Protein Sequence Domain Annotation using Language Models

Arpan Sarkar* **Kumaresh Krishnan*** **Sean R. Eddy**
Harvard University
arpan_sarkar@g.harvard.edu
{kumaresh_krishnan, seaneddy}@fas.harvard.edu

Abstract

Protein function inference relies on annotating protein domains via sequence similarity, often modeled through profile Hidden Markov Models (profile HMMs), which capture evolutionary diversity within related domains. However, profile HMMs make strong simplifying independence assumptions when modeling residues in a sequence. Here, we introduce PSALM (Protein Sequence Annotation with Language Models), a hierarchical approach that relaxes these assumptions and uses representations of protein sequences learned by protein language models to enable high-sensitivity, high-specificity residue-level protein sequence annotation. We validate PSALM's performance on a curated set of "ground truth" annotations determined by a profile HMM-based method and highlight PSALM as a promising alternative for protein sequence annotation.

1 Introduction

Proteins are composed of distinct structural and functional units conserved through evolution, known as domains. The primary aim of protein sequence annotation is to locate and characterize these domains within a given sequence. Insight into the individual functions of these domains, which may act independently or in concert with neighboring domains, may shed light on the overall biological role of the protein (Fig. 1). Since experimental characterization of protein function can be difficult, function is often inferred and annotated through sequence similarity (homology) to domains with known function [1, 2]. As the size of protein sequence databases and the number of protein sequences with unknown function continue to grow rapidly [3], methods for annotating protein sequences have been critical for exploiting this wealth of information about the molecular basis and evolutionary trajectory of life.

Databases of protein domain families, like Pfam [6], categorize millions of protein sequences into approximately 20,000 domains. The state of the art uses profile hidden Markov models (profile HMMs) to identify domains [7] and profile/profile comparison to identify homologous domains [8]. Annotation can be achieved by using databases of protein families to compare a protein sequence against 20,000 profiles rather than scanning against millions of individual sequences. Domain-based annotation goes beyond classifying the whole protein sequence; it identifies the domain composition as well as the boundaries of each domain. This "domain annotation" requires annotation at the residue level, labeling each amino acid symbol in the sequence. Domain annotation helps with function inference and avoids the "transitive identification catastrophe", where sequence-level annotations inferred from the presence of one domain can erroneously transfer between sequences due to homology of an unrelated domain [9].

Profile HMMs make strong simplifying independence assumptions when modeling residues in a protein sequence, and there is considerable interest in developing more powerful technologies that

*These authors have equal contribution

Preprint. Under review.

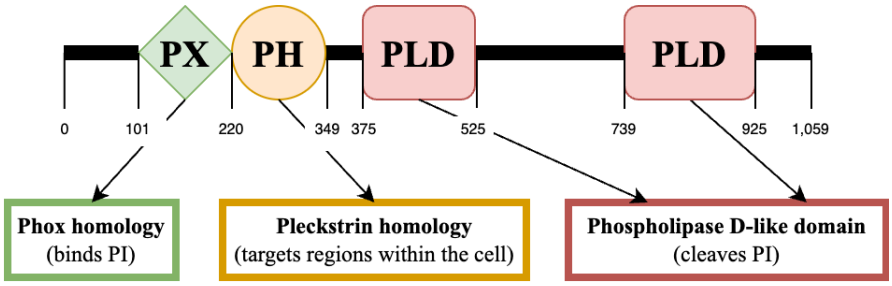

Figure 1: **Annotated domain architecture of a human phospholipase D1 protein** (Q59EA4) [4, 5], featuring PX (phox), PH (pleckstrin homology), and PLD (phospholipase D-like) domains. Together, the function of these domains suggest that the full length protein (1,059 amino acids) is involved in phosphatidylcholine (PI) cleavage and intracellular signaling, consistent with experimental evidence.

relax these assumptions, enabling the recognition of distant evolutionary relationships between protein sequences with greater sensitivity. Most attempts to use deep learning methods for protein sequence similarity recognition have focused on recognition/classification of entire protein sequences [10–14], rather than recognition of individual domain subsequences in a longer target sequence.

In this work, we introduce Protein Sequence Annotation with Language Models (PSALM), that we use to extend the capabilities of ESM-2, a pre-trained general-purpose protein language model [15], and predict *residue-level* sequence annotations with greater sensitivity and specificity than state-of-the-art profile HMM-based methods. Predicted homology from PSALM accurately annotates domain boundaries, multi-domain proteins, conserved domains across distantly-related protein sequences, and even domains that are currently unannotated.

2 Related Work

2.1 Profile HMMs

Profile HMMs use curated multiple sequence alignments (MSAs) of related domains, which reveal patterns of conservation and variability at the residue level, to model consensus using "match", "insert", and "delete" hidden states [2, 16]. Profile HMMs assume that the observed residues are conditionally independent given the hidden state. While this assumption simplifies the modeling process, it may limit the ability of profile HMMs to capture complex dependencies between residues in a sequence. These models serve as templates for comparison against the sequence of interest, enabling the identification of domains by finding subsequences that match the profile HMMs. Sequences with multiple, unrelated domains will require the use of multiple profile HMMs for annotation. Different databases organize related domains into MSAs and profile HMMs at varying levels of granularity, enabling profile-based annotations at the superfamily [17], family [6], and sub-family [18] levels.

2.2 Deep Models

Recent efforts to apply deep learning methods to predict protein function from sequence have either focused on predicting ontology-based functional annotation at the sequence level [19–21] or recognizing homology at the sequence level [11–14]. To our knowledge, ProtENN, an ensemble of convolutional neural networks, represents the first attempt to predict Pfam domains directly from protein sequences [10]. ProtENN, however, is constrained to make one domain prediction per input sequence and cannot natively identify domain boundaries or multiple domains within a sequence without ad hoc post-processing. Additionally, ProtENN cannot provide information on the contribution of an individual residue to a predicted annotation.

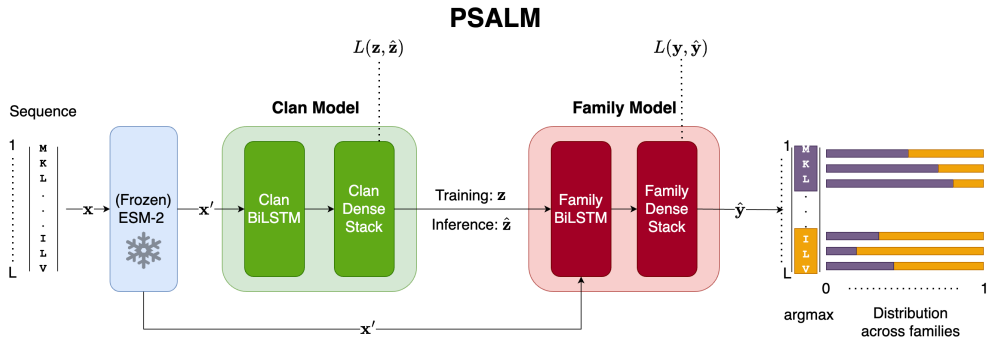

Figure 2: **Overview of residue-level protein sequence annotation with PSALM.** A sequence $\mathbf{x}$ of length L is embedded as $\mathbf{x}'$ with a frozen ESM-2. The PSALM clan and family models predict the clan annotations $\hat{\mathbf{z}}$ and family annotations $\hat{\mathbf{y}}$, respectively, and are trained to minimize cross-entropy loss $L(\cdot)$. Here, the example outputs are predicted across a set of 2 families.

3 Results

3.1 Problem Formulation

Here, we formalize the residue-level sequence annotation problem as a mapping from a protein sequence $\mathbf{x} = (x_1, x_2, \ldots, x_L)$ to a sequence of protein domain families $\mathbf{y} = (y_1, y_2, \ldots, y_L)$. For residue i in a sequence, x_i is an index $1 \ldots 25$ representing the i-th amino acid character (20 canonical, 2 non-canonical, and 3 ambiguous amino acid characters), and y_i is an index $1 \ldots D$ representing the i-th protein domain family annotation, with $D+1$ for none. Approximately 23% of protein sequences and 47% of residues across all sequences in UniProt do not belong to any Pfam domain family [6]. The goal of residue-level annotation is to learn a model that predicts domain family annotations for each residue in a protein sequence:

$$\hat{y}_i = \arg\max_f P(Y_i = f|\mathbf{x}), \tag{1}$$

where $P(Y_i)$ is the distribution over $D+1$ family annotations for a given residue i.

3.2 Protein Language Model

Numerically encoding protein sequences is necessary as an input for machine learning tasks such as classification. Protein language models (pLMs) learn vector representations of both individual residues and full-length protein sequences, which capture long-range interactions, predict function via transfer learning, and achieve state-of-the-art performance in several structure prediction tasks [22–25]. We use ESM-2, a pre-trained general-purpose pLM [15], to generate residue-level sequence embeddings $\mathbf{x}'$ for a given sequence $\mathbf{x}$. ESM-2 was trained using a BERT-style masked token prediction task [26], enabling it to capture contextual information and dependencies within protein sequences and allowing us to replace $\mathbf{x}$ with $\mathbf{x}'$.

3.3 PSALM

We introduce PSALM (Protein Sequence Annotation using Language Models), a method to predict domains across a protein sequence at the residue-level. PSALM uses a hierarchical approach that considers both individual protein domain families and clans, which are collections of evolutionarily related (homologous) protein domain families categorized by Pfam [27]. In Pfam 35.0, approximately 45% of the 19,632 Pfam families are grouped into 655 clans, and a family can only belong to at most one clan. While our primary aim is to predict protein domain families at each residue, modeling clans – the super class – is an interpretable intermediate step that aids in identifying areas of functional or structural importance that may not have clear family-level annotations.

This intermediate annotation problem is a mapping from $\mathbf{x}$ to a sequence of Pfam clans $\mathbf{z} = (z_1, z_2, \ldots, z_L)$, where z_i is an index $1 \ldots C$ representing the i-th clan annotation, with $C + 1$ for "non-clan" or $C + 2$ for none. The non-clan annotation describes a residue which belongs to a domain family that is not a member of clan, and none refers to a residue which does not belong to a domain family and thus does not belong to a clan. For a given residue, the PSALM clan and family models learn to predict:

$$\hat{z}_i = \arg\max_c P(Z_i = c|\mathbf{x}') \tag{2}$$

$$\hat{y}_i = \arg\max_f P(Y_i = f|Z_i = C(f), \mathbf{x}')P(Z_i = C(f)|\mathbf{x}'), \tag{3}$$

where $C(f)$ is the clan label to which family f belongs, and $P(Z_i)$ is the distribution over all $C + 2$ clan annotations for a given residue i. The inclusion of a separate clan prediction task ensures the interpretability of the clan model, preventing it from becoming an abstract hidden state. The family model is trained via the teacher forcing algorithm [28], where it is provided the correct clan annotation for each residue in order to mitigate error propagation.

The clan and family models follow a similar structure, where protein sequences are initially embedded at the residue level using a pre-trained and frozen instance of ESM-2. The resulting embeddings are then passed into a bidirectional Long Short-Term Memory (BiLSTM) layer to capture sequential dependencies [29] in the forwards and backwards directions, each of which, like profile HMMs, are strongly linear, left-right models [2]. The output from the BiLSTM layer is subsequently decoded using a stack of three dense layers, scaled to the number of clans or family labels, to produce logits across the prediction space. Probabilities are computed by applying softmax to the logits generated by each model.

4 Benchmarking

4.1 Data

In many machine learning contexts, data samples are often assumed to be independent instances drawn from a distribution of the data, justifying random training-test splits. However, this assumption does not hold for sequences in protein domain families, which share evolutionary relationships. Random data splits for protein sequences may lead to performance overestimation, motivating the need to explicitly consider sequence similarity when partitioning data into distinct training and test sets [30–34].

To achieve this, we begin by collecting the 1.2M "seed domains" from Pfam-A Seed 35.0, a set of curated, representative domains for each domain family that are used to build the 20K Pfam profile HMMs [6]. We apply BLUE [34], a graph-based sequence splitting algorithm, to partition the seed domains into preliminary training and test sets, defining an edge between two domains as their pairwise percent identity (PID; Eq. 4). We use a PID threshold of 25% to split the seed domains, resulting in 560K training domains and 190K test domains. The remaining 450K seed domains were discarded due to sharing $> 25\%$ PID with both test and training sets. We then retrieve the full-length sequences corresponding to these representative training and test domains from UniProt release March 2021, a comprehensive database of 230M protein sequences. From the test set, we eliminate duplicate sequences also present in the training set. All sequences across both training and test sets are annotated via the `hmmscan` tool from HMMER [7] with strict inclusion thresholds (E-value < 0.001, bitscore $\geq$ 30) in order to identify domain hits that constitute a "ground truth". For training and test purposes, family and clan labels are only assigned to ground truth domains. We discard sequences in test that do not contain an annotated ground truth domain represented by at least 20 ground truth domains from the same family in the training set, resulting in 80K test sequences. For each test sequence, we compute the maximum PID between any of its domains to any domain in the training set as a proxy for its distance from the training set (Alg. 1). We partition the test set into five subsets based on this maximum PID (Table 1). Such a partition may result in test sequences that, for example, may be placed in the $80 <$ PID ≤ 100 subset due to a single domain closely related to one in the training set, whereas the test sequence may have several other domains that share significantly lower PID with domains in the training set (this is why the average PID is near the lower bound of the

Table 1: Partition of the test set into 5 subsets

Test splits by max PID	0-20%	20-40%	40-60%	60-80%	80-100%
Num. sequences	4,420	40,247	18,807	9,259	6,328
Num. families	554	2,497	1,988	1,780	1,732
Num. clans	184	414	367	345	324
Domain coverage	65.02%	59.67%	58.64%	61.76%	56.86%
Average PID	18.35%	27.61%	42.45%	58.08%	79.96%

max PID range for many of the test subsets). The domain coverage, defined as the average percent of residues in a sequence that are labeled by Pfam domains, is similar across all test subsets.

We address the potential for unannotated domains to introduce data leakage across the training and test sets by shuffling all subsequences without family and clan labels in the test sequences to disrupt possible domain structures, preserving 0th order residue-composition [1, 7]. In the first phase of training, the unannotated regions are not shuffled, allowing the PSALM models to potentially learn genuine "no-hit" regions. Since PSALM may be sensitive enough to identify unannotated domains, it is further trained with these regions shuffled, to mitigate penalties for "false positives" (with respect to the ground truth annotations). Another source of data leakage may arise from the millions of representative sequences from the UniRef50 database release April 2021 [35] used to train ESM-2. We identify that none of the 4,420 sequences in the $0 <$ PID ≤ 20 test subset were present as representative sequences in this version of UniRef50. However, UniRef50 may contain close homologs to the sequences in this test subset.

4.2 Baseline Methods

We establish two baseline methods for comparison. We build profile HMMs from MSAs of the ground truth domains in the training set, denoted as HMMER*, and use these profiles to annotate the test sequences with `hmmscan`. This allows us to evaluate how a state-of-the-art profile HMM method compares to PSALM when using the same training and testing sets. Additionally, we implement a variant of PSALM, denoted as PSALM-onehot, where one-hot embeddings for each amino acid in a protein sequence are utilized instead of embeddings from the pre-trained protein language model ESM-2. This comparison helps discern whether differences in performance between PSALM and HMMER* are influenced by ESM-2 or solely by the subsequent neural network architecture.

4.3 Implementation Details

Both PSALM and PSALM-onehot are trained for 30 epochs with an additional 5 epochs for fine-tuning on the training set with shuffled unlabeled regions. We use cross entropy loss over the entire sequence for both family and clan annotations, and add an L_1 penalty on family predictions to the family loss for the first 10 epochs in order to promote sparsity. For training and fine-tuning, we use ADAM optimizer [36] with initial learning rate $1\text{e}{-3}$. We employ a learning rate scheduler that reduces the learning rate by a factor of 10 if the loss fails to decrease by more than 10% over 2 consecutive epochs.

The PSALM model consists of 235M parameters (Clan: 69M; Family: 166M), and the PSALM-onehot model consists of 209M parameters (Clan: 56M; Family: 153M). The family and clan models comprising PSALM and PSALM-onehot were each trained on a single NVIDIA A100 80GB GPU for approximately 22 days. To accommodate memory limitations on the GPU, all sequences are truncated to a maximum length of 4096 residues. This truncation strategy only applies to approximately 0.25% of sequences across the training and test sets and does not reflect a model limitation – PSALM can be used to annotate sequences of any length provided enough memory. All procedures from the HMMER tool suite use version 3.4 [7].

4.4 Evaluation

In protein sequence annotation, controlling the specificity of annotations is critical, as false positives can propagate through annotations, potentially impacting database entries, future annotations, and downstream analyses. The false positive rate (FPR) is defined as the fraction of true negative residues

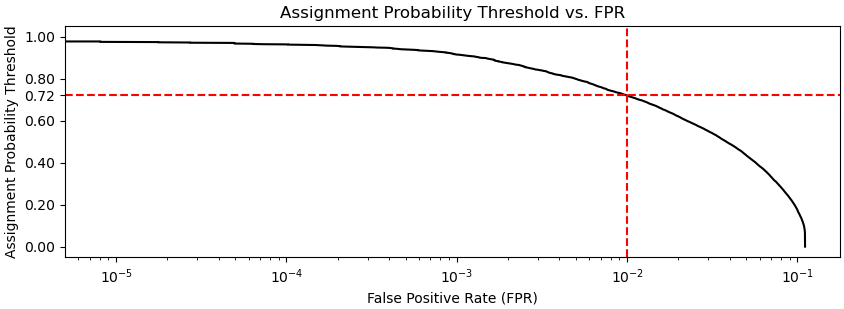

Figure 3: **False positive characterization for PSALM, evaluated on shuffled sequences sampled from the training set.** Residue assignment probability refers to the value above which the corresponding PSALM annotation (index $1 \ldots D$ for the D protein domain families, $D+1$ for none) will be called. At an FPR of 0.01, the residue assignment probability threshold is 0.72. If, at a residue, either no annotation has an assignment probability ≥ 0.72 or the probability of label $D+1$ is ≥ 0.72, then that residue is annotated as none. PSALM achieves a maximum FPR of 0.11 with no residue assignment probability threshold.

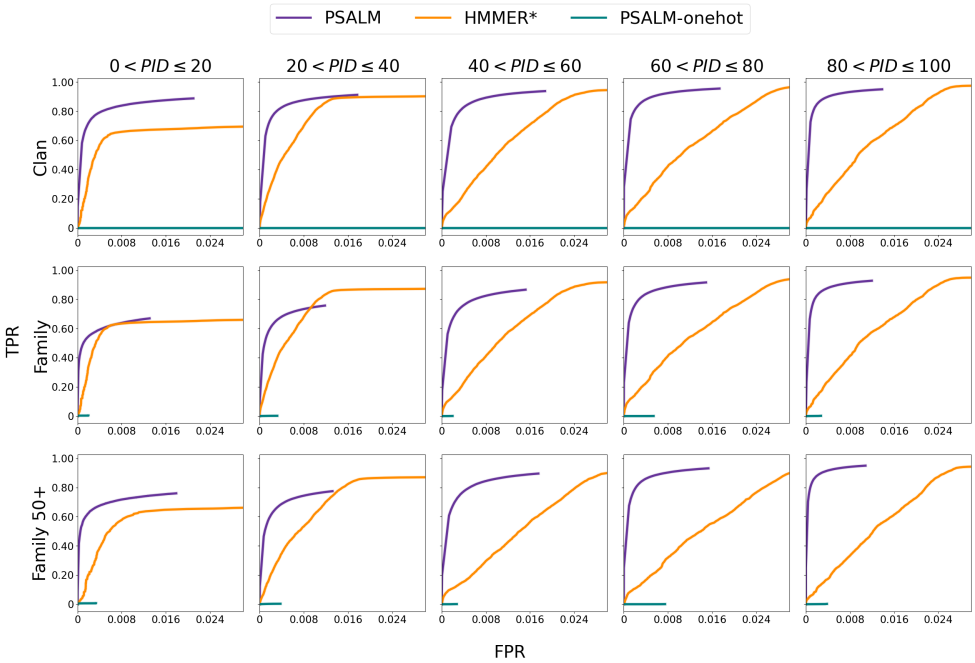

Figure 4: **Residue-level evaluation of sensitivity (within domains) of PSALM (purple), HMMER* (orange), and PSALM-onehot (teal) on each test subset against false positive rate**, plotted with 95% confidence intervals (Appendix B). Evaluations are performed at the clan and family annotation level. Further family level annotations are performed for sequences with domains represented by at least 50 related domains in the training set. The y-axis ranges from 0 to 1 while the *x-axis ranges from 0 to 0.032* to highlight differences at a low FPR.

(shuffled, preserving residue composition) incorrectly identified as homologous to a Pfam protein domain family. We set a residue assignment probability threshold that corresponds to a maximum FPR of 0.01 at the residue level on only true negative sequences (Fig. 3). Using this threshold, we evaluate the sensitivity, or true positive rate (TPR), of PSALM on the test set, where TPR is defined as the fraction of residues in ground truth domains correctly identified.

Across all splits at both the clan and family levels, PSALM demonstrates superior performance in domain annotation, accurately annotating a substantial portion of true domain regions while

minimizing false positive calls compared to HMMER* (Fig. 4). At the clan level, PSALM is more sensitive on the lowest PID test subset and nears max sensitivity with a lower false positive rate than HMMER* across all test subsets. PSALM performs similarly for annotating residues with Pfam family labels and exhibits comparable accuracy to HMMER* on remote homologs with a much lower false positive rate initially. Furthermore, when considering only domains with a substantial representation (at least 50 related domains from the same family) in the training data, PSALM's performance in family annotation improves in sequences more distantly related to the training data, whereas the performance of HMMER* does not seem to benefit from the additional training sequences. PSALM-onehot performs poorly across all test subsets, incorrectly annotating many residues at the clan-level as "non-clan" and many residues at the family-level as "none", limiting both the TPR and FPR.

4.5 Examples

In Figure 5, we compare PSALM annotations with those from InterPro [5], a database integrating domain annotations from various sources, focusing on three protein sequences drawn from the test set. This comparison provides insights into PSALM's performance, highlighting its ability to annotate multiple domains, the advantages of interpretable clan-level annotations, and its high sensitivity in detecting domains.

PSALM is able to identify multiple domains within a single sequence and accurately annotates all 7 domains in the multi-domain glucanase protein (Fig. 5 A). While most predicted domain boundaries closely match the database annotations, some residues at the domain edges exhibit lower assignment probability, reflecting uncertainty in boundary delineation, a common occurrence across annotation methods.

In cases where PSALM may not recognize a domain, it can confidently indicate that a domain exists (non-clan) and further identify the clan of the domain, if applicable. For instance, PSALM fails to annotate, at the family level, approximately two-thirds of the single domain of the stage III sporulation protein AB, a member of the test subset with the lowest max PID (Fig. 5 B). However, PSALM accurately indicates at the clan level that a non-clan domain exists in the correct location.

PSALM is highly-sensitive and is able to annotate domains that Pfam profile HMMs miss but other databases (and their models) hit as well as domains missed by all methods, as illustrated by the putative trigger factor protein example (Fig. 5 C). PSALM identifies both N- and C-terminal bacterial trigger factor domains. This is consistent with the integrated InterPro annotation, even though Pfam lacks a C-terminal annotation for this protein.[2] Additionally, PSALM annotates several residues between both domains that correspond to a FKBP-type isomerase domain, which is missed by all methods for this sequence – we validate this annotation through sequence homology in InterPro.

5 Conclusions and Limitations

In this work, we introduce PSALM, a highly sensitive and specific residue-level protein sequence annotation method. PSALM extends the abilities of self-supervised pLMs trained on hundreds of millions of protein sequences to protein sequence annotation with just a few hundred thousand annotated sequences. PSALM provides residue-level annotations and probabilities at both the clan and family level, enhancing interpretability despite possible model uncertainty. Our comparison with InterPro annotations highlights PSALM's ability to accurately annotate multiple domains within a single sequence, the benefits of clan-level annotations for better interpretability, and PSALM's high sensitivity in detecting domains, including those that are currently unannotated. Furthermore, our experiment with PSALM-onehot, which uses one-hot encodings instead of pLM embeddings, underscores the importance of contextual information provided by protein language models. We provide implementations of the PSALM family and clan models in the `protein-sequence-annotation` Python package and make all code, data, and train-test splits used in this work available (`https://github.com/Protein-Sequence-Annotation/PSALM`) to support reproducibility and further research in protein sequence annotation.

[2]Putative trigger factor protein R6SWH5 is no longer available in UniProt as of November 7th, 2023. However, the entry and all annotations can be accessed either through UniParc [3] or by reannotating with InterProScan [5].

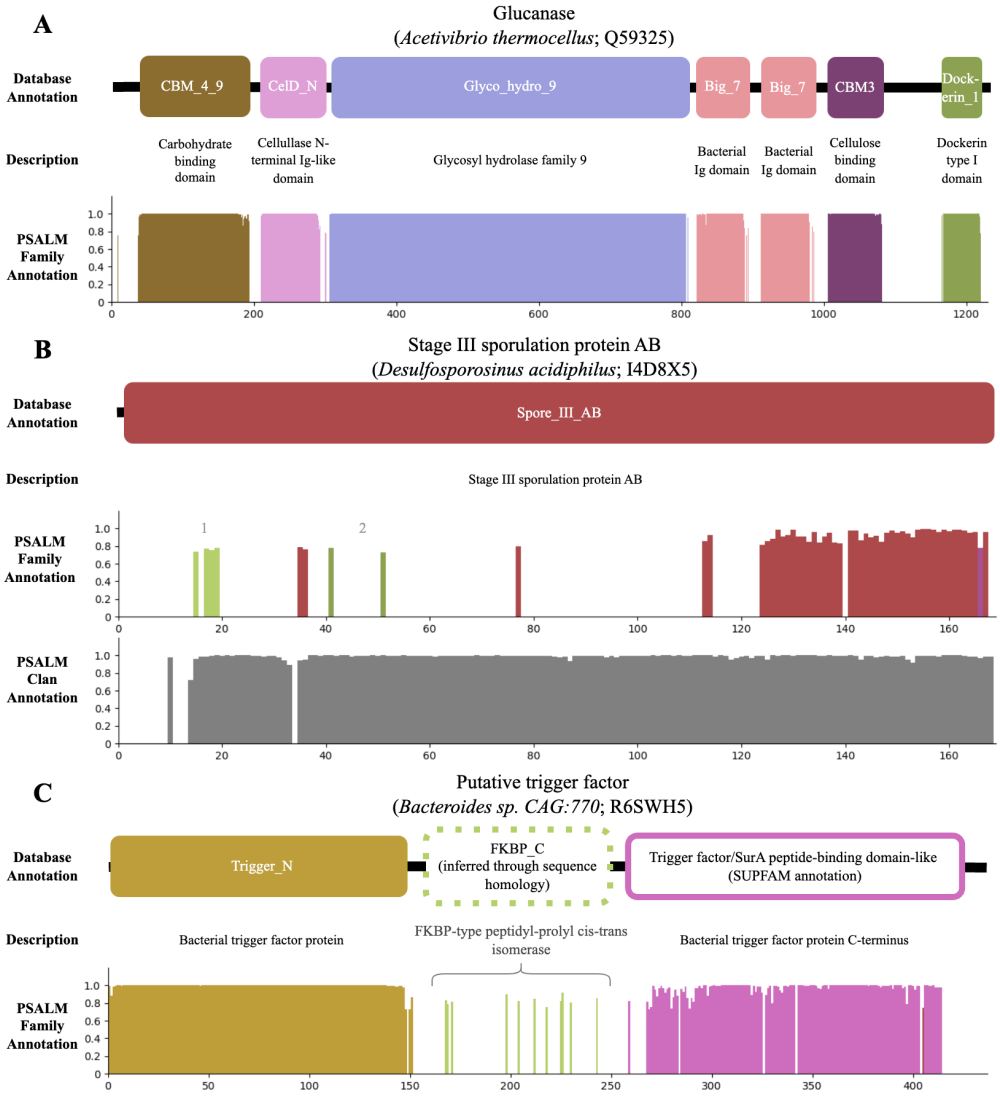

Figure 5: **Comparison of PSALM annotations with those from InterPro for three selected protein sequences. A)** PSALM family annotations on a multi-domain glucanase protein (*Acetivibrio thermocellus*; Q59325) from the $80 <$ max PID ≤ 100 test subset. **B)** PSALM family and clan annotations on a single-domain stage III sporulation protein AB (*Desulfosporosinus acidiphilus*; I4D8X5) from the $0 <$ max PID ≤ 20 test subset. Here the clan annotations (colored grey) correspond to the "non-clan" label, correctly indicating that the annotated residues belong to a domain family that is not a member of a Pfam clan. The predicted annotations at 1 correspond to a bacterial peptidase A24 N-terminal domain (Pfam family PF06750), and the predicted annotations at 2 correspond to a mitoribosomal protein mL52 (Pfam family PF18699). **C)** PSALM family annotations on a putative trigger factor protein (*Bacteroides sp. CAG:770*, R6SWH5) from the $20 <$ max PID ≤ 40 test subset. The filled database domain annotations represent InterPro annotations that correspond to Pfam domains. The solid boundary (unfilled) represents a domain annotated by another database, in this case SUPFAM. The dashed boundary domain annotation represents a domain inferred through sequence homology that is not present in the InterPro annotation – there are approximately 30,000 proteins annotated in InterPro that have similar length and predicted function and mirror the domain architecture that PSALM predicts for putative trigger factor R6SWH5. An example of one of these related proteins is trigger factor protein P44837 (*Haemophilus influenzae*).

There are, however, several limitations to the current PSALM approach. Firstly, despite our efforts to mitigate it, information from the test set may still contribute to training through the millions of sequences used to train ESM-2. While we exclude sequences used to train ESM-2 from our test subset with the lowest maximum PID, homology could still lead to indirect leakage. However, our maximum PID guarantees help minimize this risk. Ideally, retraining ESM-2 on our training data would further alleviate this concern. We have not done this in the present work because of the compute demand for training a pLM like ESM-2. Additionally, PSALM cannot distinguish between repeated domains occurring consecutively or accurately resolve split domains. For example, if a domain repeats immediately after itself, PSALM labels the entire two domain block instead of recognizing two separate domains within it. Similarly, when a domain is split, PSALM identifies the two halves as separate domains from the same family, rather than as originating from a single domain. Nevertheless, the high sensitivity and specificity of PSALM holds promise for discovering new homologs and inferring function. We believe that PSALM can be used to annotate new domains in sequences that are either fully or partially unannotated, furthering our understanding of protein function and evolution. Such usage, however, would require future work to train PSALM on a much broader dataset than the limited benchmark training set, which is relatively small compared to the scale of sequence data problems.

References

[1] William R Pearson. An Introduction to Sequence Similarity ("Homology") Searching. *Current Protocols in Bioinformatics*, 42(1):3–1, 2013.

[2] Sean R. Eddy. Profile hidden Markov models. *Bioinformatics (Oxford, England)*, 14(9): 755–763, 1998.

[3] UniProt: the Universal Protein Knowledgebase in 2023. *Nucleic Acids Research*, 51(D1): D523–D531, 2023.

[4] EMBL-EBI. Protein Classification: What Are Protein Domains? `https://www.ebi.ac.uk/training/online/courses/protein-classification-intro-ebi-resources/protein-classification/what-are-protein-domains/`, 2024.

[5] Typhaine Paysan-Lafosse, Matthias Blum, Sara Chuguransky, Tiago Grego, Beatriz Lázaro Pinto, Gustavo A Salazar, Maxwell L Bileschi, Peer Bork, Alan Bridge, Lucy Colwell, et al. InterPro in 2022. *Nucleic Acids Research*, 51(D1):D418–D427, 2023.

[6] Jaina Mistry, Sara Chuguransky, Lowri Williams, Matloob Qureshi, Gustavo A Salazar, Erik LL Sonnhammer, Silvio CE Tosatto, Lisanna Paladin, Shriya Raj, Lorna J Richardson, et al. Pfam: The protein families database in 2021. *Nucleic Acids Research*, 49(D1):D412–D419, 2021.

[7] Sean R Eddy. Accelerated Profile HMM Searches. *PLoS Computational Biology*, 7(10): e1002195, 2011.

[8] Michael Remmert, Andreas Biegert, Andreas Hauser, and Johannes Söding. HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment. *Nature Methods*, 9(2): 173–175, 2012.

[9] Tobias Doerks, Amos Bairoch, and Peer Bork. Protein annotation: detective work for function prediction. *Trends in Genetics*, 14(6):248–250, 1998.

[10] Maxwell L Bileschi, David Belanger, Drew H Bryant, Theo Sanderson, Brandon Carter, D Sculley, Alex Bateman, Mark A DePristo, and Lucy J Colwell. Using deep learning to annotate the protein universe. *Nature Biotechnology*, 40(6):932–937, 2022.

[11] Michael Heinzinger, Maria Littmann, Ian Sillitoe, Nicola Bordin, Christine Orengo, and Burkhard Rost. Contrastive learning on protein embeddings enlightens midnight zone. *NAR Genomics and Bioinformatics*, 4(2):lqac043, 2022.

[12] Vamsi Nallapareddy, Nicola Bordin, Ian Sillitoe, Michael Heinzinger, Maria Littmann, Vaishali P Waman, Neeladri Sen, Burkhard Rost, and Christine Orengo. CATHe: detection of remote homologues for CATH superfamilies using embeddings from protein language models. *Bioinformatics*, 39(1):btad029, 2023.

[13] Kamil Kaminski, Jan Ludwiczak, Kamil Pawlicki, Vikram Alva, and Stanislaw Dunin-Horkawicz. pLM-BLAST: distant homology detection based on direct comparison of sequence representations from protein language models. *Bioinformatics*, 39(10):btad579, 2023.

[14] Tymor Hamamsy, James T Morton, Robert Blackwell, Daniel Berenberg, Nicholas Carriero, Vladimir Gligorijevic, Charlie EM Strauss, Julia Koehler Leman, Kyunghyun Cho, and Richard Bonneau. Protein remote homology detection and structural alignment using deep learning. *Nature Biotechnology*, pages 1–11, 2023.

[15] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023.

[16] Richard Durbin, Sean R Eddy, Anders Krogh, and Graeme Mitchison. *Biological sequence analysis: Probabilistic models of proteins and nucleic acids*. Cambridge university press, 1998.

[17] Arun Prasad Pandurangan, Jonathan Stahlhacke, Matt E Oates, Ben Smithers, and Julian Gough. The SUPERFAMILY 2.0 database: a significant proteome update and a new webserver. *Nucleic Acids Research*, 47(D1):D490–D494, 2019.

[18] Paul D Thomas, Dustin Ebert, Anushya Muruganujan, Tremayne Mushayahama, Laurent-Philippe Albou, and Huaiyu Mi. PANTHER: Making genome-scale phylogenetics accessible to all. *Protein Science*, 31(1):8–22, 2022.

[19] Yue Cao and Yang Shen. TALE: Transformer-based protein function Annotation with joint sequence–Label Embedding. *Bioinformatics*, 37(18):2825–2833, 2021.

[20] Jiajun Hong, Yongchao Luo, Yang Zhang, Junbiao Ying, Weiwei Xue, Tian Xie, Lin Tao, and Feng Zhu. Protein functional annotation of simultaneously improved stability, accuracy and false discovery rate achieved by a sequence-based deep learning. *Briefings in Bioinformatics*, 21(4):1437–1447, 2020.

[21] Maxat Kulmanov and Robert Hoehndorf. DeepGOPlus: improved protein function prediction from sequence. *Bioinformatics*, 36(2):422–429, 2020.

[22] Tristan Bepler and Bonnie Berger. Learning the protein language: Evolution, structure, and function. *Cell Systems*, 12(6):654–669, 2021.

[23] Roshan Rao, Joshua Meier, Tom Sercu, Sergey Ovchinnikov, and Alexander Rives. Transformer protein language models are unsupervised structure learners. In *International Conference on Learning Representations*, 2020.

[24] Joshua Meier, Roshan Rao, Robert Verkuil, Jason Liu, Tom Sercu, and Alex Rives. Language models enable zero-shot prediction of the effects of mutations on protein function. *Advances in Neural Information Processing Systems*, 34:29287–29303, 2021.

[25] Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, et al. Prottrans: Toward Understanding the Language of Life Through Self-Supervised Learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(10):7112–7127, 2021.

[26] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[27] Robert D Finn, Jaina Mistry, Benjamin Schuster-Böckler, Sam Griffiths-Jones, Volker Hollich, Timo Lassmann, Simon Moxon, Mhairi Marshall, Ajay Khanna, Richard Durbin, et al. Pfam: clans, web tools and services. *Nucleic Acids Research*, 34(suppl_1):D247–D251, 2006.

[28] Ronald J Williams and David Zipser. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation*, 1(2):270–280, 1989.

[29] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9 (8):1735–1780, 1997.

[30] Johannes Söding and Michael Remmert. Protein sequence comparison and fold recognition: progress and good-practice benchmarking. *Current Opinion in Structural Biology*, 21(3): 404–411, 2011.

[31] Ian Walsh, Gianluca Pollastri, and Silvio CE Tosatto. Correct machine learning on protein sequences: a peer-reviewing perspective. *Briefings in Bioinformatics*, 17(5):831–840, 2016.

[32] David T Jones. Setting the standards for machine learning in biology. *Nature Reviews Molecular Cell Biology*, 20(11):659–660, 2019.

[33] Ian Walsh, Dmytro Fishman, Dario Garcia-Gasulla, Tiina Titma, Gianluca Pollastri, Jennifer Harrow, Fotis E Psomopoulos, and Silvio CE Tosatto. DOME: recommendations for supervised machine learning validation in biology. *Nature Methods*, 18(10):1122–1127, 2021.

[34] Samantha Petti and Sean R Eddy. Constructing benchmark test sets for biological sequence analysis using independent set algorithms. *PLOS Computational Biology*, 18(3):e1009492, 2022.

[35] Baris E Suzek, Yuqi Wang, Hongzhan Huang, Peter B McGarvey, Cathy H Wu, and UniProt Consortium. UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics*, 31(6):926–932, 2015.

[36] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[37] William R Pearson. Flexible sequence similarity searching with the FASTA3 program package. *Bioinformatics Methods and Protocols*, pages 185–219, 1999.

[38] Samuel Karlin and Stephen F Altschul. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proceedings of the National Academy of Sciences*, 87(6):2264–2268, 1990.

[39] Sofus Macskassy and Foster Provost. Confidence Bands for ROC curves. 2004.

A Further Details on Section 4.1

A.1 BLUE

We use the BLUE algorithm [34] to split the 1.2M Pfam Seed domains into preliminary train and test sets with a PID threshold of 25%. The pairwise PID between two sequences $\mathbf{x}$ and $\mathbf{y}$ is defined as follows:

$$\text{PID}(\mathbf{x}, \mathbf{y}) = \frac{\text{\# aligned residues}}{\min(\ell(\mathbf{x}), \ell(\mathbf{y}))}, \tag{4}$$

where $\ell(\mathbf{x})$ and $\ell(\mathbf{y})$ represent the lengths of sequences $\mathbf{x}$ and $\mathbf{y}$, respectively. If two domains are in the same family, PID is directly calculated from their seed alignment. If two domains are not in the same family but are in the same clan, they are aligned using the `glsearch` tool from the FASTA3 software package [37], which performs a "global-local" alignment to account for possible large differences in sequence length. If two domains are not in the same clan, they are assumed to share $< 25\%$ PID.

A.2 Maximum PID Calculation

Once the full length test sequences have been retrieved and subsequently filtered, we compute, for each test sequence, the maximum PID between any of its annotated domains and any annotated domain in train via Algorithm 1. Each test sequence is assigned to a single (out of five) test subset based on its maxmimum PID.

Algorithm 1 Percent identity splitting test set

```
Require: train sequences D^tr, test sequences D^te, Pfam family profile HMMs F
  Initialize an empty dictionary-like structure record_max_pids
  for f ∈ F do
      train_domains ← hmmsearch f against D^tr
      test_domains ← hmmsearch f against D^te
      for (domain,sequence_id) ∈ test_domains do
          MSA ← hmmalign domain to train_domains with f
          domain_pids ← esl-alipid MSA
          max_pid ← max(domain_pids)
          if sequence_id not in record_max_pids then
              record_max_pids[sequence_id] ← max_pid
          else if max_pid > record_max_pids[sequence_id] then
              record_max_pids[sequence_id] ← max_pid
          end if
      end for
  end for
  Assign each sequence in D^te to a test split based on max pid
```

The `esl-alipid` tool calculates PID for all pairs of sequences for a given MSA, and is part of the EASEL software package, which can be downloaded together with HMMER [7].

A.3 Ground Truth Annotation

We determine "ground truth" by annotating full length sequence with Pfam Seed profile HMMs using `hmmscan` with strict inclusion criteria (E-value < 0.001, bitscore ≥ 30). The highest-scoring annotation at each residue is taken as ground truth, but additional post-processing is necessary to ensure that "nested" domain structures are retained. This is accomplished by considering the "match strings" that HMMER generates for an alignment. The match strings contain characters that represent matches, where residues align to a given domain profile, and characters that represent inserts, where unaligned residues are inserted into the sequence relative to the domain profile. Annotating the highest-scoring match state at each residue preserves nested domain structure in the ground truth annotations (Fig. 6).

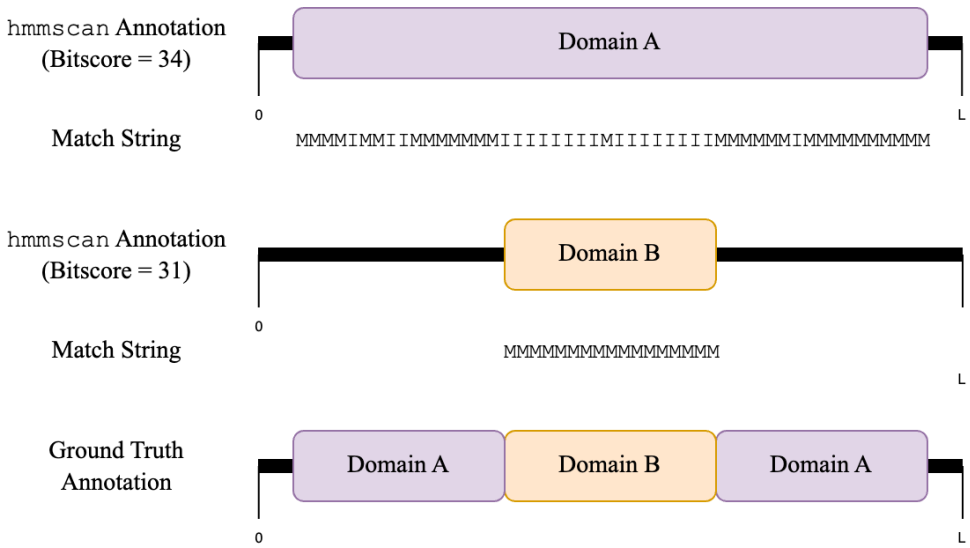

Figure 6: **A schematic of nested domains with two domains A and B in the nested format A-B-A.** As A is annotated with a higher score than B and overlaps with B, annotating residues only via highest score will fail to include domain B. Using the match state strings to identify smoothed maximal segments preserves the nested domain structure in the ground truth annotation.

In a match string, regions with majority matches may contain a few inserts and vice versa. To prevent frequently alternating annotations in the ground truth, we smooth the match and insert states in the match string by identifying maximal scoring segments within the sequence. We assign insert states a positive score and match states a negative score. The segment of the sequence with the greatest aggregate score is known as the maximal segment [38], and all residues in the maximal segment are denoted as insert states. The scores s_i for each state are inferred from the match string for a given sequence:

$$s_i \propto \log\left(\frac{q_i}{p_i}\right), \tag{5}$$

where p_i is the frequency with which the state appears in the match string, and q_i is a state's target frequency, with $\sum_i p_i = 1$ and $\sum_i q_i = 1$. We set the insert state target frequency at 0.85, the match state target frequency at 0.15, and the length threshold at 20, below which maximal segments are ignored.

B Further Details on Section 4.4

In order to evaluate the statistical significance of the benchmarking experiments (Fig. 4), we report confidence intervals on TPR (y-value) [39]. For a given observed FPR (x-value), we calculate the binomial 95% confidence interval:

$$\widehat{\text{TPR}} \pm \frac{1.96}{\sqrt{N}}\sqrt{\widehat{\text{TPR}}(1-\widehat{\text{TPR}})}, \tag{6}$$

where N is the number of residues with a ground truth protein domain family annotation. While N varies across each test subset, it is always greater than 1.5M, meaning that the confidence interval bars are too small to see. This large sample size also validates the normality assumption because both $1.5\text{M}\cdot\widehat{\text{TPR}}$ and $1.5\text{M}\cdot(1-\widehat{\text{TPR}})$ are large.