max_iter` $= 5 \cdot 10^5$, `tol` $= 5 \cdot 10^{-5}$, `penalty` $=$ `elasticnet`, `loss` $=$ `log_loss`, and `random_state` $= 72$. As we wanted to keep this routine simple and focus on the concept of our approach, we omitted any additional hyperparameter or model search in this stage.

**Experimental details** We tested our proposed approach with three finetuned STs initialised with `esm2_t12`, `esm2_t30`, and `esm2_t33`. In all cases, we ran a random search with $100$ seeds per fold. These random numbers were generated for each fold, but kept the same across model sizes to get a clearer insight as to how model size might affect performance. Apart from benchmarking against 3 models from NetSol [30], the current state-of-the-art, we also evaluated two baseline models, referenced as `NetSolP-esm2_t12` and `NetSolP-esm2_t33`, that relied on frozen mean representations extracted from `esm2_t12_35M_UR50D` and `esm2_t33_650M_UR50D`, respectively, coupled with a simple prediction head, which we kept the same as for the finetuned ST models. Note that results on the three predictors from NetSol were taken as reported in [30] and hence only displayed to 2 decimals, see Table 2.

Table 3: Validation results on the Price dataset - solubility prediction

| Predictor | Num. of params (m) | Metrics | | | |
|---|---|---|---|---|---|
| | | AU-PRC | APS | recall | accuracy |
| ST-esm2_t12 | 35 | .699 ±0.011 | .800 ±0.007 | .678 ±0.008 | .678 ±0.008 |
| ST-esm2_t30 | 150 | .752 ±0.013 | .835 ±0.010 | .708 ±0.013 | .708 ±0.013 |
| ST-esm2_t33 | 650 | **.758** ±0.010 | **.837** ±0.007 | **.709** ±0.012 | **.709** ±0.012 |
| NetSolP-esm2_t12 | 35 | .733 ±0.004 | .824 ±0.004 | .688 ±0.012 | .688 ±0.012 |
| NetSolP-esm2_t33 | 650 | .750 ±0.007 | .835 ±0.006 | .690 ±0.020 | .690 ±0.020 |

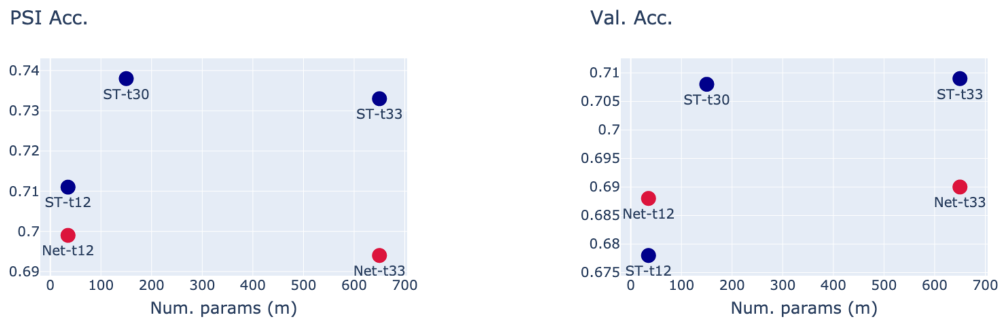

Figure 3: Comparison of model sizes and accuracy on the solubility datasets, PSI (on the **left**) and the Validation set (on the **right**). ST based models (blue dots) typically outperform NetSol counterparts (red dots), which are standard pLM based models. Note that the ST -t30 model (150 million parameters) performs and generalises better than the $4.3\times$ larger standard model Net-t33 (650 million parameters). However, the smallest ST -t12 model (35 million parameters) does not generalise as well as the similarly sized standard model Net-t12.

## A.4 CAID benchmark on disorder prediction

In order to properly validate ST-avg, we benchmarked it on the CAID2 dataset. Only the top 7 predictors were taken, see Table 4. Although ST based models (ST-top being the best of the 5 models) performed among the best ones, certainly the best among those with 100% coverage; marked by $^*$, a decisive comparison remains difficult mainly because not all predictors submitted valid predictions for all 348 sequences in the CAID2 dataset. The Coverage column of Table 4 reports the percentage of sequences for which valid predictions were found. Despite we managed to reproduce the performance metrics that were reported on the CAID website, these were calculated only on those sequences for which valid predictions were submitted. To provide a fairer comparison, we also evaluated the ST-top only on those sequences for which the other candidate predictor had valid predictions. These metrics are displayed in the rightmost column in Table 4.

Table 4: CAID2 benchmark - disorder prediction

| | Metrics | | | Coverage | ST-top |
|---|---|---|---|---|---|
| Predictor | AU-PRC | APS | f1 | (in %) | (AU/APS/f1) |
| SPOT-Disorder2 | .949 | .928 | .860 | 81.6 | .942/.914/.855 |
| SETH-0* | .930 | .893 | .830 | 100 | .938/.899/.841 |
| SETH-1* | .911 | .853 | .795 | 100 | .938/.899/.841 |
| flDPnn2 | .894 | .809 | .739 | 99.4 | .938/.899/.840 |
| DisoPred | .919 | .859 | .784 | 95.1 | .938/.896/.838 |
| AlphaFold-rsa | .944 | .916 | .849 | 85.9 | .935/.897/.839 |
| metapredict* | .932 | .877 | .819 | 100 | .938/.899/.841 |
| ST-avg* | .935 $\pm$0.002 | .892 $\pm$0.004 | .832 $\pm$0.004 | 100 | |
| ST-top* | .938 | .899 | .841 | 100 | |