ProtEx: A Retrieval-Augmented Approach for Protein Function Prediction

Peter Shaw[1*] **Bhaskar Gurram**[2] **David Belanger**[1] **Andreea Gane**[1]

Maxwell L. Bileschi[1] **Lucy J. Colwell**[1,3] **Kristina Toutanova**[1] **Ankur P. Parikh**[1*]

[1] Google DeepMind [2] Google [3] University of Cambridge

Abstract

Mapping a protein sequence to its underlying biological function is a critical problem of increasing importance in biology. In this work, we propose ProtEx, a retrieval-augmented approach for protein function prediction that leverages exemplars from a database to improve accuracy and robustness and enable generalization to unseen classes. Our approach relies on a novel multi-sequence pretraining task, and a fine-tuning strategy that effectively conditions predictions on retrieved exemplars. Our method achieves state-of-the-art results across multiple datasets and settings for predicting Enzyme Commission (EC) numbers, Gene Ontology (GO) terms, and Pfam families. Our ablations and analysis highlight the impact of conditioning predictions on exemplar sequences, especially for classes and sequences less well represented in the training data.

1 Introduction

Proteins perform a wide array of functions within organisms, captured by biologists using ontologies such as Gene Ontology (GO) terms [5], Enzyme Commission (EC) numbers [68], and Pfam families [7]. Mapping proteins to such functional annotations can address key problems in biology, medicine, and chemistry [48, 17]. However, given the expense of wet lab experiments, and the rapid growth of protein sequence databases [69], it is critical to extend coverage using computational protein function predictions. Protein function prediction techniques largely fall into two categories. Homology-based approaches align the query protein to annotated sequences via methods such as BLAST [3] or profile hidden Markov models [18] allowing propagation of a label to the query. More recently, deep learning approaches directly predict a protein's function from its amino acid sequence.

While these approaches have been successful, key obstacles remain. A critical challenge is out-of-distribution generalization across both label and sequence space. For example, recent expansions of the Pfam database were driven by the identification of new classes [44], for which there are few annotated sequences. Moreover, a substantial number of proteins belong to the "dark matter" of the protein universe [17], *i.e.* their sequences are dissimilar from those of any characterized proteins, posing considerable challenges for both homology-based and deep learning methods.

In this work, we propose ProtEx, a method for Protein function prediction via retrieved Exemplars. ProtEx is a semiparametric approach that combines aspects of both non-parametric similarity search based methods and parametric deep learning models to achieve increased accuracy and robustness. ProtEx is inspired by retrieval augmented methods in natural language processing and vision, *e.g.* [37, 47, 27, 79, 41], which show advantages over fully parametric models in capturing tail information [30] and performing few-shot tasks through conditioning on task exemplars [43, 13].

*Correspondence to petershaw@google.com, aparikh@google.com.

Preprint. Under review.

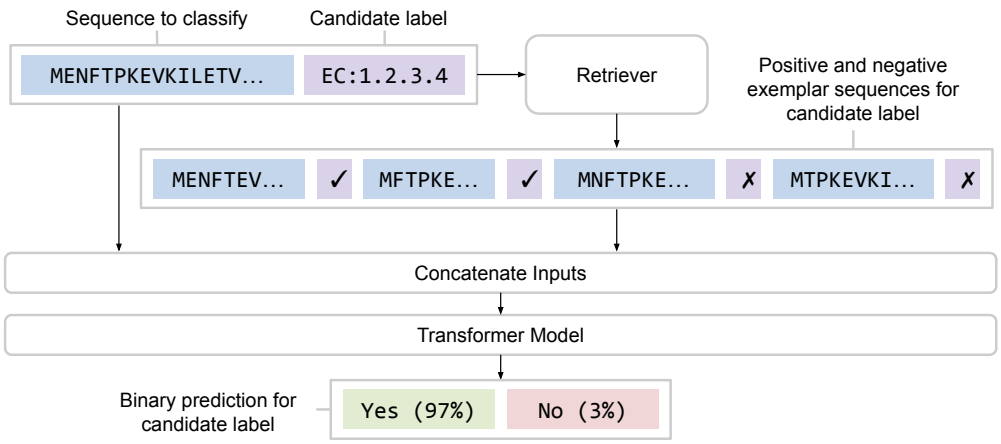

Figure 1: **Method Overview**. Our proposed method, ProtEx, predicts the relevance of various functional annotations for a given query sequence. First, for a given candidate label, positive and negative exemplar sequences are retrieved from the training data. Second, a pre-trained neural model jointly conditions on the query sequence and retrieved sequences to make a prediction.

As shown in Figure 1, given a query protein and candidate label, ProtEx first uses a homology-based retriever (such as BLAST) to obtain a class-conditioned set of positive and negative exemplars from the training set. Our neural model is trained to make a task-based comparison across the set of exemplars and the query to output a binary decision of whether the query sequence has the same label as the positive exemplars. To enable the model to efficiently learn the relationship between query and exemplar sequences, we propose a novel multi-sequence pretraining objective (Figure 2). Our formulation allows us to effectively generalize to rare and unseen classes at test time and better classify sequences that have low sequence similarity with the training data.

We evaluate our approach, ProtEx, on EC number, GO term, and Pfam family prediction tasks, showing that it achieves state-of-the-art performance across multiple settings, and consistently outperforms both existing traditional homology-based approaches and deep learning models. Our stratified analysis shows that ProtEx brings most notable improvements on rare classes and sequences that are far from the training set. We also demonstrate generalization to labels not seen at training time. Finally, our ablations and analyses highlight the efficacy of our pretraining strategy and the model's ability to leverage exemplar sequences for improved classification accuracy.

2 Background and Related Work

We review traditional methods for protein similarity search, neural models for protein function prediction, and related work on retrieval-augmented models from other domains.

Protein similarity search Proteins with similar sequences often perform similar functions, and for a given query sequence, *homologous* sequences can be retrieved from a database using a variety of similarity-based search methods [3, 28, 53]. These enable *homology-based inference* [40], where functional labels from retrieved homologs are transferred to the query. ProtEx can be seen as performing homology-based inference where information from the homologs is aggregated using a learned, non-linear model. A ubiquitous tool for similarity search is BLAST [2, 3], which identifies and scores local sequence alignments using an empirically-derived substitution cost matrix [25]. Our method leverages BLAST and BLAST-inspired techniques to retrieve exemplars (see §4). BLAST also serves as a strong baseline for homology-based inference when used in isolation.

Neural Models for Protein Function Prediction Recent work has shown that deep models mapping a protein sequence to functional predictions can outperform traditional alignment-based techniques. Models can be fit from scratch [35, 58, 10, 9, 19, 59] or fine-tuned from a model pretrained on unlabeled protein sequences [66, 16, 72, 81, 15]. The amino acid sequence largely specifies a protein's structure and function [4], and hence is often used as input to models. However, other

approaches also encode protein structures [63, 56, 34, 22, 84, 36], with broad coverage due to advancements in protein structure prediction [29, 6]. Using structures in addition to or instead of protein sequences, either for prediction or retrieval, is largely orthogonal to the contributions of this paper, and future extensions to ProtEx could incorporate such techniques. Disordered proteins, which are not well characterized by a static structure would need to be handled carefully, however [57]. We do not consider prediction approaches based on protein interaction networks [45, 78, 82], which have demonstrated high accuracy where such networks are already established, but have limited overall coverage.

Of particular relevance are recent methods that combine neural models and retrieval for protein function prediction. Notin et al. [46] combines likelihood scores from a protein language model and an MSA to predict the functional effects of mutations. Hamamsy et al. [24] distills a protein similarity function into a neural network, and Dickson and Mofrad [15] fine-tunes a retrieval model for homology-based inference. The ProtIR method of Zhang et al. [85] uses transductive learning where predictions from a retriever and neural classifier are iteratively updated towards agreement. In contrast to retrieval methods, which form predictions by aggregating pairwise similarity scores over sequences, ProtEx learns to make a prediction conditioned on multiple labeled exemplar sequences. This also differs from methods that use the correlation structure in aligned sets of homologous sequences (MSAs), without accompanying annotations of their function, to inform, *e.g.*, structure prediction [42, 29, 6] or representation learning [52].

Retrieval-Augmented Models and In-Context Fine-tuning Retrieval augmented neural models have shown success for a variety of text generation and classification tasks, with particular strength in recalling long tail knowledge, where even large parametric language models (LMs) struggle [30]. Examples include retrieval-augmented language models, and models for machine translation, question answering, semantic parsing, and text classification [32, 33, 31, 23, 37, 27, 47, 79, 11]. Retrieved context or exemplars come from large-scale unlabeled collections or labeled sets for supervised learning [73, 38]. Computer vision and multimodal models also benefit from retrieval augmentation [41, 51]. A focus in NLP have been methods to adapt a pretrained parametric language model into one that can integrate information from retrieved sequences, and joint training of a retriever and generation model. In this work, we use pre-existing protein similarity models for retrieval and pre-train a protein sequence model to make inferences from multiple proteins.

ProtEx is also related to in-context tuning methods for few-shot tasks [43, 13], where pretrained language models are meta-trained to make predictions given an input and task-relevant exemplars. These works show strong performance on unseen tasks, enabled by the LM's ability to make predictions from an input and a few in-context exemplars. Similarly, ProtEx can be seen as meta-training on multiple binary classification tasks with in-context exemplars and, as shown in Section 4.5, can adapt to new labels without additional fine-tuning.

3 Proposed Method

Given a protein represented as a sequence of amino acids, x, our goal is to predict a set of associated labels, $y \subset \mathcal{L}$, where $\mathcal{L}$ is a set of protein function labels, such as EC numbers, GO terms, or Pfam labels. Our overall approach is visualized in Figure 1. Given a sequence x and candidate label $l \in \mathcal{L}$, we retrieve relevant exemplar sequences. Then, conditioned on x, l, and the positive and negative exemplar sequences, our model predicts the relevance of l to x. Finally, we aggregate predictions across candidate labels to form the predicted label set, $\hat{y}$.

The following sections describe each step in detail. Notably, our method makes an independent prediction for each candidate label, and exemplars are selected conditioned on this label. Given our limited context window, this enables our model to focus on exemplars that are most relevant towards understanding the class boundary of the candidate label.

Retrieving and Sampling Exemplars Given a sequence x and candidate label $l \in \mathcal{L}$, the retriever determines a ranked list of positive exemplars, E_l^p, and negative exemplars, E_l^n, each consisting of sequences from the training data $\mathcal{D}$ with or without the label l, respectively. For our study, we use standard retrieval methods from prior work, such as BLAST [3], and focus our experimentation on the training recipes for models. Retriever details are provided in § 4.2 and Appendix B.5.

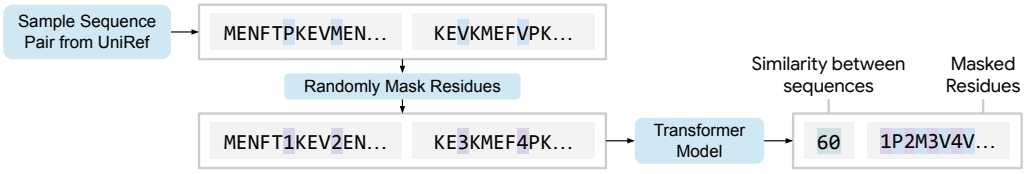

Figure 2: **Pretraining task.** We sample a pair of unlabeled sequences, and the model is tasked with predicting masked residues and a similarity score between the two sequences.

As our context window is limited, the number of positive and negative exemplars included in the model input, denoted as k^p and k^n, is restricted. At inference time, we select the first k^p and k^n exemplars from the ranked lists E_l^p and E_l^n from the retriever. At training time, we also explore sampling of exemplars using uniform or geometric sampling [47]. The latter strategy samples the jth element with probability $\propto p(1-p)^{(j-1)}$, where parameter p provides interpolation between top-k and uniform sampling. Sampling exemplars leads to greater diversity in the training data compared to training for multiple epochs over data generated using deterministic top-k selection. Also, for cases where the training data and evaluation data are not from the same distribution, such sampling can better align the distribution between query sequences and exemplar sequences seen at training time with the distribution seen at inference time. See §4.5 for analysis and details.

Inference Since we consider separate exemplars for each candidate label, this requires running the model up to $|\mathcal{L}|$ (the number of labels) times for each input sequence. However, the computational cost can be reduced by using a candidate label generator, which conditioned on the input sequence x, generates a subset of labels for consideration, $\hat{\mathcal{L}}_x \subseteq \mathcal{L}$. We then only need to generate scores for this subset, which makes inference more efficient at the cost of potentially limiting recall. Candidate label generator details are provided in § 4.2 and Appendix B.5.

For a given x, candidate label l, and retrieved exemplars that depend on both x and l, our model produces a score $s_\theta(x, l, E_l^p, E_l^n)$ with trainable parameters θ. For multilabel prediction tasks (e.g. EC and GO), we can determine our predicted label set based on a score threshold t:

$$\hat{y} = \{l \mid l \in \hat{\mathcal{L}}_x \wedge s_\theta(x, l, E_l^p, E_l^n) > t\}, \tag{1}$$

and varying t produces a tradeoff between precision and recall. For multiclass prediction tasks (e.g. Pfam) where y is always a singleton, we can determine the class with the highest score:

$$\hat{y} = \{\arg\max_{l \in \hat{\mathcal{L}}_x} s_\theta(x, l, E_l^p, E_l^n)\}. \tag{2}$$

Pre-training Prior to fine-tuning, we pre-train models on the task shown in Figure 2. Prior work has pre-trained on single unlabeled sequences, then fine-tuned with a single sequence as input to improve protein function prediction [16, 39]. In contrast, our goal is to train models that implicitly compare a query sequence with exemplar sequences. We thus propose a new pre-training objective over multiple sequences. We sample pairs of sequences with varying degrees of similarity from UniRef90 [67], then implement a version of the span denoising objective of Raffel et al. [50]. We also include a similarity score for the sequence pair in the prediction target. See Appendix A.2 for details and §4 for analysis.

Fine-tuning We construct training examples in the same format as those used at inference time, where both the query sequences and the exemplar sequences are selected from the training data, $\mathcal{D}$. We ensure that the query sequence does not appear in the returned lists of exemplars. For every sequence x and label set y in $\mathcal{D}$, we create a positive example for every label $l \in \hat{\mathcal{L}}_x$ where $l \in y$. Similarly, we can create a negative example for every label $l \in \hat{\mathcal{L}}_x$ where $l \notin y$, or we can optionally subsample negatives to avoid class imbalance (see Appendix B.6). We then generate multiple versions of each training example with different sets of exemplars, as described above.

Architecture and Model Interface As the focus of our study is the pre-training and fine-tuning recipe for our model, we use the general encoder-decoder Transformer [71] architecture of T5 [50].

As demonstrated in Raffel et al. [50], this allows for flexibility in defining different pre-training and fine-tuning tasks without changes to the underlying architecture. We evaluate Small (60M) and Base (220M) sized models, using the Base size for our main results. We also evaluated Transformer variants that offer more efficient handling of long context. See Appendix C.2 for these comparisons.

Our model input is formed by concatenating the candidate label, l, the input sequence, x, and the positive and negative exemplars. Similarly to Raffel et al. [50], to apply our encoder-decoder model as a binary classifier, our model predicts single character sequences corresponding to either positive or negative predictions. At inference time, we determine the score based on the probability assigned to the positive prediction. See Appendix A.1 for further details.

4 Experiments and Analysis

We first introduce the tasks (§4.1) and retrievers (§4.2) that we consider. Then, in §4.3 and §4.4, we study how our method performs on the EC, GO, and Pfam classification tasks. We focus on comparing with methods from prior work, and on comparing our method's performance with and without conditioning on exemplar sequences. Finally, in §4.5, we report additional analysis and ablations. Additional details and results can be found in Appendix B and C.

4.1 Tasks and Datasets

We consider several EC number, GO term, and Pfam prediction tasks summarized in Table 1. Some additional details are provided in Appendix B.

Table 1: **Dataset Statistics**. We consider prediction of EC numbers, GO terms, and Pfam families across several settings proposed in prior work. We report the number of unique classes among the training sequences, and the average number of classes per sequence.

Name	Labels	Training Sequences	Classes	Avg. # Classes Per Seq.
Random EC	EC	438,522	4,862	1.89
Random GO	GO	438,522	31,365	45.49
Clustered EC	EC	182,965	3,411	1.90
Clustered GO	GO	182,965	26,538	45.57
NEW-392	EC	227,362	5,242	1.06
Price-149	EC	227,362	5,242	1.06
PDB EC	EC	15,551	538	1.67
Clustered Pfam	Pfam	1,296,280	17,929	1

For EC and GO prediction, we adopt the random and clustered splits from Sanderson et al. [59]. These splits consist of proteins and their corresponding annotations from Swiss-Prot, the manually curated portion of UniProt [14]. For the random split, approximately 80% of proteins were reserved for training, with 10% each assigned to the development and test sets. For the more challenging clustered split, proteins were divided evenly into train, development, and test sets based on UniRef50 clusters [67], such that proteins in the development and test sets have lower sequence similarity with those in the training set than a random split.

To better compare our results on EC prediction with prior work, we also consider several other evaluations. First, we evaluate EC prediction on the setting proposed in Yu et al. [80]. Similarly to Sanderson et al. [59], this setting uses sequences from Swiss-Prot for training. There are two evaluation sets, NEW-392 and Price-149. While the training data contains sequences added to Swiss-Prot prior to April 2022, New-392 consists of 392 enzyme sequences added after this date, forming a temporal split. Price-149, originally curated by Sanderson et al. [59], consists of 149 sequences with EC numbers determined experimentally by Price et al. [48]. These sequences were inconsistently labeled by automated annotation methods, indicating a challenging setting. Second, we also adopt the PDB-based dataset proposed by Gligorijević et al. [22]. In contrast to the other EC splits based on sequences from Swiss-Prot, this dataset focuses on proteins with experimentally determined structures in PDB [8]. Therefore, the training set is considerably smaller. Notably, our method does not require structural information, but we nevertheless evaluate on this setting to compare with prior work.

Finally, we also evaluate on the Pfam seed dataset [20] where the goal is to map each protein sequence to one of 17,929 families. We use the clustered split as defined by Bileschi et al. [9] where sequences in the development and test sets have less than 25% sequence identity to the training set.

4.2 Retrievers and Candidate Labels

We considered two different approaches to implement a retriever and candidate label generator, as described in §3. We offer a summary here and more detail in Appendix B.5.

BLAST We used BLAST [3] as the retriever for the EC and GO tasks, as it has shown strong performance and is computationally feasible to run on these datasets. Specifically, we run `blastp` to retrieve up to the 100 most similar sequences in the training set. We determine the set of candidate labels by taking the union of the labels for all retrieved sequences. Similarly, positive and negative exemplars are selected from the retrieved sequences, ranked by alignment score.

Per Class Retrieval We adopted a different approach for the Pfam task. Unlike for the EC and GO tasks, BLAST does not achieve as strong performance on the Pfam clustered split. Moreover, the Pfam dataset is considerably larger than the other datasets (see Table 1). Thus running BLAST for all examples in the training set can take considerable time. Therefore, we implemented an alternative retrieval system that can be more easily parallelized and customized than BLAST. For every sequence, we select a set of exemplars for each class. For efficiency, we randomly select up to a maximum number of sequences per class, and then rank these sequences according to a local alignment score that is similar to the one computed by BLAST. Based on analysis of the effect of restricting the number of classes (see Appendix C.4.1), we opted to not use a candidate label generator and consider every class for every sequence. We also evaluated this strategy for EC prediction although it did not perform as well as using BLAST as described above (see Appendix C.3.2).

4.3 EC and GO Main Results

Here we report results on the EC and GO tasks. We used the BLAST retriever (§4.2) and up to 2 positive and 2 negative exemplars. See Appendix B for further details and Appendix C.1 for the statistical significance of key comparisons (tested differences are significant with p-values < 0.01).

We compare ProtEx (with and without exemplars) with the strongest results from prior work. For all experiments, we also report results for BLAST, following the setting of Sanderson et al. [59], which imputes the labels from the most similar sequence returned by BLAST if the score is above a bit score threshold.

Table 2: F1 scores for EC and GO prediction for random and clustered splits.

	EC		GO	
Method	Random	Clustered	Random	Clustered
ProteInfer	0.977	0.914	0.885	0.782
ProteInfer (ensemble)	0.981	0.930	0.899	0.811
BLAST	0.984	0.950	0.902	0.824
ProtEx	**0.987**	**0.958**	**0.917**	**0.854**
ProtEx (no exemplars)	0.977	0.944	0.786	0.754

First, we report results in Table 2 for the random and clustered EC and GO splits proposed by Sanderson et al. [59]. We follow Sanderson et al. [59] and report the maximum micro-averaged F1 score. We compare with a single-model and ensembled versions of ProteInfer [59], a CNN-based model. Our BLAST result reproduces that of Sanderson et al. [59], which outperforms ProteInfer. ProtEx improves over BLAST, with performance dropping considerably if exemplars are not included in the input during fine-tuning and inference. The precision and recall curves of ProtEx relative to BLAST are shown in Figure 3, demonstrating improvements in precision at all recall values.

Second, we report in Table 3 results on the NEW-392 and Price-149 evaluations proposed by Yu et al. [80], and compare with their proposed method CLEAN. As there is no development set provided in

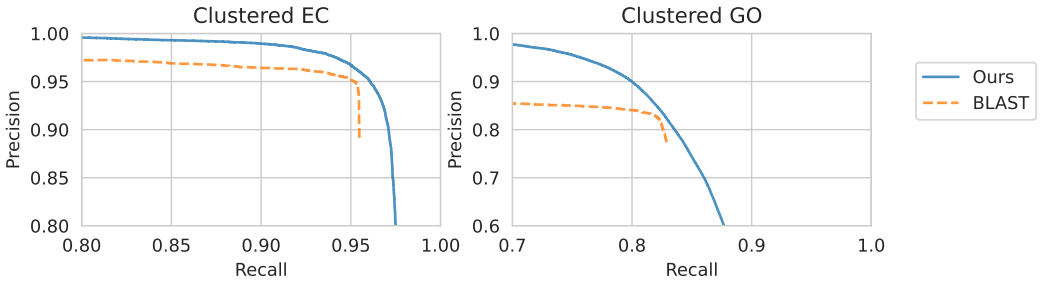

Figure 3: **Precision and Recall Curves**. ProtEx has improved precision at all recall values relative to BLAST on the clustered EC and GO tasks.

this setting, we used the hyperparameters determined for the clustered EC task, and compare results based on the weighted AUC metric reported by Yu et al. [80], which does not require selecting a score threshold. We see that ProtEx improves over both CLEAN and BLAST. Again, ablating exemplars leads to a drop in performance. Additionally, we also report F1 results in Appendix C.3.3.

Finally, we also report results on the PDB-based split proposed by Gligorijević et al. [22] in Table 4. Following prior work, we report the maximum protein-centric F1 score. Table 4 includes results for the strongest methods from prior work. We report ESM-GearNet results from Zhang et al. [83], ESM-2 [39] and PST results from Chen et al. [12], and PromptProtein results from Wang et al. [74]. Perhaps surprisingly, we see that BLAST with an alignment score threshold achieves a very competitive result on this setting. Regardless, ProtEx outperforms BLAST and other previously reported results on this setting. See Appendix C.3.4 for results stratified by similarity between test set and training set sequences, and for additional comparisons with prior work.

Table 3: Weighted AUC for EC prediction for NEW-392 and Price-149 evaluation sets.

Method	NEW-392	Price-149
CLEAN	0.740	0.733
BLAST	0.788	0.691
ProtEx	**0.932**	**0.842**
ProtEx (no exemplars)	0.926	0.839

Table 4: Protein-centric F1 for EC prediction on PDB-based split.

Method	PDB EC
ESM-GearNet	0.890
PromptProtein	0.888
ESM-2	0.892
PST	0.899
BLAST	0.900
ProtEx	**0.909**
ProtEx(no exemplars)	0.831

4.4 Pfam Main Results

For the Pfam dataset, we use the per-class retrieval approach discussed in § 4.2. We use 4 positive exemplars and zero negative exemplars, since we found in our early experiments that additional positive exemplars added more benefit. The results were computed considering all possible candidate labels for each example in the evaluation set. See Appendix C.4.1 for further details.

We compare ProtEx with two strong homology-based approaches (BLAST and Top pick HMM, as described in Dohan et al. [16]), ProtENN [9] a convolutional neural network ensemble, and ProtNLM [21] and ProtTNN [16], which are pretrained Transformer models. For the no exemplar ablation, we found that predicting a binary label without exemplars generalizes poorly when the number of classes is large, and so instead finetune our pretrained checkpoint to predict the class label as a string given the sequence, which we found performed better. See Appendix § C.4.2 for further discussion.

Results are shown in Table 5. We report family accuracy, lifted clan accuracy that groups families into higher level clans [16], and the average per-family accuracy, which gives equal weight to all classes, including rare classes. Our approach achieves state-of-the-art performance by a considerable margin. We further study the stratified performance of our method. As shown in Figure 4, while other

Table 5: Results on the Pfam clustered split. Sequences in the test set have less than 25% sequence identity to the training set.

Method	Family Accuracy	Lifted Clan Accuracy	Avg. Per-Family Accuracy
Top pick HMM	81.9	88.1	82.9
BLAST	64.1	70.1	63.7
ProtENN	87.8	89.0	80.4
ProtNLM	87.4	90.7	80.6
ProtTNN	88.4	90.5	83.4
ProtTNN (ensemble)	89.7	91.7	85.0
ProtEx	**92.6**	**93.3**	**91.7**
ProtEx (no exemplars)	76.3	80.2	65.7

methods typically show lower performance for examples with rare labels, ProtEx demonstrates relatively constant performance across training set family sizes, showcasing large improvements for examples belonging to rare families. See Appendix C.4.3 for further stratified performance analysis, showing that our method performs well across sequences that have low similarity with the closest sequence in the training data, and that similar trends are observed at the lifted clan level.

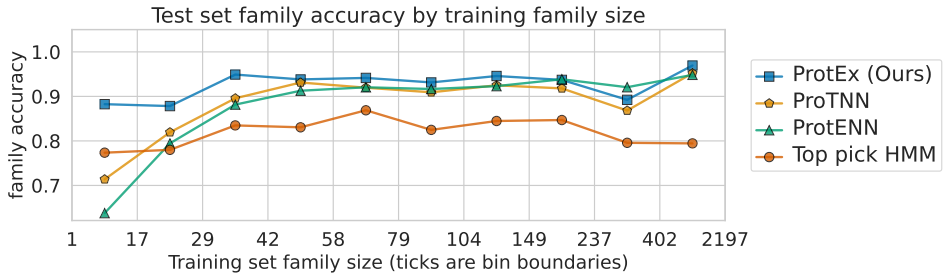

Figure 4: **Pfam stratified family accuracy**. ProtEx shows considerably improved performance for sequences belonging to rare labels.

4.5 Analysis and Ablations

Table 6: **Generalization to Unseen Labels.** Max F1 on clustered EC and GO when a randomly selected subset of labels are not included during fine-tuning.

	EC		GO	
Method	Seen	Unseen	Seen	Unseen
BLAST	0.953	0.964	0.826	0.816
ProtEx	**0.960**	**0.970**	**0.849**	**0.839**

Table 7: **Pre-training Ablations.** We report F1 on Clustered EC development split for different pre-training strategies.

Pre-training	F1
Sequence pair w/ score	**0.958**
Sequence pair	0.956
Single sequence	0.952
No pre-training	0.912

Generalization to New Labels We tested the ability of ProtEx to make predictions for new classes not seen during fine-tuning. On the EC and GO clustered splits we randomly removed 10% of classes during fine-tuning, while retaining the ability to retrieve sequences annotated with these classes at inference time. As shown in Table 6, ProtEx performs comparably or better than BLAST even on classes it has not seen during training. Relatedly, we also found that there is only a minimal performance decrease when candidate labels are not included in the model input (see Appendix C.3.1), further indicating that the model is conditioning its predictions on the exemplar sequences as opposed to directly representing the sequence to class relationship in the model parameters.

Pre-training Analysis We show results for ablating the key elements of our pre-training recipe in Table 7. Notably, there is a drop in performance when not including the similarity score as a prediction

target, and a further drop when pre-training with only a single sequence as input, as commonly done in prior work. This indicates that our pre-training task is useful for retrieval-augmented models that are fine-tuned to make comparisons across multiple sequences. See Appendix B.4 for further details.

Model Architectures and Scaling In Appendix C.2 we compare small and base sized models, finding that there is a modest benefit to increasing model size from small to base. We also compare the Fusion-in-Decoder (FiD) approach introduced by Izacard and Grave [26] with a standard Transformer, finding that this may be viable path towards scaling to more exemplars, but that there is a performance drop, indicating that cross-attention between exemplar sequences is beneficial.

Exemplar Distribution and Sampling As described in §3, we study different sampling strategies to select exemplars during training. This adds diversity to the training data, and can also help align the distribution of similarities between query and exemplar sequences seen during training with those seen during inference, which is especially useful for non-random splits. We highlight this capability on the Pfam task, which features the largest distributional shift between training and inference due to the split restricting inference sequences to have <25% sequence similarity to the training set. Figure 5 shows the corresponding distribution of similarities between query and exemplar sequences and its impact on out-of-distribution generalization as measured by family accuracy on development set.

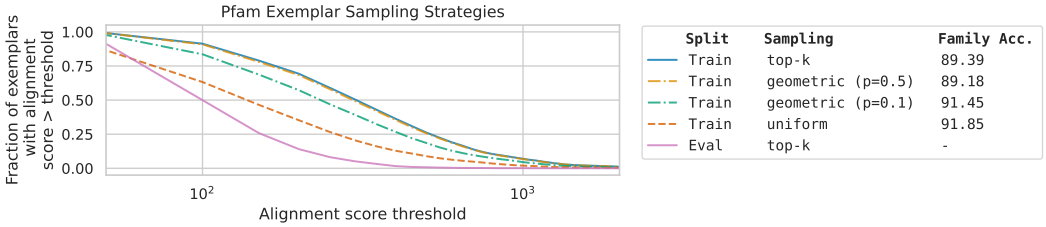

Figure 5: **Effect of exemplar sampling strategy on Pfam**. We visualize how different sampling strategies can mitigate the distribution shift between training and evaluation. Uniform sampling leads to the training distribution most similar to the evaluation distribution, and also leads to the highest family accuracy on the development set.

5 Limitations and Discussion

Computational Requirements While we use a model size and number of pre-training steps comparable to prior work, the cost of inference with our method is potentially larger due to encoding multiple protein sequences and making independent predictions for each class. To mitigate these factors, we have considered the FiD architecture (Appendix C.2) and a candidate label generator (§3), which offer a path towards more efficient inference. Regardless, through our experiments, we have demonstrated the feasibility of making predictions using ProtEx at a relevant scale.

Scope and Future Work In this work we focused on training and inference procedures that can effectively condition predictions on retrieved exemplars, using a general-purpose Transformer architecture and standard methods for retrieval (e.g. BLAST). Going forward, our approach could potentially be further improved using enhanced similarity search techniques such as those based on protein structure [84, 24, 70] and more specialized architectures. Finally, we focused on predicting EC, GO, and Pfam labels. Other tasks such as fitness prediction [55] or generating free-text descriptions of protein function [21, 1] could be of interest for future work.

Broader Impact and Ethical Considerations Our method enables more accurate and robust prediction of protein functional annotations. Any computationally derived annotation should be verified by wet lab experiments where possible, especially for critical applications. Our method extends a long history of prior work that develops such tools and the community's safeguards for how to use such proteins in an ethical manner applies here as well.

6 Conclusion

We proposed ProtEx, a semiparametric approach that combines aspects of homology-based similarity search with pre-trained neural models. ProtEx achieves state-of-the-art results on EC, GO, and Pfam classification tasks. Our work highlights the potential of retrieval-augmented methods for improving the accuracy and robustness of protein function prediction.

Acknowledgments

We thank Chris Alberti for feedback on a draft of this paper. We also thank Zuobai Zhang for assistance with evaluations.

References

[1] Hadi Abdine, Michail Chatzianastasis, Costas Bouyioukos, and Michalis Vazirgiannis. Prot2text: Multimodal protein's function generation with GNNs and transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 10757–10765, 2024.

[2] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

[3] Stephen F Altschul, Thomas L Madden, Alejandro A Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997.

[4] Christian B Anfinsen. Principles that govern the folding of protein chains. *Science*, 181(4096): 223–230, 1973.

[5] Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25–29, 2000.

[6] Minkyung Baek, Frank DiMaio, Ivan Anishchenko, Justas Dauparas, Sergey Ovchinnikov, Gyu Rie Lee, Jue Wang, Qian Cong, Lisa N Kinch, R Dustin Schaeffer, et al. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557):871–876, 2021.

[7] Alex Bateman, Lachlan Coin, Richard Durbin, Robert D Finn, Volker Hollich, Sam Griffiths-Jones, Ajay Khanna, Mhairi Marshall, Simon Moxon, Erik LL Sonnhammer, et al. The Pfam protein families database. *Nucleic acids research*, 32(suppl_1):D138–D141, 2004.

[8] Helen M Berman, John Westbrook, Zukang Feng, Gary Gilliland, Talapady N Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne. The protein data bank. *Nucleic acids research*, 28(1):235–242, 2000.

[9] 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.

[10] Yue Cao and Yang Shen. TALE: Transformer-based protein function annotation with joint sequence–label embedding. *Bioinformatics*, 37(18):2825–2833, 2021.

[11] Ilias Chalkidis and Yova Kementchedjhieva. Retrieval-augmented multi-label text classification. *arXiv preprint:2305.13058*, 2023.

[12] Dexiong Chen, Philip Hartout, Paolo Pellizzoni, Carlos Oliver, and Karsten Borgwardt. Endowing protein language models with structural knowledge. *arXiv preprint arXiv:2401.14819*, 2024.

[13] Yanda Chen, Ruiqi Zhong, Sheng Zha, George Karypis, and He He. Meta-learning via language model in-context tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 719–730, 2022.

[14] UniProt Consortium. Uniprot: a hub for protein information. *Nucleic Acids Res*, 43(D1): D204–D212, 2015.

[15] Andrew M Dickson and Mohammad RK Mofrad. Fine-tuning protein embeddings for generalizable annotation propagation. *bioRxiv*, pages 2023–06, 2023.

[16] David Dohan, Andreea Gane, Maxwell L Bileschi, David Belanger, and Lucy Colwell. Improving protein function annotation via unsupervised pre-training: Robustness, efficiency, and insights. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2782–2791, 2021.

[17] Janani Durairaj, Andrew M Waterhouse, Toomas Mets, Tetiana Brodiazhenko, Minhal Abdullah, Gabriel Studer, Gerardo Tauriello, Mehmet Akdel, Antonina Andreeva, Alex Bateman, et al. Uncovering new families and folds in the natural protein universe. *Nature*, 622(7983):646–653, 2023.

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

[19] Hehe Fan, Zhangyang Wang, Yi Yang, and Mohan Kankanhalli. Continuous-discrete convolution for geometry-sequence modeling in proteins. In *The Eleventh International Conference on Learning Representations*, 2022.

[20] Robert D Finn, Alex Bateman, Jody Clements, Penelope Coggill, Ruth Y Eberhardt, Sean R Eddy, Andreas Heger, Kirstie Hetherington, Liisa Holm, Jaina Mistry, et al. Pfam: the protein families database. *Nucleic acids research*, 42(D1):D222–D230, 2014.

[21] A Gane, ML Bileschi, D Dohan, E Speretta, A Héliou, L Meng-Papaxanthos, H Zellner, E Brevdo, A Parikh, MJ Martin, et al. ProtNLM: model-based natural language protein annotation. *Preprint*, 2022.

[22] Vladimir Gligorijević, P Douglas Renfrew, Tomasz Kosciolek, Julia Koehler Leman, Daniel Berenberg, Tommi Vatanen, Chris Chandler, Bryn C Taylor, Ian M Fisk, Hera Vlamakis, et al. Structure-based protein function prediction using graph convolutional networks. *Nature communications*, 12(1):3168, 2021.

[23] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR, 2020.

[24] 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.

[25] Steven Henikoff and Jorja G Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919, 1992.

[26] Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. In *EACL 2021-16th Conference of the European Chapter of the Association for Computational Linguistics*, pages 874–880. Association for Computational Linguistics, 2021.

[27] Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251): 1–43, 2023.

[28] L Steven Johnson, Sean R Eddy, and Elon Portugaly. Hidden markov model speed heuristic and iterative HMM search procedure. *BMC bioinformatics*, 11:1–8, 2010.

[29] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.

[30] Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. Large language models struggle to learn long-tail knowledge. In *International Conference on Machine Learning*, pages 15696–15707. PMLR, 2023.

[31] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, 2020.

[32] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. In *ICLR*, 2020.

[33] Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Nearest neighbor machine translation. In *International Conference on Learning Representations (ICLR)*, 2021.

[34] Janez Konc, Milan Hodošček, Mitja Ogrizek, Joanna Trykowska Konc, and Dušanka Janežič. Structure-based function prediction of uncharacterized protein using binding sites comparison. *PLoS computational biology*, 9(11):e1003341, 2013.

[35] Maxat Kulmanov, Mohammed Asif Khan, and Robert Hoehndorf. DeepGO: predicting protein functions from sequence and interactions using a deep ontology-aware classifier. *Bioinformatics*, 34(4):660–668, 2018.

[36] Boqiao Lai and Jinbo Xu. Accurate protein function prediction via graph attention networks with predicted structure information. *Briefings in Bioinformatics*, 23(1):bbab502, 2022.

[37] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[38] Patrick Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Küttler, Aleksandra Piktus, Pontus Stenetorp, and Sebastian Riedel. PAQ: 65 million probably-asked questions and what you can do with them. *Transactions of the Association for Computational Linguistics*, 9: 1098–1115, 2021.

[39] 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.

[40] Yaniv Loewenstein, Domenico Raimondo, Oliver C Redfern, James Watson, Dmitrij Frishman, Michal Linial, Christine Orengo, Janet Thornton, and Anna Tramontano. Protein function annotation by homology-based inference. *Genome biology*, 10:1–8, 2009.

[41] Alexander Long, Wei Yin, Thalaiyasingam Ajanthan, Vu Nguyen, Pulak Purkait, Ravi Garg, Chunhua Shen, and Anton van den Hengel. Retrieval augmented classification for long-tail visual recognition. In *CVPR 2022*, 2022.

[42] Debora S Marks, Lucy J Colwell, Robert Sheridan, Thomas A Hopf, Andrea Pagnani, Riccardo Zecchina, and Chris Sander. Protein 3D structure computed from evolutionary sequence variation. *PloS one*, 6(12):e28766, 2011.

[43] Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. MetaICL: Learning to learn in context. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2791–2809, 2022.

[44] 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.

[45] Sara Mostafavi, Debajyoti Ray, David Warde-Farley, Chris Grouios, and Quaid Morris. GeneMANIA: a real-time multiple association network integration algorithm for predicting gene function. *Genome biology*, 9:1–15, 2008.

[46] Pascal Notin, Mafalda Dias, Jonathan Frazer, Javier Marchena Hurtado, Aidan N Gomez, Debora Marks, and Yarin Gal. Tranception: protein fitness prediction with autoregressive transformers and inference-time retrieval. In *International Conference on Machine Learning*, pages 16990–17017. PMLR, 2022.

[47] Panupong Pasupat, Yuan Zhang, and Kelvin Guu. Controllable semantic parsing via retrieval augmentation. *arXiv preprint arXiv:2110.08458*, 2021.

[48] Morgan N Price, Kelly M Wetmore, R Jordan Waters, Mark Callaghan, Jayashree Ray, Hualan Liu, Jennifer V Kuehl, Ryan A Melnyk, Jacob S Lamson, Yumi Suh, et al. Mutant phenotypes for thousands of bacterial genes of unknown function. *Nature*, 557(7706):503–509, 2018.

[49] Morgan N Price, Adam M Deutschbauer, and Adam P Arkin. Filling gaps in bacterial catabolic pathways with computation and high-throughput genetics. *PLoS genetics*, 18(4):e1010156, 2022.

[50] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[51] Rita Ramos, Bruno Martins, Desmond Elliott, and Yova Kementchedjhieva. SmallCap: Lightweight image captioning prompted with retrieval augmentation. *CVPR*, 2023.

[52] Roshan M Rao, Jason Liu, Robert Verkuil, Joshua Meier, John Canny, Pieter Abbeel, Tom Sercu, and Alexander Rives. MSA transformer. In *International Conference on Machine Learning*, pages 8844–8856. PMLR, 2021.

[53] 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.

[54] Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C Lawrence Zitnick, Jerry Ma, et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15):e2016239118, 2021.

[55] Philip A Romero, Andreas Krause, and Frances H Arnold. Navigating the protein fitness landscape with gaussian processes. *Proceedings of the National Academy of Sciences*, 110(3): E193–E201, 2013.

[56] Ambrish Roy, Jianyi Yang, and Yang Zhang. COFACTOR: an accurate comparative algorithm for structure-based protein function annotation. *Nucleic acids research*, 40(W1):W471–W477, 2012.

[57] Kiersten M Ruff and Rohit V Pappu. AlphaFold and implications for intrinsically disordered proteins. *Journal of molecular biology*, 433(20):167208, 2021.

[58] Jae Yong Ryu, Hyun Uk Kim, and Sang Yup Lee. Deep learning enables high-quality and high-throughput prediction of enzyme commission numbers. *Proceedings of the National Academy of Sciences*, 116(28):13996–14001, 2019.

[59] Theo Sanderson, Maxwell L Bileschi, David Belanger, and Lucy J Colwell. ProteInfer, deep neural networks for protein functional inference. *Elife*, 12:e80942, 2023.

[60] Thibault Sellam, Dipanjan Das, and Ankur Parikh. Bleurt: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, 2020.

[61] Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR, 2018.

[62] Weijia Shi, Sewon Min, Maria Lomeli, Chunting Zhou, Margaret Li, Xi Victoria Lin, Noah A Smith, Luke Zettlemoyer, Wen-tau Yih, and Mike Lewis. In-context pretraining: Language modeling beyond document boundaries. In *The Twelfth International Conference on Learning Representations*, 2024.

[63] Artem Sokolov and Asa Ben-Hur. Hierarchical classification of gene ontology terms using the GOstruct method. *Journal of bioinformatics and computational biology*, 8(02):357–376, 2010.

[64] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

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

[66] Nils Strodthoff, Patrick Wagner, Markus Wenzel, and Wojciech Samek. UDSMProt: universal deep sequence models for protein classification. *Bioinformatics*, 36(8):2401–2409, 2020.

[67] 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.

[68] Keith Tipton and Sinéad Boyce. History of the enzyme nomenclature system. *Bioinformatics*, 16(1):34–40, 2000.

[69] Uniprot. UniProt: the universal protein knowledgebase in 2023. *Nucleic acids research*, 51 (D1):D523–D531, 2023.

[70] Michel Van Kempen, Stephanie S Kim, Charlotte Tumescheit, Milot Mirdita, Jeongjae Lee, Cameron LM Gilchrist, Johannes Söding, and Martin Steinegger. Fast and accurate protein structure search with foldseek. *Nature Biotechnology*, 42(2):243–246, 2024.

[71] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[72] Amelia Villegas-Morcillo, Stavros Makrodimitris, Roeland CHJ van Ham, Angel M Gomez, Victoria Sanchez, and Marcel JT Reinders. Unsupervised protein embeddings outperform hand-crafted sequence and structure features at predicting molecular function. *Bioinformatics*, 37(2):162–170, 2021.

[73] Shuohang Wang, Yichong Xu, Yuwei Fang, Yang Liu, Siqi Sun, Ruochen Xu, Chenguang Zhu, and Michael Zeng. Training data is more valuable than you think: A simple and effective method by retrieving from training data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3170–3179, 2022.

[74] Zeyuan Wang, Qiang Zhang, HU Shuang-Wei, Haoran Yu, Xurui Jin, Zhichen Gong, and Huajun Chen. Multi-level protein structure pre-training via prompt learning. In *The Eleventh International Conference on Learning Representations*, 2023.

[75] Zichen Wang, Steven A Combs, Ryan Brand, Miguel Romero Calvo, Panpan Xu, George Price, Nataliya Golovach, Emmanuel O Salawu, Colby J Wise, Sri Priya Ponnapalli, et al. Lm-gvp: an extensible sequence and structure informed deep learning framework for protein property prediction. *Scientific reports*, 12(1):6832, 2022.

[76] Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. Byt5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306, 2022.

[77] Alexander Yeh. More accurate tests for the statistical significance of result differences. In *COLING 2000 Volume 2: The 18th International Conference on Computational Linguistics*, 2000.

[78] Ronghui You, Shuwei Yao, Yi Xiong, Xiaodi Huang, Fengzhu Sun, Hiroshi Mamitsuka, and Shanfeng Zhu. NetGO: improving large-scale protein function prediction with massive network information. *Nucleic acids research*, 47(W1):W379–W387, 2019.

[79] Guoxin Yu, Lemao Liu, Haiyun Jiang, Shuming Shi, and Xiang Ao. Retrieval-augmented few-shot text classification. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6721–6735, 2023.

[80] Tianhao Yu, Haiyang Cui, Jianan Canal Li, Yunan Luo, Guangde Jiang, and Huimin Zhao. Enzyme function prediction using contrastive learning. *Science*, 379(6639):1358–1363, 2023.

[81] Qianmu Yuan, Junjie Xie, Jiancong Xie, Huiying Zhao, and Yuedong Yang. Fast and accurate protein function prediction from sequence through pretrained language model and homology-based label diffusion. *Briefings in bioinformatics*, 24(3):bbad117, 2023.

[82] Fuhao Zhang, Hong Song, Min Zeng, Yaohang Li, Lukasz Kurgan, and Min Li. DeepFunc: a deep learning framework for accurate prediction of protein functions from protein sequences and interactions. *Proteomics*, 19(12):1900019, 2019.

[83] Z Zhang, C Wang, M Xu, V Chenthamarakshan, AC Lozano, P Das, and J Tang. A systematic study of joint representation learning on protein sequences and structures. *Preprint at http://arxiv.org/abs/2303.06275*, 2023.

[84] Zuobai Zhang, Minghao Xu, Arian Jamasb, Vijil Chenthamarakshan, Aurelie Lozano, Payel Das, and Jian Tang. Protein representation learning by geometric structure pretraining. *arXiv preprint arXiv:2203.06125*, 2022.

[85] Zuobai Zhang, Jiarui Lu, Vijil Chenthamarakshan, Aurélie Lozano, Payel Das, and Jian Tang. ProtIR: Iterative refinement between retrievers and predictors for protein function annotation. *arXiv preprint arXiv:2402.07955*, 2024.

A Additional Method Details

A.1 Model Interface

We use the character-based vocabulary of Xue et al. [76], which ensures that amino acid sequences are tokenized into their individual amino acid residues. We represent labels as short character sequences such as `EC:1.2.3.4`. We use single characters to indicate the start and end of amino acid sequences, and to indicate whether an exemplar sequence is positive or negative.

The training target is a single character sequence, `p` or `n`, for positive and negative examples, respectively. At inference time, we determine the score based on the probability assigned to the single character sequence `p`.

A.2 Pretraining Details

We mask the sequences by randomly replacing approximately 10% of residues with a placeholder index. The prediction target during pre-training consists of predicting the masked residues as well as the similarity between the two sequences. As a simple and easy to compute measure of sequence similarity, we use Levenshtein distance normalized by the sequence length, bucketed to the nearest multiple of 5. The Levenshtein-based similarity objective encourages the model to approximately align and compare two sequences. Since we have a deterministic algorithm to compute this quantity, we can apply this objective over unlabeled sequence pairs. We expect the model to implicitly learn a more task-specific notion of similarity during fine-tuning. Sequence pairs were sampled to approximate a uniform distribution over sequence similarity buckets.

The process is loosely analogous to some methods explored in NLP. For example, Sellam et al. [60] pre-trained models to compute BLEU scores over pairs of strings. BLEU is a deterministic measure of string similarity. Models were then fine-tuned on human labeled data to learn a more task-specific notion of similarity. Pre-training on a context that includes *related* sequences is perhaps also analogous to the in-context pre-training method proposed by Shi et al. [62], which includes a language modeling objective over related documents, showing this is useful relative to randomly selected documents, for various downstream tasks.

B Additional Dataset and Experiment Details

B.1 Dataset Details

EC Labels As Table 1 shows, the number of EC classes considered varies across tasks. This is partially due to differences in which sequences, and therefore which EC labels, are included in the training set. However, different tasks also consider different tiers of the EC hierarchy. The Swiss-Prot based random and clustered splits consider labels from all 4 levels of the EC hierarchy, the PDB EC tasks considers only levels 3 and 4, and NEW-392 and Price-149 evaluations only consider level 4. Also notably, the Price-149 labels were originally derived from Price et al. [48]. However, more recent work [49] has revisited the functional annotations of some of these sequences, and should be considered for future work with this evaluation.

Dataset Licenses The EC and GO tasks are adapted from Swiss-Prot, which is the human curated portion of UniProt that is released under CC BY 4.0. The PDB EC split is also available under CC BY 4.0. The Pfam task is derived from Pfam[2] and is released under the CC0 1.0 license. For pretraining data we used Uniref90 [67], which is derived from UniProt [14] that is released under CC BY 4.0.

B.2 Hyperparameters

Pre-training We pre-trained models for 1M steps using a learning rate of 1e-3 and a batch size of 256 tokens using Adafactor [61].

EC and GO Fine-tuning We selected hyperparameters based on development set performance, focusing on the Swiss-Prot clustered splits. For all ProtEx models with exemplars we use a learning

[2] https://interpro-documentation.readthedocs.io/en/latest/pfam.html

rate of 1e-3 with Adafactor with dropout regularization [64] set to 0.1. For the Swiss-Prot based splits, we trained models for 50,000 steps. For the smaller PDB based split, we trained models for 8,000 steps. Models without exemplars were trained longer, as these models took longer to reach a stable development accuracy. We trained these models for 100,000 steps for the Swiss-Prot splits and 40,000 steps for the PDB split. We used a batch size of 256 for all experiments. For the random and clustered Swiss-Prot splits we used a maximum input sequence length of 6784 tokens, which led to some truncation of exemplars in about 1% of examples during training and inference. The other tasks did not lead to inputs that exceeded this length.

Pfam Fine-tuning For all ProtEx models with exemplars we use a learning rate of 2e-4, batch size of 128, a maximum input length of 6,528, and dropout set to 0.1. We finetune for 200,000 steps with Adafactor, and pick the best model based on development accuracy.

The configuration for the no exemplar ablation is similar except we use a higher learning rate of 1e-3 which we found worked better in practice and a beam size of 8 since we are treating the label as a string that can be tokenized into multiple tokens. Since there are no exemplars the maximum input length could be shortened to 2,688.

B.3 Compute

For training and inference we used Google Cloud TPUs (v3 and v5e) in configurations of up to 128 chips. Pre-training the Base model to 1M steps took approximately 7 days on 64 TPU v3 chips. Fine-tuning the Base model took approximately 3 hours per 10K steps of fine-tuning.

B.4 Pre-training Ablations Details

Here we provide additional details about the pre-training ablations shown in Table 7. As pre-training is computationally expensive and our Small and Base models perform comparably (see C.2), we used Small models for these comparisons. Additionally, we observed that fine-tuning performance was comparable when tuning from a checkpoint that had been pre-trained for 500K or 1M steps, indicating that most of the advantage of pre-training is accrued in the first 500K steps. Therefore, we compared models pre-training for 500K steps. Finally, given the full development set is quite large (approximately 180,000 examples), we perform this ablation on a random 10% subset.

B.5 Retriever Details

BLAST We use `ncbi-blast-2.14.1+`. We run `makeblastdb` with `-dbtype prot`, and then query the database using `blastp` with default arguments and `-max_target_seqs 100`.

We select exemplars during training using geometric sampling with $p = 0.5$.

Per Class Retrieval For flexibility and ease of parallelization, we use Biopython[3] `Align.PairwiseAligner`. We set `mode = local`, `extend_gap_score` = -1.0, `open_gap_score` = -11.0, and `substitution_matrix = BLOSUM62`.

For development and inference, for each query we find the closest 4 exemplars from each class using the pairwise aligner above. For training given the size of the dataset, for each query we sample up to 100 exemplar candidates per class and select 4 exemplars from this set using uniform sampling as detailed in § 3, which we found performs the best (Table 5).

B.6 Negative Example Sampling

As discussed in §3, we sometimes use sampling of negative examples to avoid class imbalance during training. When using the BLAST retriever this is not necessary because the set of candidate labels consists of a reasonable balance of positive and negative labels. However, when using the Per Class Retrieval method, naively generating a training example for every label would lead to an imbalance. Therefore, for each sequence, we generate a negative example for the label with the highest similarity score, and also randomly sample another negative label.

[3] https://biopython.org/

C Additional Results and Analysis

C.1 Statistical Significance and Variance

For the results reported in Table 2 and Table 4, we assessed the statistical significance of the difference in F1 score between ProtEx and BLAST, for cases where the difference was less than 0.01, using a permutation test [77]. In Table 8, we report the p-values and standard deviation of the sampled score differences under the null hypothesis that predictions from the two approaches are interchangeable, which was estimated using 100 sampled permutations of the predictions. We computed p-values using a t-test.

Table 8: Statistical significance of comparisons between ProtEx and BLAST.

Task	Metric	Null Stdev.	Observed Diff	p-value
Random EC	Micro F1	0.0002	0.003	1.32e-24
Clustered EC	Micro F1	0.0002	0.008	1.53e-59
PDB EC	Protein-centric F1	0.0031	0.009	6.17e-3

For the evaluation settings where the training and evaluation sets are small, we also computed the variance from different fine-tuning runs. The standard deviation across 3 different fine-tuning runs is shown in Table 9, which is in all cases small relative to the performance differences between ProtEx and prior work.

Table 9: Variance between fine-tuning runs.

Task	Metric	Fine-tuning Stdev.
PDB EC	Protein-centric F1	0.0006
NEW-392	Weighted AUC	0.0003
Price-149	Weighted AUC	0.0009

C.2 Model Architectures and Scaling

The computational cost of self-attention in a standard Transformer scales quadratically with input length. To more efficiently encode exemplars, we studied the Fusion-in-Decoder (FiD) approach introduced by Izacard and Grave [26]. We apply this approach to encode the query and each exemplar in a separate encoder, effectively masking attention between exemplars. Notably, this model variant consists of the same set of parameters and can be initialized from the same pre-trained checkpoint. We compare the performance of the standard and FiD architecture for Small (60M) models in Table 10.

The FiD architecture performs only slightly worse than the standard Transformer, indicating this may be one path towards more efficiently encoding a larger number of exemplars. On the other hand, the drop in performance suggests there is value in attention across exemplar sequences.

Table 10: **Model Architecture Comparisons.** We report F1 on Clustered EC and GO development splits for standard Transformer vs. Fusion-in-Decoder (FiD) for Small models

Architecture	EC	GO
Standard Transformer	**0.958**	**0.845**
FiD Transformer	0.953	0.842

We also compare Small vs. Base size models in Table 11. Given the full development set is quite large (approximately 180,000 examples), we performed both these ablations on a random 10% subset.

Table 11: **Model Size Comparisons.** We report F1 on Clustered EC and GO development splits for Small and Base sized Transformers.

Size	EC	GO
Base (220M)	**0.959**	**0.848**
Small (60M)	0.958	0.845

C.3 Additional Results on EC Prediction

C.3.1 Ablating Labels in Input

We evaluate how much the model's performance depends on being able to condition on the candidate label as input. Table 12 shows that the model achieves similar performance with and without the candidate label, indicating that the model is indeed conditioning on the exemplars. Given the full development set is quite large (approximately 180,000 examples), we performed this ablation on a random 10% subset.

Table 12: **Label Ablation.** Comparing whether we include the label being predicted in the input as shown in Figure 1, or not. Results are max F1 on the clustered EC and GO development set for Small models. Model performance is only slightly lower without access to the label.

Method	EC	GO
ProtEx-Small (with input label)	**0.958**	**0.845**
ProtEx-Small (without input label)	0.957	0.843

C.3.2 Comparing Retrieval Strategies

We compared the BLAST and per-class retrieval strategies for EC prediction on the clustered split in Table 13. Using BLAST to filter the number of classes gives considerably stronger performance on EC prediction, likely because the BLAST performance is quite high. As a result, we used the BLAST retrieval strategy for all the EC and GO results. As with Table 12, given the full development set is quite large (approximately 180,000 examples), we performed this ablation on a random 10% subset.

Table 13: **Retrieval Strategy Comparison**: Comparing the BLAST and per class retrieval approaches on EC prediction (clustered development split).

Method	EC
ProtEx (BLAST retrieval)	**0.959**
ProtEx (Per Class retrieval)	0.929

C.3.3 NEW-392 and Price-149 Results

The weighted AUC metric proposed by Yu et al. [80] averages F1 scores over classes based on their representation in the test set. Especially since the NEW-392 and Price-149 test sets only include a small subset of classes, this metric tends to emphasize higher recall and lower precision relative to more standard metrics such as micro-averaged F1. Therefore, we also report the maximum micro-averaged F1 scores for NEW-392 and Price-149 for ProtEx and BLAST in Table 14.

Table 14: Maximum Micro F1 scores for EC prediction for NEW-392 and Price-149 evaluation sets.

Method	NEW-392	Price-149
BLAST	0.593	0.391
ProtEx	**0.612**	**0.441**

C.3.4 PDB EC Stratified Results

In Table 15 we report results for the PDB EC task stratified by the similarity of test sequences to those in the training set, as reported by prior work.We also include additional results reported by prior work for DeepFRI [22], GearNet MVC [84], ESM-1b [54] (as reported by Zhang et al. [84]), LM-GVP [75], and MMSeqs [65], which was the top retriever-based result reported by Zhang et al. [85].

Table 15: Maximum protein-centric F1 for EC classification on PDB-based split.

	Sequence Similarity Cutoff		
Method	<30%	<50%	<95%
DeepFRI	0.470	0.545	0.631
LM-GVP	-	-	0.664
ESM-1b	0.737	0.797	0.864
GearNet MVC	0.744	0.808	0.874
MMseqs	0.781	0.833	0.887
ESM-GearNet	-	-	0.890
PromptProtein	0.765	0.823	0.888
ESM-2	-	-	0.892
PST	-	-	0.899
BLAST	0.801	0.848	0.900
ProtEx	**0.820**	**0.862**	**0.909**

C.4 Additional Results on Pfam

C.4.1 Analyzing Effect of Class Filtering

Unlike for the EC and GO tasks, in Pfam we do not use a candidate label generator and consider all potential classes for each query sequence. We made this decision based on the following analysis.

We used `PairwiseAligner` (as in Appendix B.5) to select the single closest exemplar per candidate label for a random subset of 1000 sequences in the development set which gives $|\mathcal{L}|$ exemplars for each sequence. We can then restrict the number of candidate labels to K by taking the corresponding classes for the closest K exemplars to the query sequence in this selected set. Table 16 shows the results for various values of K, showing that it is beneficial to consider a large number of classes.

Table 16: **Analysis of Class Filtering for Pfam**: Table showing how filtering by a homology based approach (`PairwiseAligner`) reduces the accuracy ceiling.

Number of Candidate Labels	Accuracy Ceiling
10	82.3
50	87.8
100	88.7
500	93.4
1000	95.3
2000	96.2
5000	98.2
17929	100

C.4.2 No Exemplar Ablation

We experiment with ablations for Pfam that remove exemplars. The first is to finetune the model following the procedure as our other results for ProtEx *i.e.* to generate per-class binary predictions, but with no exemplars. The second strategy is to finetune our pretrained checkpoint to directly predict the label string from the sequence.

As shown in Table 17, we find that the first approach performs poorly compared to the second. Upon further analysis, we believe the reason for this is the large number of classes (17,929) in Pfam. We hypothesize that, without exemplars, the model does not learn to effectively discriminate between the positive class and all competing classes when trained using binary supervision, which requires sampling of negative classes to avoid class imbalance.

Table 17: **No Exemplar Ablation Comparison**: Comparing different no exemplar approaches for Pfam seed.

Method	Family Accuracy (Dev)
No Exemplar Binary Prediction	40.3
No Exemplar Label String Prediction	74.7

C.4.3 Stratified Performance

We show family accuracy stratified by sequence similarity in Figure 6. This shows that our approach consistently performs well across sequences with low similarity to the closest sequence in the training data. We also show the stratified performance by lifted clan accuracy in Figure 7 and Figure 8 that shows similar trends to Figure 6 and Figure 4.

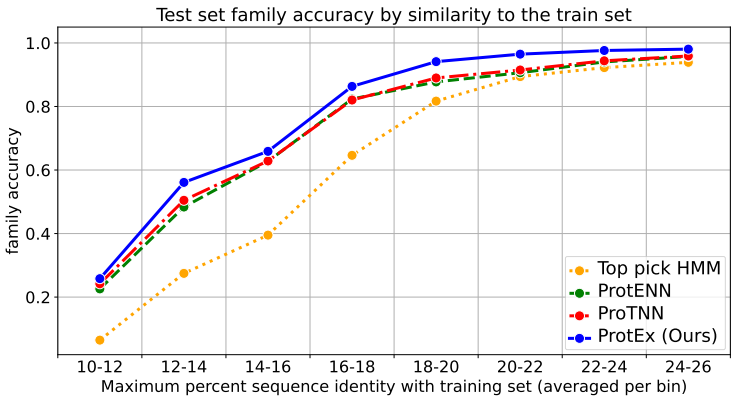

Figure 6: Pfam family accuracy stratified by sequence similarity.

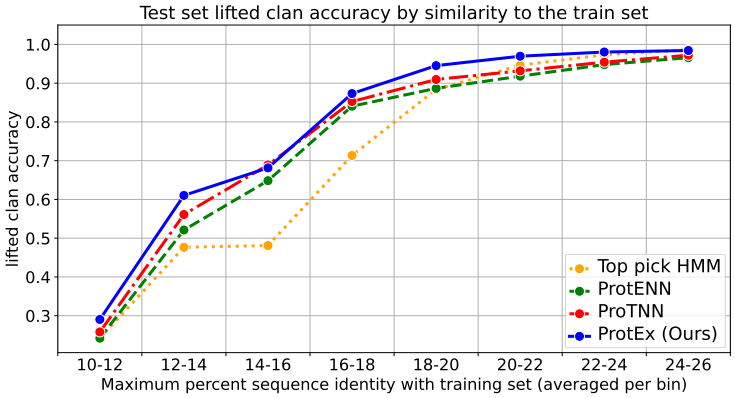

Figure 7: Pfam lifted clan accuracy stratified by sequence similarity.

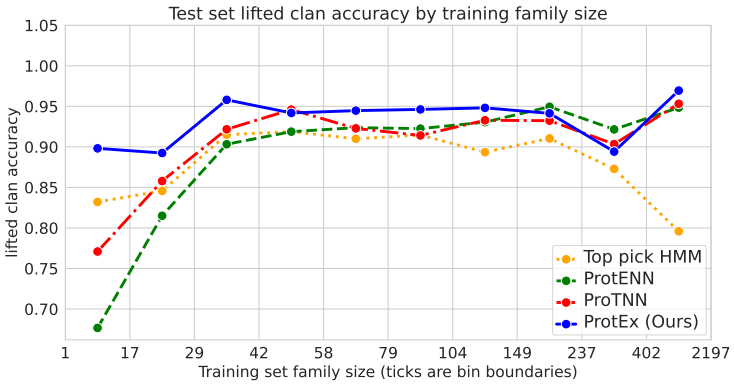

Figure 8: Pfam lifted clan accuracy stratified by number of training examples per class.