PROTRIEVER: END-TO-END DIFFERENTIABLE PROTEIN HOMOLOGY SEARCH FOR FITNESS PREDICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Retrieving homologous protein sequences is essential for a broad range of protein modeling tasks such as fitness prediction, protein design, structure modeling, and protein-protein interactions. Traditional workflows have relied on a two-step process: first retrieving homologs via Multiple Sequence Alignments (MSA), then training models on one or more of these alignments. However, MSA-based retrieval is computationally expensive, struggles with highly divergent sequences and complex insertions/deletions, and operates independently of downstream modeling. We introduce Protriever, an end-to-end differentiable framework that unifies retrieval and task modeling. Focusing on protein fitness prediction, we show that Protriever achieves performance on par with the most sensitive MSA-based tools while being orders of magnitude faster at retrieval, as it relies on efficient vector search. Protriever is both architecture- and task-agnostic, and can flexibly adapt to different retrieval strategies and protein databases at inference – offering a scalable alternative to alignment-centric approaches.

Proteins evolve under strict constraints that preserve their function, creating specific mutation landscapes (Göbel et al., 1994). Understanding these landscapes is vital for biological discovery and applications like enzyme and antibody design. Homologous proteins are particularly informative, as their shared evolutionary origin reveals fundamental sequence constraints. This makes homology crucial for various protein modeling tasks, from mutation impact prediction (Frazer et al., 2021) to protein design (Russ et al., 2020) and structure prediction (Jumper et al., 2021). Traditional approaches use Multiple Sequence Alignment (MSA) based models trained on homolog families (Krogh, 1998; Hopf et al., 2014; Frazer et al., 2021). While effective, these methods struggle with shallow or inaccurate MSAs, significant insertions/deletions, and non-alignable sequences (Riley et al., 2023). They also require new MSAs and models for each protein family. Recent Protein Language Models (pLMs) offer alignment-free approaches that leverage diverse protein sequences (Elnaggar et al., 2021; Lin et al., 2023; Nijkamp et al., 2023). However, these single-sequence models often underperform family-specific methods, particularly for specialized proteins (Notin et al., 2023). Hybrid approaches like MSA Transformer (Rao et al., 2021), Tranception (Notin et al., 2022), and PoET (Truong Jr & Bepler, 2023) combine family-specific context with broader modeling, but current retrieval frameworks remain static and cannot optimize their retrieval choices.

In this work, we propose **Protriever**, a retrieval-based protein language model that provides fast homology retrieval through a learned vector database and integrates these retrieved sequences to yield accurate zero-shot fitness predictions. Our contributions are as follows:

- We introduce an efficient mechanism, based on the Fusion-in-Decoder (Izacard & Grave, 2021), to extend off-the-shelf *single-sequence* autoregressive pLMs by allowing them to *condition* their predictions on relevant homologous sequences (§ 2.1);
- We introduce Protriever, a novel approach for *end-to-end differentiable* retrieval and sequence modeling, enabling the model to learn which sets of homologous sequences are most informative to decode a given target protein, and to adaptively refine its retrieval embeddings (§ 2.2);
- We demonstrate the value of Protriever for protein fitness prediction, across an extensive number of Deep Mutational Scanning (DMS) assays from the ProteinGym benchmark (Notin et al., 2023) (§ 3).

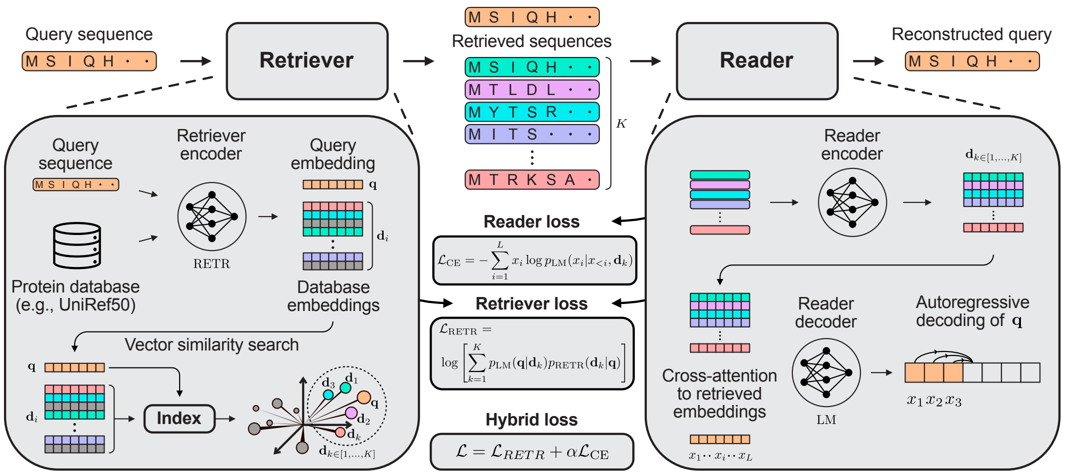

Figure 1: **Protriever**. The Protriever framework is composed of three parts: a learned retriever, index, and a reader. The two neural networks work together to produce a conditional sequence likelihood. The Retriever selects a set of sequences $\mathcal{D}_K = \{\mathbf{d}_k\}_{1,\ldots,K}$ to be passed on to the reader, using similarity search of the query to the embeddings of protein sequences in the index . This set of sequences is then passed on to the Reader, that learns to reconstruct the query from just the conditioning set $\mathcal{D}_K$. During training, the reader calculates the relevance score of each document $p_{LM}\left(\mathbf{q} \mid \mathbf{d}_k\right)$, and these relevance scores are then used to train the retriever.

Protriever addresses a key limitation in protein sequence modeling by making homology sets dynamic and learnable, rather than static, allowing adaptation to new evolutionary insights while avoiding MSA constraints.

1 Related work

Evolutionary sequence models Evolutionary models analyze protein families by first retrieving homologs, aligning them in Multiple Sequence Alignments (MSAs), and fitting statistical models to these alignments. Approaches range from site-independent models to those capturing co-evolving positions Hopf et al. (2017) and higher-order correlations using variational autoencoders Riesselman et al. (2018); Frazer et al. (2021). While effective, these alignment-based methods must be retrained per family and struggle with extensive insertions, gaps, and novel indels not present in reference alignments. These challenges can significantly distort biological interpretations derived from the data.

Protein language models (pLMs) pLMs use self-supervised learning on massive protein databases to learn evolutionary constraints that generalize across families. These models are typically trained to predict masked or next amino acids in sequences, learning the underlying patterns and dependencies that characterize functional proteins. Following UniRep's Alley et al. (2019) pioneering LSTM-based approach, Transformer-based architectures emerged, including GPT-based models (ProGen, Tranception, ProtGPT2) Madani et al. (2020); Notin et al. (2022); Ferruz et al. (2022), BERT-based models (PRoBERTa, ESM) Nambiar et al. (2020); Rives et al. (2021), and others Raffel et al. (2023); Elnaggar et al. (2021). While versatile, these models often underperform family-specific approaches and require significant computational resources.

Conditional pLMs Conditional pLMs bridge unconditional and family-specific approaches, combining language model capabilities with evolutionary insights from homologous sequences. Tranception Notin et al. (2022) merges predictions from an unconditional language model with MSA-derived frequencies through a learned weighting scheme, but remains constrained by MSA limitations. MSA Transformer Rao et al. (2021) learns directly from aligned sequences using axial attention Ho et al. (2019), while PoET Truong Jr & Bepler (2023) and ProtMamba Sgarbossa et al. (2024) eliminate alignment requirements by modeling homologous sequence sets through order-invariant mechanisms. However, these approaches rely on predefined homology sets using traditional sequence similarity metrics, rather than leveraging the model's learned representations to identify the most informative sequences for prediction - a limitation we address with Protriever's trainable retrieval mechanism.

1.1 Retrieval

Retrieval refers to searching a reference database and extracting useful information for the task of interest. In NLP, retrieval has been used for open-domain question answering Robertson & Zaragoza (2009); Grave et al. (2017); Wang et al. (2018); Karpukhin et al. (2020); Khandelwal et al. (2019) and more recently generative language models have been augmented with knowledgebases to answer highly specific questions Lee et al. (2019). The latter make use of a 'retriever', which retrieves useful context, and a 'reader', which conditions on both the query and the context to generate an answer. Going further than question-answering, methods such as REALM Guu et al. (2020) and RAG Lewis et al. (2020) introduced the concept of retrieval-augmented pre-training, where the model retrieves document chunks from a large corpus at both training and inference time in order to predict following tokens. In particular, REALM showed that it was possible to train both the retriever and reader models in an end-to-end differentiable fashion.

MSA generation is sometimes framed as a retrieval process (and has been used to augment pLMs as in the conditional pLMs section). Some work has also focused on differentiable or protein language model-based MSA generation Hong et al. (2021); Petti et al. (2023); Llinares-López et al. (2023). Closest to our retrieval-augmented methods are AIDO.RAG Li et al. (2024) and RSA Ma et al. (2024). AIDO.RAG trains a retriever to generate UniClust30 IDs, which are then used to generate better MSAs for structure prediction, and then retrains an encoder model on top of these MSAs. RSA is probably closest to RAG: it uses a frozen pretrained ESM-1b encoder as retriever and fine-tunes a reader model on a property prediction task. However, no one has yet integrated an end-to-end joint training of sequence retrieval and generation for protein sequence modeling, as we present here.

2 Methods

2.1 Protriever Framework

The Protriever framework is composed of three different components, the **Retriever** model, the **Index**, and the **Reader** model (shown in Fig. 1). The query sequence is passed through the retriever, which performs a similarity search against a fixed index of sequence embeddings. The reader model is then tasked with reconstructing the query sequence from the set of retrieved sequences. During training, the reader model learns which sequences provide useful context for reconstruction, providing feedback to the retriever that adjusts sequence relationships in embedding space given their utility for the task.

Retriever model We use an ESM-2 (Lin et al., 2023) encoder architecture as our retriever model, initialized with pre-trained weights (35M parameters). Average pooling is applied over the outputs of the last layer to obtain one 480-dimensional vector representation per sequence. A similarity score between the query $\mathbf{q}$ and each other sequence $\mathbf{d}$ is then obtained by computing the cosine similarity $s(\mathbf{d}, \mathbf{q})$ between their corresponding embeddings.

Index The retriever encoder is used to score all sequences in our database, constituting an index of UniRef50 (Suzek et al., 2015) sequences, which is searchable at very high speed using Faiss (Johnson et al., 2021), a k-nearest neighbor vector similarity search method. While previous work has shown benefits from periodically updating the index during training to maintain consistency with the evolving retriever (Izacard et al., 2022), we adopt a computationally efficient strategy of maintaining a static index during training and generating a final updated index for inference with the trained retriever. This approach significantly reduces computational overhead while still capturing the learned embedding space in the final model.

Reader model For the reader, we use the Fusion-in-Decoder model introduced in Izacard & Grave (2021) which was shown to be effective for retrieval methods in NLP (Izacard et al., 2022). The model is an encoder-decoder model, where each sequence is encoded independently from other sequences by an encoder. The decoder then attends to the concatenation of the resulting representations of all retrieved sequences. The model thus performs evidence fusion in the decoder only, and is therefore

Model type	**Model name**	**# Params**	**Spearman by MSA depth** (↑)			
			Low	Medium	High	All
Encoders	ESM2-S	35M	0.239	0.271	0.453	0.321
	ESM2-M	150M	0.306	0.358	0.500	0.388
	ESM2-L	650M	0.335	0.406	**0.517**	0.419
	ESM2-XL	3B	0.348	**0.415**	0.491	0.418
Decoders	Tranception-S	85M	0.258	0.295	0.321	0.291
	Tranception-M	300M	0.293	0.349	0.382	0.341
	Tranception-L	700M	0.358	0.371	0.417	0.382
FiD	FiD + MSA	150M	0.352	0.411	0.498	**0.420**
	FiD + frozen Protriever	150M	0.287	0.354	0.386	0.342
	FiD + trained Protriever	150M	**0.365**	0.401	0.483	0.416

Table 1: **Zero-shot substitution DMS benchmark by MSA depth**. Average Spearman's rank correlation between model scores and experimental measurements by MSA depth on the ProteinGym substitution benchmark. Tranception models are without inference-time MSA retrieval. Alignment depth is defined by the ratio of the effective number of sequences N_{eff} in the MSA, following Hopf et al. (2017), by the length covered L (Low: N_{eff}/L <1; Medium: 1< N_{eff}/L <100; High: N_{eff}/L >100). The All column is the average across the three depths.

referred to as Fusion-in-Decoder (FiD). This architecture offers significant computational advantages over a standard decoder that processes all sequences simultaneously. In a standard decoder model, processing k sequences each of length l with a query sequence also of length l, the self-attention mechanism must compute attention scores between all positions across all sequences, resulting in a complexity of $O(((k+1)l)^2)$ for the attention matrix computation. In contrast, FiD first processes each sequence independently through the encoder, then only computes cross-attention between the query sequence and the encoded representations of the conditioning sequences. This results in a complexity of $O(kl^2 + l^2k)$ where the first term represents the independent encoding of k sequences, and the second term represents the cross-attention computation in the decoder of lk hidden states by query sequence of length l . The cross-attention complexity scales linearly with the number of sequences k in the conditioning set compared to quadratic scaling in decoder-only models (see Appendix B for more details).

2.2 Training losses for the retriever

We evaluate two loss functions for training the retriever, building on approaches benchmarked in Atlas (Izacard et al., 2022) for end-to-end text retrieval. These loss functions are compatible with efficient attention mechanisms like Flash Attention 2 (Dao, 2023) as they don't require explicit attention scores.

Our approach leverages the language model's performance to guide retriever training. Specifically, if a homologous protein proves valuable for the reader's sequence modeling task, the retriever is encouraged to rank it closer to the query sequence in embedding space. This differs from traditional NLP retrieval, where document relevance is typically scored based on its utility for question answering. Adopting notation commonly used in the NLP literature, we denote the retrieved homologous proteins as $\mathbf{d}$ (documents).

The initial relevance score of a protein sequence $\mathbf{d}$ to a query sequence $\mathbf{q}$ is

$$p_{\text{RETR}}(\mathbf{d} \mid \mathbf{q}) = \frac{\exp(s(\mathbf{d}, \mathbf{q})/\theta)}{\sum_{k=1}^{K} \exp\left(s\left(\mathbf{d}_k, \mathbf{q}\right)/\theta\right)} \tag{1}$$

where $s(\mathbf{d}, \mathbf{q})$ represents the dot product of query sequence and retrieved sequences. Note the sum is over $\mathcal{D}_K = \{\mathbf{d}_k\}_{1,\ldots,K}$ of top-K retrieved sequences. This formulation approximates the true relevance score over the entire index while maintaining computational tractability by limiting backpropagation to only K sequences.

Model type	**Model name**	**# Params**	**Spearman by Function Type** (↑)				
			Activity	Binding	Expression	Organismal Fitness	Stability
Encoders	ESM2-S	35M	0.314	0.291	0.343	0.217	0.439
	ESM2-M	150M	0.391	0.326	0.402	0.305	**0.510**
	ESM2-L	650M	**0.425**	**0.337**	0.415	0.368	0.523
	ESM2-XL	3B	0.417	0.321	0.403	0.378	0.509
Decoders	Tranception-S	85M	0.288	0.286	0.349	0.321	0.27
	Tranception-M	300M	0.349	0.284	0.406	0.364	0.342
	Tranception-L	700M	0.401	0.288	0.413	0.387	0.381
FiD	FiD + MSA	150M	0.411	0.311	0.426	**0.390**	0.452
	FiD + frozen Protriever	150M	0.362	0.242	0.380	0.327	0.379
	FiD + trained Protriever	150M	0.406	0.308	**0.418**	0.384	0.436

Table 2: **Zero-shot substitution separated by function type**. Average Spearman's rank correlation between model scores and experimental measurements on the ProteinGym substitution benchmark, separated into five functional categories (Activity, Binding, Organismal Fitness, Stability, and Expression), as defined in the benchmark. Tranception models are without inference-time MSA retrieval.

End-to-end training of Multi-Document Reader and Retriever (EMDR) We first implement the approach introduced by Sachan et al. (2021), which is inspired by the expectation-maximization (EM) algorithm. In this approach, retrieved sequences are treated as latent variables. Given a query $\mathbf{q}$, and the set $\mathcal{D}_K$ of top-K retrieved sequences with the current retriever, the EMDR retriever loss is defined as:

$$\mathcal{L}_{EMDR} = -\log \left[\sum_{k=1}^{K} p_{\text{LM}} \left(\mathbf{q} \mid \mathbf{d}_k \right) p_{\text{RETR}} \left(\mathbf{d}_k \mid \mathbf{q} \right) \right]$$

where p_{RETR} represents the probability distribution over the top-K retrieved sequences, as defined by Eq. (1). During optimization, we apply a stop-gradient operator to p_{LM}, ensuring updates are limited to the retriever parameters. The theoretical optimum of this loss function is a degenerate distribution that assigns all probability mass to the single sequence maximizing the language model's likelihood of generating the correct output.

Perplexity Distillation (PDist) The second approach, introduced by Izacard et al. (2022), trains the retriever to predict the degree to which each sequence improves the language model's perplexity of the reconstructed query sequence. To that end, we minimize the KL-divergence between the relevance score for the retrieved sequence (Eq. (1)), and the posterior distribution of retrieved sequence scores based on the language model:

$$p_k = \frac{\exp \left(\log p_{\text{LM}} \left(\mathbf{q} \mid \mathbf{d}_k \right) \right)}{\sum_{i=1}^{K} \exp \left(\log p_{\text{LM}} \left(\mathbf{q} \mid \mathbf{d}_i \right) \right)}$$

2.3 Vector similarity search

Our implementation leverages Faiss for GPU-accelerated vector similarity search (Johnson et al., 2021; Douze et al., 2024). The retrieval *index* is initialized with embeddings from the pre-trained retriever encoder before model training begins. For efficient search at scale, we use an inverted file index (IVF), where the database of protein sequence embeddings is clustered using a k-means algorithm (also know as applying a *coarse* quantizer). At retrieval time, a query is compared to the resulting K_{IVF} centroids, where the nearest P_{IVF} centroids' associated vectors are searched exhaustively. To optimize memory usage and computation speed, we apply product quantization (PQ) to compress the database vectors while maintaining retrieval accuracy. The process of training the quantizers for rapid indexing is known as *training* the index. Due to the size of UniRef50 (≈ 64 million sequences) and the associated embeddings, the index is divided into partitions which are distributed across available GPUs. Queries are processed independently on each partition, with results aggregated to produce the final retrieval set. Detailed specifications of the indexing and search approach are provided in Appendix C.

Training strategies	**EMDR**	**PDist**
Pretrained reader	0.347	0.347
+retriever training on fixed BLAST sets	0.379	0.376
+retriever training on ESM sets	0.385	0.380
+retriever and reader training on ESM sets	0.404	0.397

Table 3: **Spearman on validation set for different training strategies and losses**. We evaluate the FiD model with retrieved sets, sampled with the **Distance + Uni90** scheme, using a retriever trained with each strategy. Each strategy allows the retriever to adjust relevance scores to return relevant sequences, subsequently increasing the validation performance, indicating that we are adapting our framework for the retrieval evaluation. EMDR performs slightly better than the PDist loss.

2.4 Sequence generation and scoring

Protriever is an autoregressive model, trained to predict the next token in a sequence of amino acids given retrieved sequences as context. The standard autoregressive factorization of the probability of a sequence of l amino acids is as follows:

$$P(x) = P\left(x_i \mid x_1, \ldots, x_{i-1}\right) = \prod_{i=1}^{l} P\left(x_i \mid x_{<i}\right)$$

In our retrieval framework, we extend this by conditioning on a set of K retrieved sequences $\mathcal{D}_K = \mathrm{TopK}(P_{\mathrm{RETR}}(\mathbf{d}|x))$, yielding the full probability:

$$P(x) = P_{\mathrm{RETR}}(\mathcal{D}_K|x) \prod_{i=1}^{l} P_{\mathrm{LM}}(x_i|x_{<i}, \mathcal{D}_K),$$

Following (Frazer et al., 2021; Notin et al., 2022), we evaluate the fitness of a mutated protein sequence x^{mut} via its log-likelihood ratio with respect to the wild-type sequence x^{wt}:

$$\log \frac{P\left(x^{\mathrm{mut}}\right)}{P\left(x^{\mathrm{wt}}\right)} = \log \frac{P_{\mathrm{LM}}\left(x^{\mathrm{mut}}|\mathcal{D}K\right)}{P_{\mathrm{LM}}\left(x^{\mathrm{wt}}|\mathcal{D}_K\right)} \tag{2}$$

In practice, if both x^{mut} and x^{wt} are close in sequence space (e.g., differ by a handful of mutated positions), they will share the same conditioning set $\mathcal{D}_K$.

3 Results

3.1 Pretrained reader model architecture and training

We first pre-trained a reader model composed of an ESM encoder (35M parameters) and a Tranception decoder (85M parameters). Using the Fusion-in-Decoder approach, we add cross-attention layers to the encoder, attending to the last-layer hidden state of the ESM encoder model. This results in a model with 150M parameters. This approach is agnostic to the particular choice of encoder and decoder architectures, requiring only a projection layer to ensure that the encoder and decoder have the same hidden dimension as the cross-attention layer. We pre-train the FiD model on the same dataset used for training PoET, composed of 32 millions BLAST clusters (see Appendix D). We refer to this baseline architecture as FiD.

3.2 Fitness prediction performance on ProteinGym

We evaluate the Protriever framework on ProteinGym (Notin et al., 2023), a large-scale benchmark for evaluating the zero-shot fitness prediction performance of protein sequence models. The benchmark contains more than 250 deep mutational scanning (DMS) experiments probing the natural function of many protein variants. An effective generative model of protein sequences would be expected to capture evolutionary constraints and thus perform well at mutational effect prediction.

We first establish baseline performance using our FiD architecture with traditional MSA inputs (Table 1, row FiD + MSA). This evaluation explores context sizes k of 10, 25, and 50, across the

Val Set Spearman's	**Closest + Random**	**Closest + Uni90**	**Distance + Random**	**Distance + Uni90**
Protriever (Frozen Retriever)	0.310	0.324	0.339	0.347
Protriever (Trained Retriever)	0.372	0.378	0.391	0.404

Table 4: We test four different strategies of sampling UniRef100 sequences for conditioning with either the frozen or trained retriever. We see that the **Distance + Uni90** sampling strategy yields the best results for either retrieval type, indicating that the initial frozen ESM retrieval already captures some notion of homology, which we are able to maintain and improve by training the retriever.

same five similarity thresholds used in PoET. The FiD architecture outperforms comparable models of similar size, achieving an average Spearman correlation of 0.420 compared to 0.388 for ESM-2 (150M parameters) and 0.341 for Tranception (300M parameters) across MSA depths.

To assess the impact of learned retrieval, we first evaluate FiD with a frozen retriever using pre-trained ESM embeddings (FiD + frozen Protriever). Despite optimizing the inference strategy through distance-based filtering and UniRef90 sampling (detailed in Section 3.4), this direct substitution of MSA sequences with retrieved sequences substantially degrades performance compared to the MSA-based approach. However, with end-to-end training of the retriever (FiD + trained Protriever), we achieve comparable performance to that of the MSA-based model (0.416 vs. 0.420). Notably, the trained retriever shows particular strength in low MSA depth regimes, outperforming the MSA-based approach (0.365 vs. 0.352). The following sections detail the training and inference strategies that enable this performance.

3.3 Retrieval at training strategies

We investigated three training strategies for the Protriever framework, evaluating their performance on both retrieved sequence sets and MSA set conditioning. When evaluating the retrieved sets, we sample them according to the inference scheme **Distance + Uni90** described in the next section. The results of these different approaches are shown in Section 3.1. We evaluated different training strategies, starting with fixing the reader model and only training the retriever.

- **Fixed BLAST sets:** In this strategy, we only train the retriever and fix the reader (the reader at this point is still an external pre-trained decoder model). We rely on a precomputed all-vs.-all BLAST dataset as our retrieval method at this stage. These sets represent good ground-truth sequences similar to the query, and the frozen reader model is used to provide training signal for the retriever.
- **ESM sets:** Here, we use the index search based on pre-trained embeddings, but still keep the reader fixed. As the retriever weights are updated, the query embedding will change, resulting in different sets potentially being selected. This makes this strategy closer to end-to-end training than the fixed BLAST sets.
- **Joint training:** Finally, we end-to-end train the retriever and reader. The retriever selects for each query what sequences to condition on, based on closest similarity embeddings in the index, and the reader learns how to use the retrieved sequences for sequence reconstruction.

For all retrieval methods, we used a consistent training configuration of 50,000 steps with a fixed retrieval size of $k = 10$, selecting the closest homologous sequences in the UniRef50 index. We applied separate learning rates of 4×10^{-4} for the reader and 5×10^{-5} for the retriever, using linear learning rate decay with 500 warm-up steps and a batch size of 16.

Each method was initialized from the checkpoint of the previous method. We compare in Section 3.1 the performance of all three approaches, trained with the two proposed retriever losses, EMDR and the PDist, along with a baseline using the pre-trained reader evaluated on the frozen retriever. We iteratively improve inference prediction with the three retrieval strategies and observe better results with the EMDR loss function on the validation set, which was subsequently used in the retriever configuration which scored all assays in Tables 1 and 2.

3.4 Retrieval at inference strategies

Prior to inference, we calculate an index of all UniRef50 sequences (either with a frozen ESM as encoder or with our trained retriever). This process includes calculating embeddings for all sequences in UniRef50, which takes a few hours ($\approx$ 4 hours on four A100s). We then train the Faiss index using these embeddings which takes a few minutes (Table 5). The embedding generation, while expensive, is cached for any follow-up query, and the index can be exported, as it is only a few gigabytes of VRAM (see Fig. 3 for breakdown of memory use by parameters) for subsequent use. At inference time, we test different strategies for the selection of conditioning sets for zero-shot fitness estimation:

- We take the K closest UniRef50 clusters given query/embedding similarities, where K is set to the number of sequences we can put in our conditioning set. We then sample a corresponding UniRef100 sequence, either randomly (**Closest + Random**) or inversely proportional to the size of corresponding UniRef90 (**Closest + Uni90**).
- We sample from larger set of closest $N = 300$ UniRef50 clusters, weighted by distance to the query, and sample a corresponding UniRef100 sequence randomly (**Distance + Random**) or inversely proportional to the size of corresponding UniRef90 (**Distance + Uni90**).

We also follow the experimental results from PoET and sample over multiple conditioning sets k, with k set to 10, 25, and 50. We do not sample over multiple similarity thresholds, as in PoET although later work will look into establishing similarity thresholds based on cosine similarities.

As shown in Section 3.3, we evaluate these different strategies on the validation set of ProteinGym and observe a consistent increase in predictive performance going from left to right, for *both* a frozen and trained retriever. We therefore use the best parameter combination, **Distance + Uni90**, to generate the results in Tables 1 and 2.

4 Discussion

The key advantage of Protriever is the replacement of MSA-based retrieval with vector similarity search. Given the pre-trained index, retrieval is rapid, lightweight, and scalable as shown in Appendix C. The MSA sequences used by methods such as EVE Frazer et al. (2021) rely on sensitive retrieval tools such as JackHMMER (Johnson et al., 2010), which can take hours or days to complete for a given query sequence. Other retrieval routines such as BLAST (Altschul et al., 1990) or MMseqs2 (Steinegger & Söding, 2017) can offer speed-ups but at the cost of lower retrieval precision. In contrast, our approach is 30-100x faster than even recent MMseqs2-GPU implementation (Kallenborn et al., 2025), as shown in Fig. 2, allowing large efficiency gains when predicting at scales of the entire proteome.

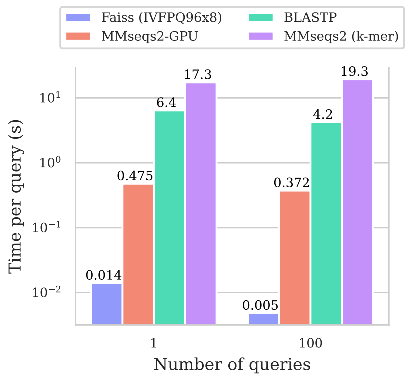

Figure 2: Retrieval time per query sequence using embedding similarity search with our approach (details in Appendix C) against BLASTP, MMseqs2, and GPU-accelerated MMseqs2, taken from Kallenborn et al. (2025). Despite the significant improvement with MMseqs2-GPU, our approach is still orders of magnitude faster.

A significant benefit of using a vector database for retrieval is that it is *dynamic*, allowing an inference dataset to be different than the trained one. This allows us to limit the inference dataset to specialized databases like GISAID (Shu & McCauley, 2017) for targeted downstream tasks, particularly useful in the case of proprietary or sensitive biological sequences.

The framework is model-agnostic: while we demonstrate it with an encoder-decoder reader, it can be adapted to decoder-only architectures like PoET and MSAGPT Chen et al. (2024), or alternative retrieval encoders such as structure-aware models (help address the stability prediction gap observed in Table 2). This flexibility, combined with the interpretability gained from analyzing retrieved sequences, opens promising directions for future research.

Meaningfulness Statement

Understanding a protein's evolutionary context through homology is crucial for meaningful biological representations. While Multiple Sequence Alignments have been the standard approach for capturing evolutionary relationships, they rely on rigid sequence similarity heuristics. Protriever reimagines this paradigm through end-to-end learned homology search, where the model discovers which sequences are truly informative for understanding protein function. By jointly optimizing retrieval and sequence reconstruction, we uncover subtle functional relationships often missed by traditional alignment methods. This creates a more nuanced and dynamic view of protein sequence space, bridging modern language models with evolutionary insights while maintaining biological interpretability through explicit sequence relationships

References

Ethan C. Alley, Grigory Khimulya, Surojit Biswas, Mohammed AlQuraishi, and George M. Church. Unified rational protein engineering with sequence-based deep representation learning. *Nature Methods*, pp. 1–8, 2019.

S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, October 1990. ISSN 0022-2836. doi: 10.1016/S0022-2836(05)80360-2.

Bo Chen, Zhilei Bei, Xingyi Cheng, Pan Li, Jie Tang, and Le Song. MSAGPT: Neural Prompting Protein Structure Prediction via MSA Generative Pre-Training, October 2024. URL `http://arxiv.org/abs/2406.05347`. arXiv:2406.05347 [q-bio].

Tri Dao. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning, July 2023. URL `http://arxiv.org/abs/2307.08691`. arXiv:2307.08691 [cs].

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The Faiss library, September 2024. URL `http://arxiv.org/abs/2401.08281`. arXiv:2401.08281 [cs].

Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Wang Yu, Llion Jones, Tom Gibbs, Tamas B. Fehér, Christoph Angerer, Martin Steinegger, Debsindhu Bhowmik, and Burkhard Rost. ProtTrans: Towards Cracking the Language of Lifes Code Through Self-Supervised Deep Learning and High Performance Computing. *IEEE transactions on pattern analysis and machine intelligence*, PP, 2021.

Noelia Ferruz, Steffen Schmidt, and Birte Höcker. ProtGPT2 is a deep unsupervised language model for protein design. *Nature Communications*, 13, 2022.

Jonathan Frazer, Pascal Notin, Mafalda Dias, Aidan Gomez, Joseph K. Min, Kelly P. Brock, Yarin Gal, and Debora S. Marks. Disease variant prediction with deep generative models of evolutionary data. *Nature*, 2021.

Edouard Grave, Moustapha M Cisse, and Armand Joulin. Unbounded cache model for online language modeling with open vocabulary. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper/2017/hash/f44ee263952e65b3610b8ba51229d1f9-Abstract.html`.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval Augmented Language Model Pre-Training. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 3929–3938. PMLR, November 2020. URL `https://proceedings.mlr.press/v119/guu20a.html`. ISSN: 2640-3498.

Ulrike Göbel, Chris Sander, Reinhard Schneider, and Alfonso Valencia. Correlated mutations and residue contacts in proteins. *Proteins: Structure, Function, and Bioinformatics*, 18(4):309–317, 1994. ISSN 1097-0134. doi: 10.1002/prot.340180402. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/prot.340180402`. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/prot.340180402.

Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial Attention in Multidimensional Transformers. *ArXiv*, abs/1912.12180, 2019. URL https://api.semanticscholar.org/CorpusID:209323787.

Liang Hong, Siqi Sun, Liangzhen Zheng, Qingxiong Tan, and Yu Li. fastMSA: Accelerating Multiple Sequence Alignment with Dense Retrieval on Protein Language, December 2021. URL https://www.biorxiv.org/content/10.1101/2021.12.20.473431v1. Pages: 2021.12.20.473431 Section: New Results.

Thomas A Hopf, Charlotta P I Schärfe, João P G L M Rodrigues, Anna G Green, Oliver Kohlbacher, Chris Sander, Alexandre M J J Bonvin, and Debora S Marks. Sequence co-evolution gives 3D contacts and structures of protein complexes. *eLife*, 3:e03430, September 2014. ISSN 2050-084X. doi: 10.7554/eLife.03430. URL https://doi.org/10.7554/eLife.03430. Publisher: eLife Sciences Publications, Ltd.

Thomas A. Hopf, John B. Ingraham, Frank J. Poelwijk, Charlotta P. I. Schärfe, Michael Springer, Chris Sander, and Debora S. Marks. Mutation effects predicted from sequence co-variation. *Nature Biotechnology*, 35(2):128–135, February 2017. ISSN 1546-1696. doi: 10.1038/nbt.3769.

Gautier Izacard and Edouard Grave. Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering, February 2021. URL http://arxiv.org/abs/2007.01282. arXiv:2007.01282 [cs].

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, November 2022. URL http://arxiv.org/abs/2208.03299. arXiv:2208.03299 [cs].

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-Scale Similarity Search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, July 2021. ISSN 2332-7790. doi: 10.1109/TBDATA.2019.2921572. URL https://ieeexplore.ieee.org/document/8733051. Conference Name: IEEE Transactions on Big Data.

L. Steven Johnson, Sean R. Eddy, and Elon Portugaly. Hidden Markov model speed heuristic and iterative HMM search procedure. *BMC Bioinformatics*, 11(1):431, August 2010. ISSN 1471-2105. doi: 10.1186/1471-2105-11-431. URL https://doi.org/10.1186/1471-2105-11-431.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A A Kohl, Andrew J Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, July 2021.

Herve Jégou, Matthijs Douze, and Cordelia Schmid. Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, January 2011. ISSN 1939-3539. doi: 10.1109/TPAMI.2010.57. URL https://ieeexplore.ieee.org/document/5432202. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

Felix Kallenborn, Alejandro Chacon, Christian Hundt, Hassan Sirelkhatim, Kieran Didi, Sooyoung Cha, Christian Dallago, Milot Mirdita, Bertil Schmidt, and Martin Steinegger. GPU-accelerated homology search with MMseqs2, January 2025. URL https://www.biorxiv.org/content/10.1101/2024.11.13.623350v3. Pages: 2024.11.13.623350 Section: New Results.

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 (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, Online, November

2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550. URL https://aclanthology.org/2020.emnlp-main.550/.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through Memorization: Nearest Neighbor Language Models. In *International Conference on Learning Representations*, 2019.

Anders Krogh. An introduction to hidden Markov models for biological sequences. In Steven L. Salzberg, David B. Searls, and Simon Kasif (eds.), *New Comprehensive Biochemistry*, volume 32 of *Computational Methods in Molecular Biology*, pp. 45–63. January 1998. doi: 10.1016/S0167-7306(08)60461-5. URL https://www.sciencedirect.com/science/article/pii/S0167730608604615.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent Retrieval for Weakly Supervised Open Domain Question Answering. In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 6086–6096, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1612. URL https://aclanthology.org/P19-1612/.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Pan Li, Xingyi Cheng, Le Song, and Eric Xing. Retrieval Augmented Protein Language Models for Protein Structure Prediction, December 2024. URL https://www.biorxiv.org/content/10.1101/2024.12.02.626519v1. Pages: 2024.12.02.626519 Section: New Results.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, March 2023. doi: 10.1126/science.ade2574. URL https://www.science.org/doi/10.1126/science.ade2574. Publisher: American Association for the Advancement of Science.

Felipe Llinares-López, Quentin Berthet, Mathieu Blondel, Olivier Teboul, and Jean-Philippe Vert. Deep embedding and alignment of protein sequences. *Nature Methods*, 20(1):104–111, January 2023. ISSN 1548-7105. doi: 10.1038/s41592-022-01700-2. URL https://www.nature.com/articles/s41592-022-01700-2. Publisher: Nature Publishing Group.

Chang Ma, Haiteng Zhao, Lin Zheng, Jiayi Xin, Qintong Li, Lijun Wu, Zhihong Deng, Yang Young Lu, Qi Liu, Sheng Wang, and Lingpeng Kong. Retrieved Sequence Augmentation for Protein Representation Learning. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 1738–1767, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.104. URL https://aclanthology.org/2024.emnlp-main.104/.

Ali Madani, Bryan McCann, Nikhil Naik, Nitish Shirish Keskar, Namrata Anand, Raphael R. Eguchi, Po-Ssu Huang, and Richard Socher. ProGen: Language Modeling for Protein Generation, 2020. _eprint: 2004.03497.

Ananthan Nambiar, Maeve Heflin, Simon Liu, Sergei Maslov, Mark Hopkins, and Anna Ritz. Transforming the Language of Life: Transformer Neural Networks for Protein Prediction Tasks. In *Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, BCB '20, pp. 1–8, New York, NY, USA, November 2020. Association for Computing Machinery. ISBN 978-1-4503-7964-9. doi: 10.1145/3388440.3412467. URL https://dl.acm.org/doi/10.1145/3388440.3412467.

Erik Nijkamp, Jeffrey A. Ruffolo, Eli N. Weinstein, Nikhil Naik, and Ali Madani. ProGen2: Exploring the boundaries of protein language models. *Cell Systems*, 14(11):968–978.e3, November 2023. ISSN 2405-4712, 2405-4720. doi: 10.1016/j.cels.2023.10.002. URL `https://www.cell.com/cell-systems/abstract/S2405-4712(23)00272-7`. Publisher: Elsevier.

Pascal Notin, Mafalda Dias, Jonathan Frazer, Javier Marchena-Hurtado, Aidan Gomez, Debora S. Marks, and Yarin Gal. Tranception: protein fitness prediction with autoregressive transformers and inference-time retrieval, May 2022. URL `http://arxiv.org/abs/2205.13760`. arXiv:2205.13760 [cs].

Pascal Notin, Aaron Kollasch, Daniel Ritter, Lood van Niekerk, Steffanie Paul, Han Spinner, Nathan Rollins, Ada Shaw, Rose Orenbuch, Ruben Weitzman, Jonathan Frazer, Mafalda Dias, Dinko Franceschi, Yarin Gal, and Debora Marks. ProteinGym: Large-Scale Benchmarks for Protein Fitness Prediction and Design. *Advances in Neural Information Processing Systems*, 36:64331–64379, December 2023. URL `https://papers.nips.cc/paper_files/paper/2023/hash/cac723e5ff29f65e3fcbb0739ae91bee-Abstract-Datasets_and_Benchmarks.html`.

Samantha Petti, Nicholas Bhattacharya, Roshan Rao, Justas Dauparas, Neil Thomas, Juannan Zhou, Alexander M Rush, Peter Koo, and Sergey Ovchinnikov. End-to-end learning of multiple sequence alignments with differentiable Smith–Waterman. *Bioinformatics*, 39(1):btac724, January 2023. ISSN 1367-4811. doi: 10.1093/bioinformatics/btac724. URL `https://doi.org/10.1093/bioinformatics/btac724`.

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, September 2023. URL `http://arxiv.org/abs/1910.10683`. arXiv:1910.10683 [cs].

Roshan M. Rao, Jason Liu, Robert Verkuil, Joshua Meier, John Canny, Pieter Abbeel, Tom Sercu, and Alexander Rives. MSA Transformer. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 8844–8856. PMLR, July 2021. URL `https://proceedings.mlr.press/v139/rao21a.html`. ISSN: 2640-3498.

Adam J Riesselman, John B Ingraham, and Debora S Marks. Deep generative models of genetic variation capture the effects of mutations. *Nature Methods*, 15(10):816–822, 2018. Publisher: Nature Publishing Group.

Andrew C. Riley, Daniel A. Ashlock, and Steffen P. Graether. The difficulty of aligning intrinsically disordered protein sequences as assessed by conservation and phylogeny. *PLOS ONE*, 18(7):e0288388, July 2023. ISSN 1932-6203. doi: 10.1371/journal.pone.0288388. URL `https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0288388`. Publisher: Public Library of Science.

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

Stephen Robertson and Hugo Zaragoza. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.*, 3(4):333–389, April 2009. ISSN 1554-0669. doi: 10.1561/1500000019. URL `https://doi.org/10.1561/1500000019`.

William P. Russ, Matteo Figliuzzi, Christian Stocker, Pierre Barrat-Charlaix, Michael Socolich, Peter Kast, Donald Hilvert, Rémi Monasson, Simona Cocco, Martin Weigt, and Rama Ranganathan. An evolution-based model for designing chorismate mutase enzymes. *Science*, 369:440 – 445, 2020. URL `https://api.semanticscholar.org/CorpusID:220714458`.

Devendra Sachan, Mostofa Patwary, Mohammad Shoeybi, Neel Kant, Wei Ping, William L. Hamilton, and Bryan Catanzaro. End-to-End Training of Neural Retrievers for Open-Domain Question Answering. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International*

Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 6648–6662, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.519. URL https://aclanthology.org/2021.acl-long.519/.

Damiano Sgarbossa, Cyril Malbranke, and Anne-Florence Bitbol. ProtMamba: a homology-aware but alignment-free protein state space model, May 2024. URL https://www.biorxiv.org/content/10.1101/2024.05.24.595730v2. Pages: 2024.05.24.595730 Section: New Results.

Yuelong Shu and John McCauley. GISAID: Global initiative on sharing all influenza data – from vision to reality. *Eurosurveillance*, 22(13):30494, March 2017. ISSN 1560-7917. doi: 10.2807/1560-7917.ES.2017.22.13.30494. URL https://www.eurosurveillance.org/content/10.2807/1560-7917.ES.2017.22.13.30494. Publisher: European Centre for Disease Prevention and Control.

Martin Steinegger and Johannes Söding. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature Biotechnology*, 35(11):1026–1028, November 2017. ISSN 1546-1696. doi: 10.1038/nbt.3988. URL https://www.nature.com/articles/nbt.3988. Publisher: Nature Publishing Group.

Baris E. Suzek, Yuqi Wang, Hongzhan Huang, Peter B. McGarvey, and Cathy H. Wu. UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics*, 31(6):926–932, March 2015. ISSN 1367-4803. doi: 10.1093/bioinformatics/btu739. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4375400/.

Timothy Truong Jr and Tristan Bepler. PoET: A generative model of protein families as sequences-of-sequences. *Advances in Neural Information Processing Systems*, 36:77379–77415, December 2023. URL https://papers.nips.cc/paper_files/paper/2023/hash/f4366126eba252699b280e8f93c0ab2f-Abstract-Conference.html.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. R3: reinforced ranker-reader for open-domain question answering. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18, pp. 5981–5988, New Orleans, Louisiana, USA, February 2018. AAAI Press. ISBN 978-1-57735-800-8.

A Appendix

B Fusion in Decoder architecture

The Fusion-in-Decoder (FiD) model integrates multiple passages by independently encoding each one and concatenating their hidden representations along the sequence dimension prior to decoding. Formally, given a batch size B and N passages per example, each passage p_i is encoded to produce hidden states $H_i \in \mathbb{R}^{B\times L_e \times d_{\text{model}}}$, where L_e is the encoder sequence length and d_{model} is the model's hidden dimension. These are concatenated to form $H_{\text{enc}} = [H_1; H_2; \ldots; H_N] \in \mathbb{R}^{B\times NL_e\times d_{\text{model}}}$.

During decoding, the cross-attention mechanism computes attention weights using queries $Q \in \mathbb{R}^{B\times L_d\times d_k}$ derived from the decoder's hidden states through a linear projection with weights $W^Q \in \mathbb{R}^{d_{\text{model}}\times d_k}$. The keys $K \in \mathbb{R}^{B\times NL_e\times d_k}$ and values $V \in \mathbb{R}^{B\times NL_e\times d_v}$ are obtained by linearly projecting the concatenated encoder outputs H_{enc} using weights $W^K \in \mathbb{R}^{d_{\text{model}}\times d_k}$ and $W^V \in \mathbb{R}^{d_{\text{model}}\times d_v}$, respectively. The attention is then computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V,$$

where L_d is the decoder sequence length, d_k is the dimensionality of the keys and queries, and d_v is the dimensionality of the values. This process allows the decoder to attend over all passages simultaneously, effectively fusing their information.

Regarding computational complexity, compared to conditioning on a single passage ($N = 1$), the cross-attention per layer increases linearly with N, as its complexity scales with $O(L_d \times NL_e \times d_{\text{model}})$.

C Vector similarity search with Faiss

We rely on Faiss for GPU-accelerated vector similarity search (Johnson et al., 2021; Douze et al., 2024), whose terminology we adopt. The vector similarity search is facilitated with an *index*, whose task it is to search a large database of vectors, $\mathbf{d}$ and return the *K* most similar ones to the query, $\mathbf{q}$, given a similarity metric.

The most simple index is a *flat* index, where the query is compared to all database entries. With the commonly used maximum inner product similarity measure, this reduces to computing $\mathbf{q} \cdot \mathbf{d}^T$ and extracting the *K* largest entries. While this search is exact, it is both slow and requires storing all database vectors in memory which is prohibitively expensive. For fast search, an inverted file index (IVF) can be used. Prior to searching the index, all entries are clustered using a "coarse quantizer", e.g., a k-means clustering algorithm, given some predefined number of centroids, K_{IVF}. At search time, the query $\mathbf{q}$ is compared to all K_{IVF} centroids, of which the P_{IVF} most similar centroids, often referred to as the number of *probes*, are searched, reducing the number of comparisons from *N* to

$$N_{\text{comparisons}} = K_{\text{IVF}} + P_{\text{IVF}} \frac{N}{K_{\text{IVF}}},$$

as per equation 18 in Douze et al. (2024). While using an IVF index reduces search time, it still requires storing the full database in memory, which for the ≈ 64 million UniRef50 sequences requires > 110 GB of memory, given the 480 dimensional mean-pooled ESM-2 35M embeddings. To overcome this major challenge, further quantization is required. We rely on a *product quantizer* (PQ) to effectively reduce the dimensionality of each vector (Jégou et al., 2011). The product quantizer partitions each vector into M sub-vectors, where each sub-vector is further separately quantized using a k-means clustering. Defining the product quantizer requires setting two parameters: the *code size*, *M*, and the number of bits with which to represent each sub-vector, where either 8 or 10 are commonly used.

Using a product quantizer dramatically reduces the index size. The memory requirement of indexing UniRef50 using three different code sizes and using a flat IVF index can be seen in Fig. 3. `IVFPQ32x8` refers to product-quantized IVF index, where each vector is divided into $M = 32$ sub-vectors, each of which is represented by 8 bits. The shown memory uses are *solely* for storing the index in memory. Using a quantizer such as PQ is therefore necessary in order to additionally store and train the Protriever model.

The process of preparing the coarse and product quantizers, e.g., by running k-means algorithms to facilitate fast search is called *training the index*.

C.1 Distributed index

We use Protriever in a distributed setting using N_{GPU} GPUs via the implementation in Izacard et al. (2022). We shard our dataset into N_{GPU} equal-sized partitions and train separate indices, where each index is responsible for $N_{\text{index}} = N/N_{\text{GPU}}$ sequences. At search time, the query is used to search each index, the results of which are aggregated. We let $N_{\text{GPU}} = 4$.

C.2 Choosing index parameters

We need to select a number of index parameters, namely the number of centroids for the coarse quantizer, K_{IVF}, the number of probes, P_{IVF}, the code size *M*, and the number of bits for the product quantizer. We have three main considerations: memory, speed, and accuracy. Given the memory uses shown Fig. 3, we now focus on gauging accuracy and speed.

C.2.1 Recall

We measure search accuracy using recall by randomly sampling $N_{\text{sample}} = 10000$ UniRef50 sequences as queries and investigating whether the query sequences are returned when searching the index.

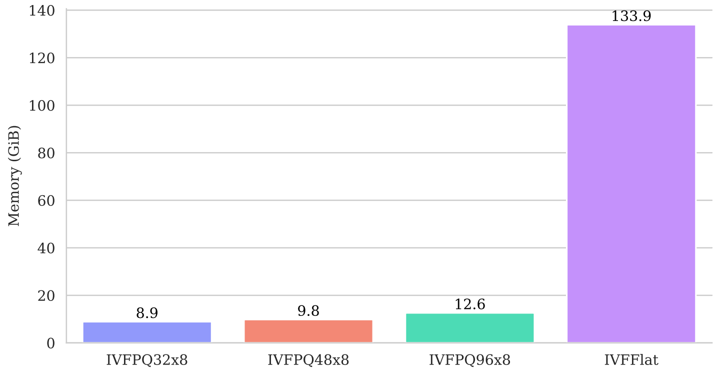

Figure 3: Index memory use. `IVFPQ32x8` refers to product-quantized IVF index, where each vector is divided into $M = 32$ sub-vectors, each of which is represented by 8 bits, while `IVFFlat` refers to an IVF index with no further quantization. Using a product quantizer dramatically reduces the memory use, potentially at the cost of search quality.

We investigate code sizes of 32, 48, and 96 (the embedding dimension needs to be divisible by the code size), as coarser quantization led to poor performance. We experiment with three different centroid counts, determined by database size: $K_{\text{IVF}} \in \{\sqrt{N_{\text{index}}}, 4\sqrt{N_{\text{index}}}, 8\sqrt{N_{\text{index}}}\}$. We fix the number of probes to $P_{\text{IVF}} = 2048$, which is the upper limit in the Faiss GPU implementation and for simplicity, we fix the number of bits per sub-vector to 8. This leads to three indices `IVFPQ32x8`, `IVFPQ48x8`, and `IVFPQ96x8`, with $K_{\text{IVF}} = \{3941, 15764, 31528\}$ (as $N_{\text{index}} \approx 15.5$ million). We use the 10000 sampled queries to search across the nine index configurations, retrieving the $K = 2048$ nearest neighbors and calculating the average recall rate at powers of 2. The results can be seen in Fig. 4. The code size has a significant impact on search quality, where $M = 32$ fails to reach

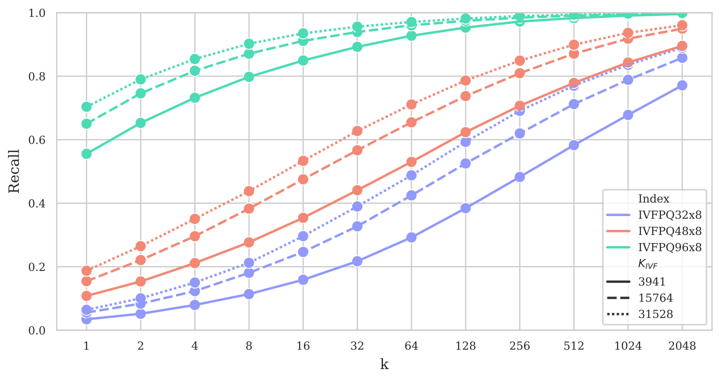

Figure 4: Recall rate vs. neighborhood sizes for `IVFPQ` indices at different quantization levels and centroids counts. 10.000 UniRef50 sequences are randomly sampled and used as queries. For each query sequence, the 2048 nearest neighbors are found. The recall indicates whether the query sequence was successfully recovered. Decreasing the quantization from 48 sub-vectors to 96 sub-vectors leads to a significant increase in recall, while doubling the number of centroids per index from $K_{\text{IVF}} = 15764$ to $K_{\text{IVF}} = 31528$ only has a marginal performance increase.

recall rates above 0.9 for $K = 2048$. Increasing the code size to $M = 96$ massively increases the recall rates, where the majority of the single-nearest neighbors return the query sequence. The search performance is less sensitive to the number of centroids, where recall increases with centroid count. For $M = 96$, we observe a persistent performance gap when using $K_{\text{IVF}} = 4\sqrt{N_{\text{index}}} = 15764$, particularly for the lower neighbor counts.

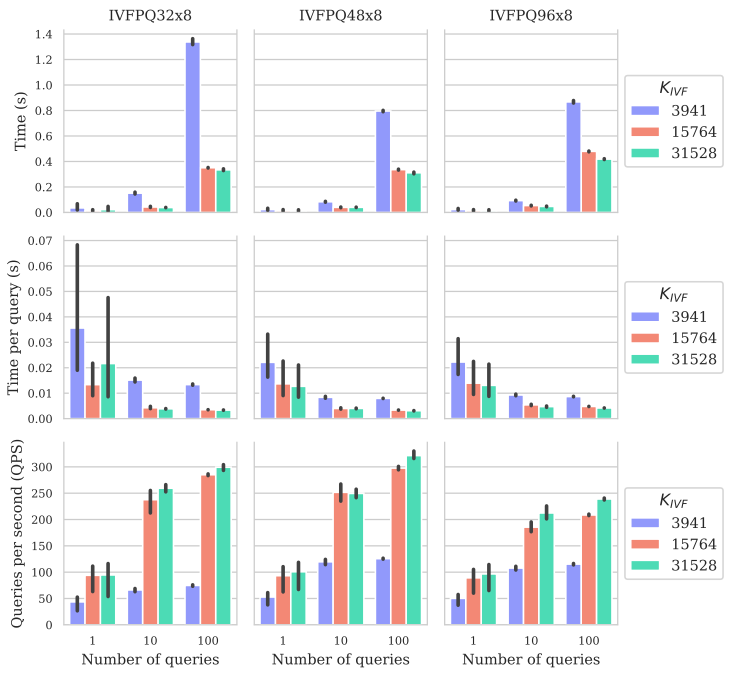

Figure 5: Impact of parameter settings on search time metrics (total time, time per query, and queries per second) across 9 configurations with varying number of queries. Results show longer search times with fewer centroids, higher per-query costs for single queries due to batch optimization, and QPS improvements that scale with number of queries..

C.2.2 SEARCH SPEED

We investigate the impact of parameter settings on search speed by measuring it over a range of scenarios. For each of the nine parameter configurations, we search using 1, 10, and 100 queries, repeating each five times. We can then visualize the search time (averaged over repeats), the time per query, and the queries per second (QPS) for each configuration. These results can be seen in Fig. 5. We see that, as expected, the search time increases with the number of queries. Using a low number of centroids (shown in blue) consistently leads to longer search times. While the search process has fewer comparisons to centroids initially, this is not outweighed by the correspondingly larger clusters. In the second row of Fig. 5 we observed that the search time per query is longer when using only a single query. This is expected as Faiss is optimized for batched searches. We also observe that using code sizes 48 and 96 approximately takes the same search time per input query. In the last row we observe that the queries per second (QPS) generally increases with the number of queries and for code size 96 appears to near a saturation point.

C.2.3 INDEX TRAINING TIME

We lastly examine how long it takes to train the index with the different parameter configurations. We train each of the nine configurations on $N_{\text{index}} \approx 15.5$ million UniRef50 sequences a total of three times. The average training time in seconds and standard error can be seen in Table 5. The training time is not sensitive to the code size but appears to linearly scale with number of centroids.

D TRAINING DETAILS ON HOMOLOGOUS DATABASE SETS FROM POET

We reuse the newest version of the Homologous database sets as was obtained by PoET, done over the UniRef50 version 2021/03 Suzek et al. (2015). The data was given to use from the PoET team and thank them for this great resource. The data was obtained by running an all vs all search of UniRef50 using Diamond, using the fol-

	Training time [s]		
	$K_{\text{IVF}} = 3941$	$K_{\text{IVF}} = 15764$	$K_{\text{IVF}} = 31528$
$M = 32$	39.00 ± 1.36	80.63 ± 0.47	190.13 ± 0.38
$M = 48$	40.33 ± 0.52	82.89 ± 0.73	191.72 ± 1.59
$M = 96$	47.03 ± 0.60	89.98 ± 0.84	200.13 ± 1.69

Table 5: **Index training times**. Average index training times (and standard error) for different parameter configurations. Each index covers $N_{\text{index}} \approx 15.5$ million UniRef50 sequences. The indexing time only slightly decreases with increased quantization. The number of centroids has a large impact on indexing time which appears to scale linearly.

lowing command, `diamond blastp -q uniref50.fasta -d diamond/uniref50 -f 6 {header -k 200000 {max-hsps 1 -e 0.001 -p 96 -o output.tab`. The command returns, for each sequence in UniRef50, a set containing all its putative homologs in UniRef50. Diamond was used over other homology search tools due to its high performance (>100x speed of BLAST). The distribution of UniRef50 clusters sizes can be seen in Fig. 6.

Following PoET's methodology, we retain only clusters with more than 10 members, yielding 32 million clusters. During training, we sample UniRef50 clusters with weight inversely proportional to the size of the UniRef50 cluster, in order to not overly represent large clusters (see Fig. 6 for an overview of cluster sizes). We then replace each UniRef50 sequence, whether query or cluster member, with a UniRef100 sequence with weight inversely proportional to the size of the corresponding UniRef90 cluster. As a form of data augmentation, we randomly reverse query sequences during training (with 50% probability), allowing reconstruction from either N-terminus to C-terminus, or vice versa. At inference time, we score sequences in both directions, a strategy shown to improve predictive performance Notin et al. (2022).

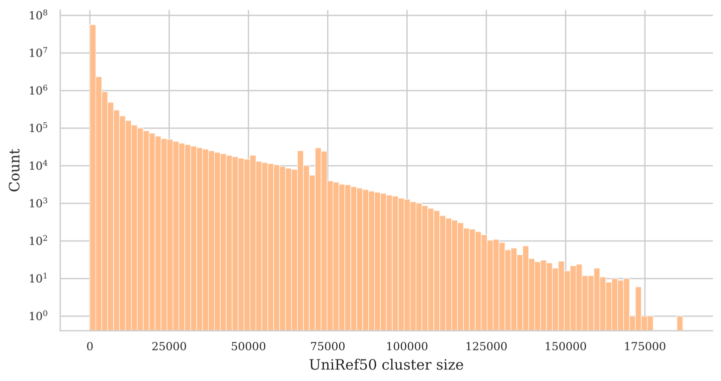

Figure 6: Distribution over cluster sizes of UniRef50.

E Software

We make our code available at `https://anonymous.4open.science/r/anon-FBE3/`.