ProtMamba: a homology-aware but alignment-free protein state space model

Damiano Sgarbossa,[1,2,†] Cyril Malbranke[1,2,†] and Anne-Florence Bitbol[1,2,*]

[1]Institute of Bioengineering, School of Life Sciences, École Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland and [2]SIB Swiss Institute of Bioinformatics, CH-1015 Lausanne, Switzerland
[†]These authors contributed equally to this work. [*]Corresponding author: anne-florence.bitbol@epfl.ch

Abstract

Motivation: Protein language models are enabling advances in elucidating the sequence-to-function mapping, and have important applications in protein design. Models based on multiple sequence alignments efficiently capture the evolutionary information in homologous protein sequences, but multiple sequence alignment construction is imperfect.
Results: We present ProtMamba, a homology-aware but alignment-free protein language model based on the Mamba architecture. In contrast with attention-based models, ProtMamba efficiently handles very long context, comprising hundreds of protein sequences. It is also computationally efficient. We train ProtMamba on a large dataset of concatenated homologous sequences, using two GPUs. We combine autoregressive modeling and masked language modeling through a fill-in-the-middle training objective. This makes the model adapted to various protein design applications. We demonstrate ProtMamba's usefulness for sequence generation, motif inpainting, fitness prediction, and modeling intrinsically disordered regions. For homolog-conditioned sequence generation, ProtMamba outperforms state-of-the-art models. ProtMamba's competitive performance, despite its relatively small size, sheds light on the importance of long-context conditioning.
Availability: A Python implementation of ProtMamba is freely available in our GitHub repository: `https://github.com/Bitbol-Lab/ProtMamba-ssm`.

Key words: Protein sequences, homologous proteins, protein language model, protein fitness prediction, protein design

1. Introduction

Proteins are essential building blocks of life, serving vital roles in most cellular processes. Composed of chains of amino acids, proteins fold into specific three-dimensional structures that are critical for their biological functions. The sustained growth of extensive protein sequence databases [1] has opened up new avenues for exploiting the breadth of biological evolution. The sequences of homologous proteins, which share a common ancestry, a similar three-dimensional structure and a similar function, are shaped by functional constraints. These constraints can be captured by methods based on information theory [2], statistical physics [3–5] and, most recently, deep learning. In particular, protein language models are enabling advances in elucidating the sequence-to-function mapping, and have important applications in protein design.

Protein language models rely on recurrent [6], transformer [7] or convolutional [8] architectures, and are trained on large ensembles of protein sequences [9]. The representations learned by these models correlate with key biological properties of proteins, such as function and structure [7, 10–13]. These models have been used with success to generate protein sequences [12, 13] and to evaluate the fitness of variants [14, 15]. The vast majority of them are trained on ensembles of single protein sequences and do not have direct access to homology. Models trained on multiple sequence alignments (MSAs) of homologous sequences have also been introduced, despite raising memory challenges and potentially suffering from the imperfections of MSAs [16]. Successful MSA-based transformer models, such as MSA Transformer [17] or the EvoFormer module of AlphaFold2 [18] alternate attention along protein sequences and across homologs. More recently, PoET [19] was trained on concatenations of non-aligned homologous sequences, offering a promising autoregressive alternative to MSA Transformer for protein fitness prediction.

State space models such as S4 [20], Hyena [21] and Mamba [22] are catching up with transformers thanks to their ability to efficiently handle very long sequences of tokens. These models were quickly adapted to work with biological data. Approaches such as HyenaDNA [23] or Evo [24] were trained on long DNA sequences and capture regulatory mechanics. Meanwhile, PTM-Mamba addresses the question of predicting post-translational modifications of protein sequences [25].

In this paper, we present ProtMamba, a novel homology-aware but alignment-free protein language model, trained on concatenated sequences of homologous proteins. Based on the Mamba architecture [22], ProtMamba is able to handle extremely long contexts, corresponding to large numbers of homologs. In fact, during inference, there is no length limitation. Trained to autoregressively predict the next amino acid, but also with a fill-in-the-middle (FIM) objective, it can be used for multiple different tasks. First, ProtMamba can autoregressively generate novel sequences without contextual information. Second, by providing ProtMamba with sequences from a specific protein family or subfamily as context, users can prompt it to generate sequences tailored to their specifications. This conditional generation approach is a key strength of the model (see also [19]), and could become an alternative to fine-tuning. Third, ProtMamba supports sequence inpainting, i.e., filling specific masked regions with the desired number of amino acids. For this, along with homologous sequences, used as context, the model is provided with a target sequence to be modified. This generation mode opens novel methods of designing specific parts of protein sequences. Furthermore, ProtMamba is useful for fitness prediction tasks. Users can input a sequence with specific masked positions, prompting the model to output the probability distribution of all mutations in each variant with

a single forward pass. Across these various tasks, we obtain competitive results with larger protein language models and task-specific methods. For homolog-conditioned sequence generation, ProtMamba outperforms current state-of-the-art models.

2. Methods

2.1. Key technical contributions

1. To harness the evolutionary information present in homologous sequences without relying on multiple sequence alignments (MSAs), we use as input a concatenation of homologous sequences for each protein family. In each of these long arrays, sequences are separated with a specific token. The motivation is that evolutionary information is extremely useful for protein modeling [17, 18, 26], but MSAs can be inaccurate. This approach is similar to that used recently in the autoregressive transformer PoET [19].
2. We develop an architecture based on Mamba blocks, an alternative to attention recently proposed by [22] that relies on state space models. In Mamba, which is a recurrent neural network, time complexity scales linearly in sequence length, bypassing the quadratic time complexity constraints of transformers. This allows handling significantly longer input sequences, in addition to being faster to train and to use at inference. This is a key asset here, as concatenating homologous sequences results in long inputs. Note that [19] employed attention matrix chunking to address this issue, but this results in potential losses of statistical dependence signals, and only partially solves the memory limit.
3. We combine elements of both autoregressive modeling and masked language modeling (MLM), by training our model using the fill-in-the-middle (FIM) objective [27–29]. The model learns to predict masked patches extracted randomly from a sequence and positioned at the end of it, and can therefore leverage the full sequence context, while being trained autoregressively. This is of particular interest for biological sequences, because preceding and subsequent tokens can all be informative to predict a new token. While autoregressive models are generative by definition, they yield the probability of each new token conditioned on previous ones (ignoring subsequent ones). Besides, MLM can be productively used for protein sequence generation [30].
4. To promote the model's ability to reason over in-sequence positions, which is particularly useful for the FIM task, we modify the original Mamba implementation by introducing sequence-level positional embeddings. This enables the model to pay attention to relative positions inside each sequence. In inference and generation, it opens the possibility of controlling the number of amino-acids to generate.

2.2. Model architecture and training strategy

ProtMamba's architecture is adapted from Mamba [22]. An important modification is that we introduce learned positional embeddings for the input tokens. Among different variants (see supplementary section C), we observed that the most effective and stable method to integrate positional embeddings is to concatenate them with the input token embeddings into a single vector. Specifically, we allocated half of the embedding dimension d to token information and the other half to positional information.

We trained a 107 million parameters model with 16 layers, embedding dimension $d = 1024$, and hidden state dimension equal to embedding dimension. We started with a maximal total input sequence length of $2^{11} = 2048$ amino acids (recall that input sequences are concatenated homologous protein sequences). The model was trained following [22] with some minor modifications. We used the AdamW optimizer with the following parameter values: weight decay $w = 0.1$ and $(\beta_1, \beta_2) = (0.9, 0.95)$. We scheduled the learning rate to increase from zero to 6×10^{-4} with a linear warm-up of 500 steps followed by a constant learning rate. To optimize memory usage, we trained the model using the `bfloat16` format.

To avoid training instabilities observed in [24, 31], we implemented a callback mechanism to revert to a previous checkpoint if the loss never assumed values below a threshold for 10 successive evaluation steps. The threshold value was chosen as the lowest training loss increased by 0.5%. This ensures that the loss decreases overall, while allowing it to transiently increase. We also prevented gradient explosions by clipping the gradient norm to 1.0.

The model was trained by scheduling the context length of the input using sequence length warm up (SLW) [23]. Initially, we used inputs of length $L = 2^{11}$ tokens with a batch size of 64. We doubled input length each time the loss reached a plateau, simultaneously reducing batch size to maintain a fixed total number of tokens per batch. In case of memory constraints, we decrease the batch size and use gradient accumulation. This heuristic approach is based on the idea that a longer context should provide more information. It is useful because of training instabilities for long contexts [23, 24]. Note that we did not start training the model with a long context to benefit from a larger batch size, which helps to approximate the loss landscape more efficiently. Finally, once we reached a context length of $L = 2^{17}$, we implemented gradient checkpointing to minimize memory consumption. This allowed us to increase the batch size for the final part of the training and obtain a better approximation of the loss landscape, see [23, 24].

The model was trained on one NVIDIA RTX 6000 GPU for 35 days, and then on two of them for 15 days. This allowed us to keep the batch size large enough when the context size increased. In total, the model was trained on 1.95×10^{11} tokens (approximately 1.5 epochs) and used 2.0×10^{20} FLOPs during training. These numbers show the huge improvements that the Mamba architecture has in terms of training speed with respect to transformers. As a comparison, the smallest ESM3 model [32] was trained with 0.8×10^{11} tokens using 6.72×10^{20} FLOPs, which means that given a fixed amount of compute, ProtMamba can see 8.5 times the tokens seen by ESM3. See Figure S1 for the training curves.

We consider two different ProtMamba versions that were obtained by saving checkpoints at different moments of the training. Our model *ProtMamba, Foundation* was trained on a maximum context length of 2^{15} tokens. Our model *ProtMamba Long, Foundation* was trained until the context length reached 2^{17} tokens. Both models were fine-tuned for 2 days on predicting only the FIM amino acids to improve inpainting capabilities, yielding the models *ProtMamba/ProtMamba Long, Fine-tuned*.

We also performed multiple ablations on the model architecture and on the different modalities by training models on 10B tokens for 50k steps (see supplementary section C).

2.3. Dataset construction

We trained ProtMamba on OpenProteinSet [33], a dataset which comprises 16 millions MSAs, one for each sequence cluster within Uniclust30 [34]. This dataset was curated to train OpenFold [35]. We used a filtered subset of the full dataset, consisting

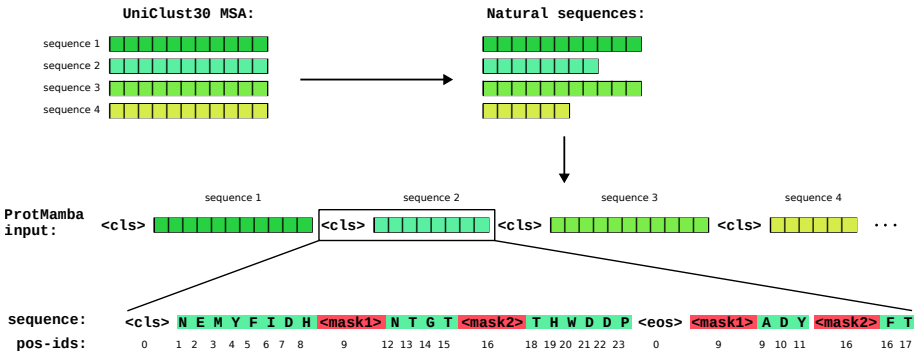

Fig. 1. Input to ProtMamba. Each element of the input is a concatenation of unaligned homologous sequences separated by <**cls**> tokens. Each sequence starts with a <**cls**> token and ends with an <**eos**> token. Masked segments are replaced by numbered mask tokens, <**mask1**>, . . . , <**mask5**>. The masked tokens are appended to the sequence, after the <**eos**> token, each masked segment being preceded by its associated mask token. The position indices ("pos-ids") follow the succession of tokens in the natural sequence. Thus, the masked tokens have their initial position indices in the natural sequence. The position index of each mask is set to that of the first associated masked token. In this particular example we sampled two masks $i = 1, 2$ with length $P_1 = 3$ and $P_2 = 2$.

of maximally diverse representative MSA clusters, built by iteratively eliminating redundant clusters whose representative sequences appeared in other clusters' MSAs [33]. This ensures that each representative sequence is only present in its cluster, as detailed in [33]. This dataset comprises 268,000 clusters including a total of 508 million sequences and 110 billion residues (see Figure S2 for additional statistics). A validation set and a testing set are formed by holding out respectively 192 and 500 randomly chosen clusters from the training set. Importantly, our use of the filtered version of OpenProteinSet [33] ensures that overlap between clusters in the training, validation and test set is strongly minimized. Indeed, this filtering is based on selecting only MSAs of maximal diversity and ensuring that the reference sequences used to build each cluster are not present in any other cluster.

Figure 1 illustrates the construction of a training example. First, a cluster is randomly selected from the filtered OpenProteinSet database described in Section 2.3. As OpenProteinSet uses MSAs, we restore the original unaligned sequences by removing gaps and converting all lowercase insertion residues to uppercase. Each amino acid is tokenized using a unique token. Then, N sequences are sampled uniformly at random and concatenated into a single array, with a <**cls**> token separating each sequence from the next one. The value of N is chosen for the total length of the concatenated sequence to exceed the desired training context length L (e.g. $L = 2^{11}$ at the beginning of training), and the input is then cropped precisely at L. Next, the sequences are prepared for the FIM task. For each sequence, some patches of consecutive tokens are randomly sampled (see below) and masked by replacing them with a mask token <**mask i**>, with one such token representing patch i. For each patch, we append to the sequence another mask token followed by the corresponding masked amino acids (which are unmasked). An <**eos**> token is used to separate the main (masked) sequence from its unmasked patches.

The following rules are applied when masking each sequence:

1. The number of masked patches in a sequence is sampled from a Poisson distribution with $\lambda = 1$, and capped at 5 (by resampling in case values above 5 are obtained). This yields no mask in 36% of sequences, one mask in 36% of sequences, and more in 28% of sequences.
2. The starting position of each patch is sampled uniformly (without replacement) from all possible positions in the sequence.
3. The length P_i of each patch i is sampled uniformly in $[1, \max(P_i)]$, where $\max(P_i)$ is 0.2 times the distance from the start point of patch i to the start point of patch $i+1$ (or to the end of the sequence for the last patch). This ensures that no more than 20% of all tokens in each sequence are masked, in line with masking fractions of similar models [7, 17].

Finally, each token is allocated a position index (used to obtain the associated positional embedding) that tracks its position in the original sequence. The position indices of <**cls**> and <**eos**> are set to zero, while the mask tokens <**mask i**> have the same position indices as the first token they are masking, see Figure 1.

3. Results

3.1. ProtMamba benefits from long context

To evaluate the effectiveness of incorporating context information in ProtMamba, we examine the scaling of the model's perplexity with context length for natural sequences. Perplexity is commonly used to evaluate autoregressive models and assesses how uncertain they are about a sequence. It is the exponential of the cross entropy loss. Figure 2 shows the perplexity for the masked parts of sequences versus the number of context sequences, when using the FIM objective. ProtMamba Long (Fine-tuned) achieves remarkably low values of perplexity for small numbers N_m of masked tokens. Furthermore, perplexity decreases when increasing the number of context sequences, revealing the positive impact of richer context on model performance. This decrease tends to be steeper for larger N_m, suggesting that these difficult tasks particularly benefit from richer context. Given the diverse lengths of sequences across protein families, we report perplexity versus the number of sequences in the context rather than versus the total length of the context. Indeed, there can be different amounts of information in contexts of similar total lengths but composed of different numbers of sequences of various lengths.

Furthermore, we study the scaling of the per-sequence perplexity (i.e. the standard autoregressive perplexity of the full non-masked sequence) computed on the test set using ProtMamba Long (Foundation), see Figure S4. Initially, we

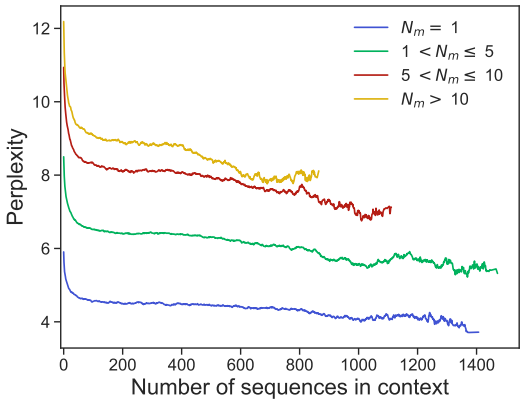

Fig. 2. Scaling of the FIM perplexity with the number of context sequences. We show the FIM perplexity for different numbers N_m of masked amino acids versus the number of context sequences. Results are averaged over all 500 clusters of the test set and 100 replicates for each cluster (differing by the random sampling of context sequences). Context sizes go up to 2^{17} amino acids. To reduce noise, we take the exponential moving average, and we restrict to cases where the count of samples is at least 100. See Figure S3 for a log-log version of this figure.

observe a decrease of perplexity to a minimum of 7.70 as the number of sequences in the context increases, with lower perplexity values for shorter individual sequences, but this decrease plateaus after a certain point. We attribute this behavior to the finite size ($d = 1024$, see Section 2) of the hidden state of the model, which limits its capacity to effectively leverage context information at each step. Indeed, when making a prediction with Mamba models (or recurrent neural networks), all the information from previous tokens in a sequence is transferred to the next token via a single hidden state vector. We hypothesize that a larger model with a higher-dimensional hidden state, or a hybrid model incorporating attention layers [31], could increase the amount of information transferred from the context to the next predicted token.

For completeness, we also report perplexity versus context length measured in tokens (see Figure S5). There, we observe a rise in perplexity when the context sizes reaches $2^{17} = 131,072$ tokens, which is the highest context length seen during training. We expect that further training the model for longer contexts could lead to lower perplexity values, yet ultimately reaching a lower bound due to the limitations imposed by the hidden state dimension and model size.

3.2. ProtMamba autoregressively generates promising novel sequences

Can ProtMamba generate synthetic homologs from given protein families by homolog-conditioned generation? To address this question, we perform autoregressive generation using ProtMamba with a context of known homologs, composed of members of a given cluster of sequences.

First, we generate sequences from 19 randomly selected clusters in the test set, varying the following parameters: temperature (T), top-k number, and top-p fraction, following the approach proposed by [12]. These parameters are commonly employed to control the output of autoregressive models. At each step, top-k limits their output to the top-k most probable tokens, while top-p only includes the top tokens reaching a cumulative probability p. Meanwhile, temperature T adjusts the randomness of sampling. Additionally, we vary the number of sequences in the context to assess the impact of different levels of conditioning on the generated sequences. Specifically, for each cluster, we perform generation using context lengths of $n = 10$, 100, 500, 1000 and N sequences, where N is the total number of sequences in the cluster. For each value of n, we consider the following $(T, \text{top-}k, \text{top-}p)$ triplets: $(0.8, 10, 0.9)$, $(0.9, 10, 0.95)$, $(1, 10, 0.95)$, $(1, 10, 1)$, $(1, 15, 1)$. We generate 100 sequences for each $(n, T, \text{top-}k, \text{top-}p)$, obtaining a total of 2500 sequences per family. As expected, we observe that the parameters which promote higher sampling variability tend to yield sequences with higher perplexity. Note that sequences with more than 750 amino acids, i.e. longer than the longest natural sequence considered here, were discarded from further analysis. They represented $\sim$5% of the generated sequences.

We compare the sampled sequences (aggregated across all parameter sets mentioned above) with natural sequences from the cluster used as context for generation using various scores evaluating novelty, homology, and structure.

1. Novelty is assessed by computing the pairwise Hamming distance (using pairwise Smith-Waterman alignment) with each natural sequence in the cluster, after which it is possible to focus on distance to the closest natural neighbor if desired.
2. Homology evaluation involves training an HMM (using HMMER [36]) on the cluster's MSA, obtained from OpenProteinSet, and computing the scores it gives to generated sequences.
3. Structure is assessed by predicting the structure of each sampled sequence using ESMFold [37]. As ESMFold is a single sequence model, it provides predictions that are less biased by MSAs than those of MSA-based models. Futhermore, it is faster than AlphaFold2. ESMFold's confidence measures, both global with pTM scores and local with pLDDT scores, allow for a precise comparison of different sequences sampled from the same cluster.

We observe that ProtMamba's estimated sequence perplexity correlates well with HMMER scores, Hamming distance to the closest natural neighbor in the cluster and structural scores (see Figure S6). Thus, ProtMamba assigns lower perplexity values to sequences that are more likely to be part of the cluster. The absolute Pearson correlation value averaged over all clusters and scores is above 0.57. Detailed results for each family and each score are presented in Figure S7. Figure 3 shows that the median scores of our generated sequences that have low perplexity are comparable to those of natural ones. Overall, these results are promising for protein design applications.

Next, in Table 1, we compare the generative ability of ProtMamba to that of other models that can perform sequence generation conditioned on homologs from a specific protein family. Following the approach of [39], we randomly sample 250 clusters from our test set, and, for each cluster, we generate a sequence using ProtMamba conditioned on homologs randomly sampled from the cluster. We then compare two structural scores of these generated sequences with those obtained for sequences generated by EvoDiff-MSA [39], MSA Transformer [17] and Potts models [40] and provided in [39], and for natural sequences. We find that ProtMamba outperforms existing models on the homolog-conditioned generation task, and generates sequences that obtain scores comparable to those of natural sequences. Note that the Hamming distance to the closest natural homolog of the sequences sampled using ProtMamba is 0.56 ± 0.10, similar to the value obtained for natural sequences (0.48 ± 0.17) and not smaller. This is consistent with Figure 3, and illustrates again the novelty of generated sequences.

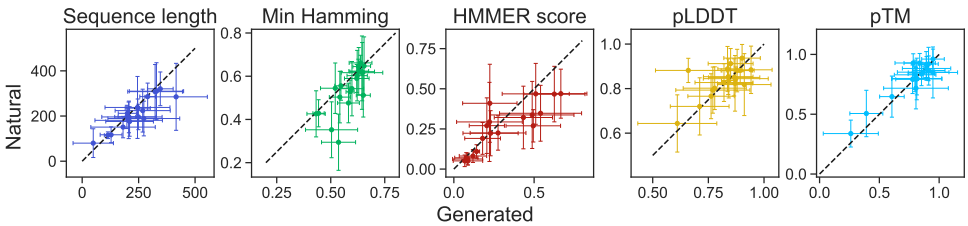

Fig. 3. Comparison of low-perplexity generated sequences with natural ones. We report the median and the standard deviation of sequence length, Hamming distance to the closest natural neighbor in the sequence cluster from which the context is drawn ("Min Hamming"), HMMER score (rescaled), pLDDT and pTM scores from ESMFold. For each of 19 test clusters, we compare the 100 sequences with lowest perplexity values out of 2500 generated sequences (x-axis) with a randomly chosen subset of 100 natural sequences in the sequence cluster (y-axis). Dashed black lines: $y = x$.

Model	ProtMamba	EvoDiff-MSA	MSA Trans.	Potts	Natural
pLDDT (↑)	$\mathbf{0.75 \pm 0.13}$	0.60 ± 0.16	0.54 ± 0.18	0.56 ± 0.14	0.77 ± 0.13
scPerplexity (↓)	$\mathbf{2.63 \pm 0.45}$	3.17 ± 0.58	3.37 ± 0.64	3.17 ± 0.51	2.66 ± 0.49

Table 1. Performance of ProtMamba and other models at homolog-conditioned generation. We report two structural scores, namely the pLDDT from ESMFold [37] and the scPerplexity from ProteinMPNN [38] for a set of 250 protein sequences generated using ProtMamba, each from a different cluster in our test set. Note that scPerplexity is the self-consistency Perplexity computed by ProteinMPNN from the ESMFold structures obtained for each generated sequence. We compare these values to those obtained for 250 protein sequences generated by EvoDiff-MSA, MSA-Transformer and Potts models, retrieved from the Zenodo archive associated to the EvoDiff paper [39], and which were generated each from a different cluster of the EvoDiff validation set. We also compare to a subset of the same size of natural sequences sampled from the same test set clusters as for ProtMamba. ↑ (resp. ↓) indicates that higher (resp. lower) scores are better.

3.3. ProtMamba can inpaint functional motifs

In addition to autoregressive generation, ProtMamba was also trained with the FIM objective. Does this allow ProtMamba to perform inpainting of protein sequences? To assess this, we considered three proteins with distinct experimentally resolved structures comprising functional motifs investigated in [41] through RFdiffusion. We masked these functional motifs in their sequences, and used ProtMamba to inpaint the masked regions. We generated 100 inpainted candidate sequences for each protein, and predicted their structures using ESMFold. Structural similarity to the native proteins was assessed using RMSD and TM-score. Figure 4 shows the predicted structures of examples inpainted sequences, and statistics are provided in Figure S8. Across all three proteins, the generated sequences substantially differ from the original ones, but exhibit high TM-scores, as well as low RMSDs both over whole structures and over motifs, suggesting that inpainted motifs match the native conformation. These results illustrate ProtMamba's ability to generate diverse but structurally consistent inpainted sequences.

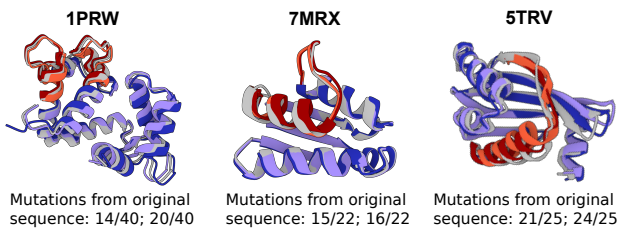

Fig. 4. Inpainted functional motifs generated by ProtMamba. For each of three protein structures (identified by their PDB IDs), we show the ESMFold-predicted structures of two example sequences inpainted by ProtMamba. Light gray: original structures; light and dark red: predicted structures of the generated functional motifs; light and dark purple: predicted structures of the rest of the sequence, which remained unchanged. The number of mutations compared to the original motif is indicated below (e.g., 14/40 means 14 mutated residues out of 40 motif residues).

3.4. ProtMamba predicts mutational effects in different protein families

Next, we evaluate ProtMamba's ability to predict mutational effects, leveraging its inpainting capabilities arising from the FIM training objective. Indeed, by masking specific amino acids in the wild-type sequence of interest, we can predict the fitness of all variants at these sites. Our first step to evaluate variant fitness is to collect a context of homologs to the wild-type sequence. We use the ColabFold protocol [42] for this, ensuring that diverse sequences are found in a few minutes. Then, we randomly subsample 200 sequences among those that have between 30% and 98% similarity to the wild type to construct the context, and we sort these sequences by increasing similarity to the wild type, as in [19].

To evaluate the effect of a variant with a single mutated site, we append the wild-type sequence to the context, mask the mutated residue in it, and predict this residue using the FIM method. Let $\mathcal{C}$ denote the union of the context sequences and of the wild-type sequence masked at the mutated position i. We evaluate the effect of mutations at position i by their fitness score $\mathcal{F}$, defined as:

$$\mathcal{F}(i, x_i, \mathcal{C}) = \log p(x_i|\mathcal{C}) - \log p(x_i^{WT}|\mathcal{C}) \,,$$

for all residues x_i different from the wild-type residue x_i^{WT}. Using this method based on the FIM objective allows us to evaluate the effects of all mutations at position i, decreasing 20-fold the number of passes through the model needed to evaluate all mutations with respect to the typical method used with autoregressive protein language models. To predict the fitness effects of variants involving mutations at multiple sites, we add all the single mutation likelihoods. This approximate, but fast method avoids computing the complete likelihood for all variants, thus reducing the number of calls to ProtMamba. It is accurate when the mutations can be considered independent. We also test variant scoring by ProtMamba using the autoregressive log-likelihood loss instead of the FIM loss (this approach is called "ProtMamba AR"). Details on the different approaches

Model type	Model	#params	ρ	Time	Citation
Alignment-based	Site-Independent	-	0.359	-	[44]
	GEMME	-	0.455	-	[45]
PLM	Tranception L (w/o R)	700M	0.374	-	[46]
	ESM-2	150M	0.387	-	[37]
	ESM-2	650M	0.414	-	[37]
Homology-aware PLM	**ProtMamba** (single)	107M	0.406	7m	
	ProtMamba AR (single)	107M	0.367	1h 39m	
	PoET (single)	201M	0.447	9h 51m	[19]
	PoET (ensemble)	201M	0.470	148h*	[19]
Alignment+ PLM	**ProtMamba** (w/ R)	107M	0.432	10m	
	MSA-Transformer	100M	0.421	-	[17]
	Tranception L (w/ R)	700M	0.434	-	[46]
	VespaG	3B	0.458	-	[47]
Structure-aware	ESM-IF1	142M	0.422	-	[48]
	SaProt	650M	0.457	-	[49]
	ProSST	110M	0.507	-	[50]

Table 2. Performance of ProtMamba and of existing models on the ProteinGym benchmark. For different models classified by type, and whose numbers of parameters are given, we show Spearman correlation ρ values between predicted and experimentally measured variant effects in ProteinGym. We denote the MSA-augmented methods with retrieval by "(w/ R)". New models introduced here are highlighted in bold. They include ProtMamba with and without retrieval, where variants are scored using the FIM loss, and ProtMamba AR, where they are scored using the autoregressive log-likelihood. Published results were obtained from `https://proteingym.org/`. For ProtMamba and PoET, we also report the time needed to score all variants in ProteinGym (excluding homolog retrieval). Top performances in terms of ρ and time are highlighted in gray.
* Estimated as 15× the time taken by PoET (single).

we employed to predict mutational effects with ProtMamba are given in Supplementary section B.

We consider the ProteinGym benchmark [43], which contains 217 datasets of substitutions in protein sequences (both single and multiple) and allows comparing to state-of-the-art methods. In Table 2, we report the performance of ProtMamba, and we compare it to published models classified by type: alignment-based models, single sequence protein language models (PLMs), aligment-enhanced PLMs, homology-aware PLMs, and structure-aware models. Table 2 shows that variant scoring by ProtMamba using FIM outperforms using the autoregressive log-likelihood (ProtMamba AR). All ProtMamba performances reported in Table 2 are those of ProtMamba Long fine-tuned on the FIM task, except for ProtMamba AR where we used ProtMamba Long. Indeed, considering the 4 different ProtMamba versions (see Section 2), we found that ProtMamba models fine-tuned on the FIM task outperform foundation models, and that ProtMamba Long performs better than ProtMamba, confirming the importance of training the model with a long context (see Figure S9). In Figure S10(a), we break down the performance of ProtMamba Long for different context lengths and different protein sequences lengths. We observe that variants with long sequences particularly benefit from long contexts, as they allow including more sequences. This interpretation is supported by Figure S10(b), which shows that this dependence on context length is weaker when considering context length in terms of number of sequences. Based on performance on a validation set (see supplementary Section A and Figure S11), we chose to use a context of 200 sequences to predict fitness using ProtMamba Long (fine-tuned).

Table 2 shows that ProtMamba outperforms single-sequence PLMs ("PLM" type) of the same size (ESM-2, 150M), and performs similarly or better than larger models like Tranception L and ESM-2 (650M). This illustrates the power of homology information for mutational effect prediction.

Since MSA information remains useful in scoring variants, in the rows "Alignment + PLM" of Table 2, we show results where explicit use of MSAs was made, either via retrieval, i.e. ensembling the models with an independent-site model, as in [43] (denoted by "R"), or by combining a PLM with GEMME in [47]. Using retrieval, ProtMamba (w/ R) obtains similar performance as Tranception L (w/ R) and as MSA Transformer, which leverage MSA information. These two models were trained using more than one order of magnitude more FLOPs than ProtMamba.

In Table 2, the state-of-the-art model on the ProteinGym benchmark is ProSST [50], a structure-aware model. Among structure-agnostic models, the state-of-the-art model is PoET, a homology-aware transformer. ProtMamba reaches a performance which is only slightly lower than PoET. This is notable, as PoET has twice more parameters, and is much slower at scoring variants than ProtMamba. Table 2 shows that ProtMamba can score all ProteinGym variants in ∼7 to 10 minutes (1.93 second per dataset on average) on a single NVIDIA RTX A6000 GPU, while PoET takes ∼10 hours for this, and up to ∼6 days in the ensemble mode (the top structure-agnostic method), using the same hardware. This speed increase is partly due to the ability to cache context efficiently. For reference, scoring the first variant from a ProteinGym dataset takes an average of 1.17 second, while all the other variants in the same dataset can be scored in the remaining 0.76 seconds (averaged over all datasets) thanks to the use of FIM and the caching process.

In Table S1, we break down results by MSA depth and by number of mutations. In Figure S12, we further break down the comparisons between models on ProteinGym by category of experiment (panel (a)), taxonomic category (panel (b)) and sequence length (panel (c)). We also show scores for different models on randomly selected example experimental datasets in Figure S13.

3.5. ProtMamba accurately predicts the activity of chorismate mutase enzymes

We further evaluate ProtMamba, and in particular the power of the FIM objective, on a dataset of experimentally tested natural and *in silico* generated sequences from the chorismate mutase family from [40]. Chorismate mutase functions as an enzyme involved in the catalysis of synthesis of amino acids, and

is a domain of the bifunctional chorismate mutase/prephenate dehydratase. We use ProtMamba to evaluate the activity of experimentally studied variants of this enzyme. For this, we sample 100 sequences, either randomly among all natural sequences that were experimentally studied, or randomly among the subset of those that were experimentally shown to be active in *Escherichia coli*. For these two types of context, we test three different protocols to predict the activity of the other variants in the dataset of [40] with ProtMamba. First, we use only the chorismate mutase domains (cropped sequences) as context, and autoregressively evaluate the likelihood of the full sequence ("from left to right"). Second, we use the full sequences (chorismate mutase/prephenate dehydratase) as context and we evaluate the perplexity of the full sequence autoregressively from left to right. Third, we use the full sequences (chorismate mutase/prephenate dehydratase) as context and evaluate the perplexity of the chorismate mutase domain using the FIM objective.

	Domain	**Protein**	**Protein, FIM**
Published methods			
DCA energy	0.41	-	-
Logistic Regression	0.43	-	-
ProtMamba			
Context: any variant	0.41	0.44	0.46
Context: active variants	0.50	0.52	**0.53**

Context size							
# seq.	0	5	10	20	50	100	200
# residues	0k	1k	3k	7k	19k	38k	77k
Spearman ρ	0.31	0.46	0.48	0.51	0.49	0.53	0.52

Table 3. Activity prediction of chorismate mutase variants. (Top) For published methods [40], and for ProtMamba with various context types (rows) and protocols (columns, see main text), we report the Spearman correlation ρ between experimental activity and predictions. Associated ROC curves are shown in Figure S14. (Bottom) Effect of increasing context size (number of sequences and corresponding total number of residues) on Spearman correlation ρ for ProtMamba predictions, using only active variants in context, full domains and FIM.

The top panel of Table 3 provides a comparison of ProtMamba against published methods [40] for the two different context types and the three different protocols. We observe that using only active variants in the context consistently improves the predictive power of ProtMamba. With both context types, using full sequences is better than using only domains, and using FIM improves accuracy. Note in addition that using FIM reduces the computation time per variant compared to autoregressively scoring the full sequence.

The bottom panel of Table 3 shows the impact of context size on the performance of our best ProtMamba-based activity predictor (using only active variants in context, full domains and FIM). The accuracy of this predictor initially strongly increases with context length, and then appears to plateau from a context length around 75 sequences or 28k residues (see also Figure S15 **a**). Thus, context size plays a critical role for these activity predictions. In Figure S15 **b** and **c**, we further compare the perplexity of the variants, when using only active variants as context, and when using both inactive and active variants as context. We observe that the perplexity of inactive variants is often higher when using a context of active variants, indicating a better ability to predict inactivity. Furthermore, the perplexity of active variants is often lower in this case, showing a better ability to predict activity with high-quality context. Additionally, we display the distribution of the perplexity for ProtMamba using FIM and a context composed of active variants, compared to the experimental activity in Figure S16, and the same perplexity versus the model score from [40] in Figure S17.

3.6. ProtMamba robustly models disordered regions

Since ProtMamba does not rely on MSAs, we finally ask how it compares to MSA-based models for modeling intrinsically disordered regions, which are typically difficult to align [51]. To explore this, we considered proteins with residues labeled as structured or disordered in the Critical Assessment of Intrinsic Disorder (CAID) [52, 53], and collected homologous sequences for each of them. We evaluated residue-level likelihoods using two approaches: (i) MSA Transformer [17], using masked language modeling (MLM) on aligned homologous sequences, and (ii) ProtMamba, using its fill-in-the-middle (FIM) objective on concatenated unaligned homologous sequences.

Figure 5 shows the distribution of negative log-likelihoods in disordered versus structured (non-disordered) regions obtained from both models. MSA Transformer exhibits a noticeable performance drop in disordered regions, reflected by a heavier-tailed distribution of negative log-likelihoods, indicating that disordered regions often have smaller likelihoods than structured ones, and a Kullback–Leibler (KL) divergence of 0.038 between the two region types. In contrast, ProtMamba shows much more consistent performance between structured and disordered regions, with a KL divergence of only 0.011. Note that computation took 7 min using ProtMamba, compared to 7.5 h using MSA Transformer, and that we had to exclude 54 of the 250 initially randomly chosen sequences from the analysis because they were too long to be used as input of MSA Transformer ($>$ 1024 residues). These results suggest that ProtMamba is more effective than MSA-based models at modeling sequences that are long and challenging to align, such as intrinsically disordered regions.

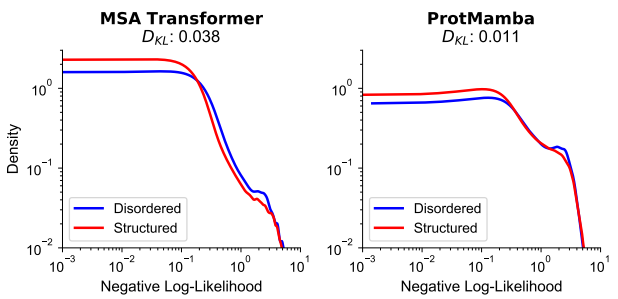

Fig. 5. Residue log-likelihoods in disordered and structured regions. We show the distributions of the negative log-likelihoods of amino acids in disordered regions versus structured regions, as predicted by ProtMamba (via FIM) and MSA Transformer (via MLM). We report KL divergences (D_{KL}) between disordered and structured distributions. Kernel density estimates of the distributions were computed over 196 protein sequences from the CAID benchmark. For each sequence, we recovered homologous sequences from UniRef100 via MMseqs2 (coverage threshold of 80% and E-value cutoff of 10^{-5}) and used them as context.

4. Discussion

Here, we presented ProtMamba, a homology-aware but alignment-free generative protein language model. ProtMamba leverages the long-context capabilities of state space models, allowing it to handle concatenated sequences of homologous proteins. It also benefits from their faster speed compared to attention-based models [22], allowing fast sequence generation and mutational effect prediction. ProtMamba was trained using a hybrid strategy combining autoregressive modeling and masked language modeling via the FIM objective. This allows ProtMamba to efficiently predict the next amino acid in a protein sequence as well as to inpaint masked regions.

ProtMamba is versatile, allowing conditioned generation, sequence inpainting, and protein fitness prediction for both close and distant variants. For homolog-conditioned generation, ProtMamba outperforms the state-of-the-art model EvoDiff-MSA [39]. ProtMamba also shows promise for sequence inpainting via FIM, generating diverse but structurally consistent functional motifs. For fitness prediction, the sequence inpainting abilities of ProtMamba proved particularly useful. Indeed, they allow the model to exploit the full sequence context, without restricting to previous tokens as with autoregressive generation. This allows ProtMamba to reach similar performance levels as larger models, in a fraction of the time. Finally, ProtMamba is a promising approach for modeling intrinsically disordered regions, and potentially other sequences that are challenging to align.

Overall, ProtMamba benefits from capturing signal across multiple scales. In particular, it is able to predict fitness by exploiting constraints shared broadly across the proteome via its pre-training, but also specific constraints shared between homologs via the context, and it can exploit the full context of a given protein sequence when predicting only part of it.

Limitations. So far, ProtMamba did not reach perplexity values as low as those of larger transformer models like PoET [19] for full sequences. However, it can handle longer context sizes and requires much shorter training and inference times, which is highly beneficial for sequence inpainting. Scaling the model to larger sizes and training times (comparable to PoET) may result in comparable performance, while retaining ProtMamba's assets of lower memory cost and inference time.

Perspectives. ProtMamba is flexible, as it allows for precise conditioning by carefully choosing the context information (e.g. restricting to active sequences). Thus, ProtMamba responds very well to prompt engineering. We propose that this could become an alternative or complement to fine-tuning of language models. ProtMamba is also naturally designed to take advantage of retrieval augmented generation (RAG) techniques [54], as it allows for using retrieved protein sequences from any external database, to condition the generation process.

Furthermore, we envision using the model for homology search, by scoring sequences within specific contexts. This would be very fast, because only one forward pass would be required.

An interesting further extension of ProtMamba would be to make it explicitly structure-aware, e.g. using a structural alphabet [55], along the lines of SaProt [49] or ProstT5 [56]. Another possible extension would be to include Gene Ontology (GO) terms to condition sequence generation [13, 57].

References

1. The UniProt Consortium, "UniProt: the universal protein knowledgebase in 2021," *Nucleic Acids Research*, vol. 49, no. D1, pp. D480–D489, 2021.
2. S. Dunn, L. Wahl, and G. Gloor, "Mutual information without the influence of phylogeny or entropy dramatically improves residue contact prediction," *Bioinformatics*, vol. 24, no. 3, pp. 333–340, 2008.
3. M. Weigt *et al.*, "Identification of direct residue contacts in protein-protein interaction by message passing," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 106, no. 1, pp. 67–72, 2009.
4. F. Morcos *et al.*, "Direct-coupling analysis of residue coevolution captures native contacts across many protein families," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 108, no. 49, pp. E1293–1301, 2011.
5. D. S. Marks *et al.*, "Protein 3D structure computed from evolutionary sequence variation," *PLoS ONE*, vol. 6, no. 12, pp. 1–20, 2011.
6. T. Bepler and B. Berger, "Learning protein sequence embeddings using information from structure," in *ICLR*, 2019.
7. A. Rives *et al.*, "Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 118, no. 15, 2021.
8. K. K. Yang, N. Fusi, and A. X. Lu, "Convolutions are competitive with transformers for protein sequence pretraining," *Cell Syst*, vol. 15, no. 3, pp. 286–294, Mar 2024.
9. H. Khakzad *et al.*, "A new age in protein design empowered by deep learning," *Cell Systems*, vol. 14, no. 11, pp. 925–939, 2023.
10. A. Elnaggar *et al.*, "ProtTrans: Towards cracking the language of life's code through self-supervised deep learning and high performance computing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021.
11. J. Vig *et al.*, "BERTology meets biology: Interpreting attention in protein language models," in *ICLR*, 2021.
12. N. Ferruz, S. Schmidt, and B. Höcker, "ProtGPT2 is a deep unsupervised language model for protein design," *Nature communications*, vol. 13, no. 1, p. 4348, 2022.
13. A. Madani *et al.*, "Large language models generate functional protein sequences across diverse families," *Nat. Biotechnol.*, vol. 41, no. 8, pp. 1099–1106, 2023.
14. J. Meier *et al.*, "Language models enable zero-shot prediction of the effects of mutations on protein function," in *Advances in Neural Information Processing Systems*, A. Beygelzimer *et al.*, Eds., 2021.
15. P. Kantroo, G. P. Wagner, and B. B. Machta, "Pseudo-perplexity in one fell swoop for protein fitness estimation," *arXiv*, p. 2407.07265, 2024.
16. J. D. Thompson *et al.*, "A comprehensive benchmark study of multiple sequence alignment methods: current challenges and future perspectives," *PLoS One*, vol. 6, no. 3, p. e18093, Mar 2011.
17. R. M. Rao *et al.*, "MSA Transformer," in *Proceedings of the 38th International Conference on Machine Learning*, vol. 139. PMLR, 2021, pp. 8844–8856.
18. J. Jumper *et al.*, "Highly accurate protein structure prediction with AlphaFold," *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.
19. T. Truong Jr and T. Bepler, "PoET: A generative model of protein families as sequences-of-sequences," *Advances in Neural Information Processing Systems*, vol. 36, 2024.
20. A. Gu, K. Goel, and C. Ré, "Efficiently modeling long sequences with structured state spaces," *arXiv:2111.00396*, 2021.
21. M. Poli *et al.*, "Hyena hierarchy: Towards larger convolutional language models," in *International Conference on Machine Learning*. PMLR, 2023, pp. 28 043–28 078.
22. A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," *arXiv*, 2023.
23. E. Nguyen *et al.*, "HyenaDNA: Long-range genomic sequence modeling at single nucleotide resolution," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.
24. E. Nguyen *et al.*, "Sequence modeling and design from molecular to genome scale with evo," *Science*, vol. 386, no. 6723, p.

eado9336, 2024.
25. Z. Peng, B. Schussheim, and P. Chatterjee, "PTM-Mamba: A PTM-aware protein language model with bidirectional gated Mamba blocks," *bioRxiv*, 2024.
26. J. Abramson *et al.*, "Accurate structure prediction of biomolecular interactions with AlphaFold 3," *Nature*, May 2024.
27. M. Bavarian *et al.*, "Efficient training of language models to fill in the middle," *arXiv*, p. 2207.14255, 2022.
28. D. Fried *et al.*, "Incoder: A generative model for code infilling and synthesis," *arXiv:2204.05999*, 2022.
29. C. Raffel *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer," *arXiv*, p. 1910.10683, 2020.
30. D. Sgarbossa, U. Lupo, and A.-F. Bitbol, "Generative power of a protein language model trained on multiple sequence alignments," *Elife*, vol. 12, p. e79854, 2023.
31. R. Waleffe *et al.*, "An empirical study of mamba-based language models," *arXiv*, p. 2406.07887, 2024.
32. T. Hayes *et al.*, "Simulating 500 million years of evolution with a language model," *Science*, p. eads0018, 2025.
33. G. Ahdritz *et al.*, "OpenProteinSet: Training data for structural biology at scale," *Advances in Neural Information Processing Systems*, vol. 36, 2024.
34. M. Mirdita *et al.*, "Uniclust databases of clustered and deeply annotated protein sequences and alignments," *Nucleic Acids Research*, vol. 45, no. D1, pp. D170–D176, 2017.
35. G. Ahdritz *et al.*, "Openfold: Retraining alphafold2 yields new insights into its learning mechanisms and capacity for generalization," *Nature Methods*, pp. 1–11, 2024.
36. S. C. Potter *et al.*, "Hmmer web server: 2018 update," pp. W200–W204, 2018.
37. Z. Lin *et al.*, "Evolutionary-scale prediction of atomic-level protein structure with a language model," *Science*, vol. 379, no. 6637, pp. 1123–1130, 2023.
38. J. Dauparas *et al.*, "Robust deep learning–based protein sequence design using proteinmpnn," *Science*, vol. 378, no. 6615, pp. 49–56, 2022.
39. S. Alamdari *et al.*, "Protein generation with evolutionary diffusion: sequence is all you need," *bioRxiv*, 2023.
40. W. P. Russ *et al.*, "An evolution-based model for designing chorismate mutase enzymes," *Science*, vol. 369, no. 6502, pp. 440–445, 2020.
41. J. L. Watson *et al.*, "De novo design of protein structure and function with RFdiffusion," *Nature*, vol. 620, no. 7976, pp. 1089–1100, Aug. 2023.
42. M. Mirdita *et al.*, "ColabFold: making protein folding accessible to all," *Nat Methods*, vol. 19, no. 6, pp. 679–682, 2022.
43. P. Notin *et al.*, "Proteingym: Large-scale benchmarks for protein fitness prediction and design," *Advances in Neural Information Processing Systems*, vol. 36, 2024.
44. T. A. Hopf *et al.*, "Mutation effects predicted from sequence co-variation," *Nat Biotechnol*, vol. 35, no. 2, p. 128, 2017.
45. E. Laine, Y. Karami, and A. Carbone, "Gemme: A simple and fast global epistatic model predicting mutational effects," *Molecular Biology and Evolution*, vol. 36, no. 11, pp. 2604–2619, 08 2019.
46. P. Notin *et al.*, "TranceptEVE: Combining family-specific and family-agnostic models of protein sequences for improved fitness prediction," *bioRxiv*, 2022.
47. C. Marquet *et al.*, "Expert-guided protein language models enable accurate and blazingly fast fitness prediction," *Bioinformatics*, vol. 40, no. 11, p. btae621, 2024.
48. C. Hsu *et al.*, "Learning inverse folding from millions of predicted structures," *PMLR*, pp. 8946–8970, 2022.
49. J. Su *et al.*, "SaProt: Protein language modeling with structure-aware vocabulary," *bioRxiv*, 2023.
50. M. Li *et al.*, "Prosst: Protein language modeling with quantized structure and disentangled attention," *bioRxiv*, pp. 2024–04, 2024.
51. A. C. Riley, D. A. Ashlock, and S. P. Graether, "The difficulty of aligning intrinsically disordered protein sequences as assessed by conservation and phylogeny," *PLOS ONE*, vol. 18, no. 7, p. e0288388, 2023.
52. M. Necci, D. Piovesan, and S. C. Tosatto, "Critical assessment of protein intrinsic disorder prediction," *Nature methods*, vol. 18, no. 5, pp. 472–481, 2021.
53. A. D. Conte *et al.*, "Critical assessment of protein intrinsic disorder prediction (CAID) - Results of round 2," *Proteins: Structure, Function, and Bioinformatics*, vol. 91, no. 12, pp. 1925–1934, 2023.
54. P. Lewis *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," pp. 9459–9474, 2020.
55. M. van Kempen *et al.*, "Fast and accurate protein structure search with Foldseek," *Nat. Biotechnol.*, 2023.
56. M. Heinzinger *et al.*, "Bilingual language model for protein sequence and structure," *NAR Genomics and Bioinformatics*, vol. 6, no. 4, p. lqae150, 2024.
57. E. Nijkamp *et al.*, "Progen2: exploring the boundaries of protein language models," *Cell systems*, vol. 14, no. 11, pp. 968–978, 2023.
58. T. Truong Jr and T. Bepler, "PoET: A high-performing protein language model for zero-shot prediction," https://www.openprotein.ai/poet-a-high-performing-protein-language-model-for-zero-shot-prediction.

Supplementary material for "ProtMamba: a homology-aware but alignment-free protein state space model"

A. ProteinGym assays used in validation

Here, we list the 20 assays we extracted from the ProteinGym benchmark to choose some hyperparameters (see Figure S11).

A0A2Z5U3Z0_9INFA_Wu_2014
AMFR_HUMAN_Tsuboyama_2023_4G3O
CAR11_HUMAN_Meitlis_2020_lof
CBS_HUMAN_Sun_2020
CUE1_YEAST_Tsuboyama_2023_2MYX
DYR_ECOLI_Nguyen_2023
GDIA_HUMAN_Silverstein_2021
HIS7_YEAST_Pokusaeva_2019
HXK4_HUMAN_Gersing_2023_abundance
KCNE1_HUMAN_Muhammad_2023_expr
KKA2_KLEPN_Melnikov_2014
PITX2_HUMAN_Tsuboyama_2023_2L7M
PPM1D_HUMAN_Miller_2022
R1AB_SARS2_Flynn_2022
RDRP_I33A0_Li_2023
S22A1_HUMAN_Yee_2023_abundance
SCN5A_HUMAN_Glazer_2019
SHOC2_HUMAN_Kwon_2022
TRPC_SACS2_Chan_2017
VILI_CHICK_Tsuboyama_2023_1YU5

B. Details on mutational effect prediction with ProtMamba

In Table 2, we consider different ways of computing mutational effects using ProtMamba, and compare with other models. Here, we explain our different approaches in more detail.

- The entry **ProtMamba (single)** of Table 2 is the result reported when using the FIM technique at the end of the sequence to evaluate mutations. It is the fastest method among homology-aware ones. The procedure is the following:
1. Subsample a predetermined number of homologs of the target sequence considered in the DMS to be used as context, based on a diversity filter.
2. Run ProtMamba on the context and collect the last hidden state of the context (as the model is recurrent).
3. Start from the last hidden state as initial state for every variant to score. Scoring is then very fast, since we only need to apply ProtMamba on a single sequence per variant, and not on the full context. As Mamba scales linearly in sequence length, this allow to evaluate many different variants very fast on a single GPU (hundreds to thousands per batch). The mutated residues are put at the end of the sequence in the FIM mask. For single mutations, we can then evaluate in one shot the likelihood of every mutation.
4. We compare the likelihood of the WT to the likelihood of the variant using Fill-in-the-middle to get a proxy of variant fitness. We can then evaluate the fitness of variants comprising each of the 20 amino-acids at the mutated site(s) in a single shot.
- The entry **ProtMamba AR (autoregressive)** of Table 2 is the result reported when evaluating the likelihood of a variant without using the FIM technique. The procedure is:
1. Perform steps 1 and 2 as in ProtMamba (single) above.
2. Start from the last hidden state as initial state for every variant to score. We then evaluate the autoregressive likelihood of the full variant. Since some logits are computed after the mutation in this setup (because they are positioned after in the sequence), we cannot evaluate the fitness of the 20 amino-acids in a single shot (in contrast to the previous approach), which increases the number of calls to ProtMamba, and hence requires more time.
3. Compare the likelihood of the WT over the full sequence to the likelihood of the variant to get an evaluation of variant fitness.
- The entry **ProtMamba (w/ R), i.e. with retrieval** of Table 2 is the result reported when combining ProtMamba with a prior based on the frequency of amino-acids in the MSA of the relevant protein family.
1. Perform steps 1, 2 and 3 as in ProtMamba (single) above.
2. Load the MSA, compute a log-likelihood prior ($\log p_{\text{retrieval}}$) based on the frequency of every amino-acid, and sum it with ProtMamba's log-likelihood ($\log p_{\text{ProtMamba}}$) :

$$\log p_{\text{ProtMamba (w/R)}} = \alpha \log p_{\text{retrieval}} + (1-\alpha) \log p_{\text{ProtMamba}} .$$

The parameter α was optimized using the validation set introduced in Section A (see Figure S11). This operation can be parallelized across CPUs or GPUs. We report the results using 16 workers.

Note that we tested ProtMamba AR both with our model fine-tuned on FIM (ProtMamba Long Finetuned) and with our foundation model (ProtMamba Long), and we obtained similar results (respectively 0.361 and 0.367).

In Table S1, we break down results by MSA depth and by number of mutations. We observe that for datasets with more than one mutation (last column in Table S1), ProtMamba with retrieval slightly outperforms the overall state-of-the-art model TranceptEVE L and reaches performance close to the structure-based model ESM-IF1. However, averaging over all datasets, ProtMamba does not reach the same performance as TranceptEVE L. But since ProtMamba performs better than Tranception L, ensembling ProtMamba and EVE predictions might yield comparable performance.

Model	Par.	Spearman correlation by MSA depth				by mutations	
		All depths	Deep	Medium	Shallow	1	2+
ESM-2	150M	0.387	0.497	0.358	0.306	0.367	0.379
ESM-IF1	142M	0.422	0.544	0.431	0.300	0.413	0.471
Tranception S (w/o R)	85M	0.303	0.320	0.295	0.258	0.293	0.262
Tranception L (w/o R)	700M	0.374	0.419	0.371	0.358	0.358	0.390
ProtMamba (w/o R)	107M	0.406	0.465	0.411	0.391	0.376	0.444
MSA Transformer	100M	0.421	0.473	0.435	0.393	0.392	0.435
Tranception S (w/ R)	85M	0.418	0.444	0.415	0.428	0.389	0.409
Tranception L (w/ R)	700M	0.434	0.473	0.438	0.432	0.404	0.463
TranceptEVE L	>700M	0.456	0.492	0.467	0.451	0.426	0.467
ProtMamba (w/ R)	107M	0.432	0.472	0.438	0.448	0.404	0.469

Table S1. Performance of different models on the ProteinGym benchmark. We report Spearman correlation values obtained both based on retrieval (w/ R) and non-retrieval (w/o R) methods, and parameter count for each model. We report results divided according to MSA depth and number of mutations in the benchmark dataset. Results for benchmark models were obtained from `https://proteingym.org/`. Note that PoET-205M [19] reports an overall Spearman correlation of 0.474 [58] on ProteinGym, but it is not yet on the ProteinGym website, and no information is given about the training time or resources.

C. Ablation Studies

We investigated ablations or alternative implementations of ProtMamba using two models: a small model with 14 million parameters (8 layers, hidden dimension 512) and the standard architecture with 107 million parameters (16 layers, hidden dimension 1024). Both models were trained for 10 billion tokens (50k steps with a batch size of 128) and evaluated on a validation set of 500 unseen clusters. During evaluation, a context of 25 sequences was used. The perplexity of the models was assessed both autoregressively (left-to-right) and in the fill-in-the-middle (FIM) spans.

The performance of these alternatives (described in more detail below) is summarized in Table S2. Perplexity values are reported for both autoregressive and FIM modes in the small model and the larger one.

Perplexity	14M Parameters		107M Parameters	
	Autoregressive	FIM	Autoregressive	FIM
Only FIM from scratch	Fail	13.90 ± 0.34	Fail	15.59 ± 0.27
AR only	**12.58** ± 0.31	18.03 ± 0.25	11.05 ± 0.36	Fail
No positional encoding	13.01 ± 0.30	16.71 ± 0.47	12.31 ± 0.37	17.20 ± 0.58
Additive positional encoding	12.72 ± 0.31	13.60 ± 0.33	12.58 ± 0.38	13.81 ± 0.31
One mask, one token	12.76 ± 0.31	15.54 ± 0.29	11.04 ± 0.33	16.60 ± 0.36
Masking fraction 50%	13.02 ± 0.31	**13.44** ± 0.33	**10.94** ± 0.36	**11.59** ± 0.35
ProtMamba	13.00 ± 0.30	13.89 ± 0.32	11.35 ± 0.33	12.62 ± 0.30

Table S2. Alternatives to ProtMamba. Perplexity values are reported for different alternatives to ProtMamba, evaluated on small (14M parameters) and larger (107M parameters) models, for both autoregressive and FIM tasks.

The alternative implementations tested against our main ProtMamba model, and whose performance is reported in Table S2, were constructed as follows:

- **Only FIM from scratch:** This approach backpropagates the loss exclusively from the FIM tokens, disregarding the main amino-acid chain. Training this way from scratch disables autoregressive (left-to-right) next-token prediction and degrades performance, including on FIM tasks.
- **Autoregressive (AR) only:** Trains the model without sampling FIM spans. While this slightly improves autoregressive performance, it significantly degrades FIM capabilities.
- **No positional encoding:** Omits positional encodings entirely. In autoregressive mode, the model can partially rely on its recurrent architecture, but in FIM mode, performance suffers due to the absence of positional information in input.
- **Additive positional encoding:** Uses additive positional encoding (summing token embeddings with positional encoding) instead of concatenated positional encoding (concatenating token embeddings with positional encoding). This approach showed mixed results, with slight improvement in the small model but degradation in the larger model.
- **One mask, one token:** Uses one mask per token (as in the T5 model [29]) instead of one mask per span of tokens (as in our approach, inspired by [27]). This approach led to performance degradation in FIM, likely due to insufficient training on larger number of mask tokens.
- **Masking fraction 50%:** Samples 50% of the tokens for FIM (compared to ProtMamba's 20%). This alternative brought minor but noticeable improvements, suggesting potential for further development.

D. Alternative Models

We compared Mamba against Hyena and GPT using flash attention, with small scale models. Models were built to have 14M parameters: 8 layers, 512 dimension representations (384 for GPT). We trained these models with context size 32.000 for 72h on single H100 GPUs, and we tested their perplexity on a held out validation set using both 16,000 and 120,000 context length. We report the results in Table S3.

Model	**Training Time**	**Tokens Seen**	**Context Short**		**Context Long**	
			PPL	**Eval Rate**	**PPL**	**Eval Rate**
Hyena (14M)	72h	81.9B	14.42	$16.6s^{-1}$	Fail	Fail
Mamba (14M)	72h	76.8B	12.48	$16.6s^{-1}$	11.89	$3.84s^{-1}$
GPT2 (14M)	72h	33.3B	10.68	$1.25s^{-1}$	11.72	$0.03s^{-1}$
ProtMamba (107M)	400h	190B	9.69	$8.33s^{-1}$	9.26	1.04

Table S3. Performance Metrics for Various Models, after 72h of training. Perplexity (PPL) and evaluation rate are reported using various architectures, for short (16,000 tokens) and long (120,000 tokens) context lengths.

E. Supplementary figures

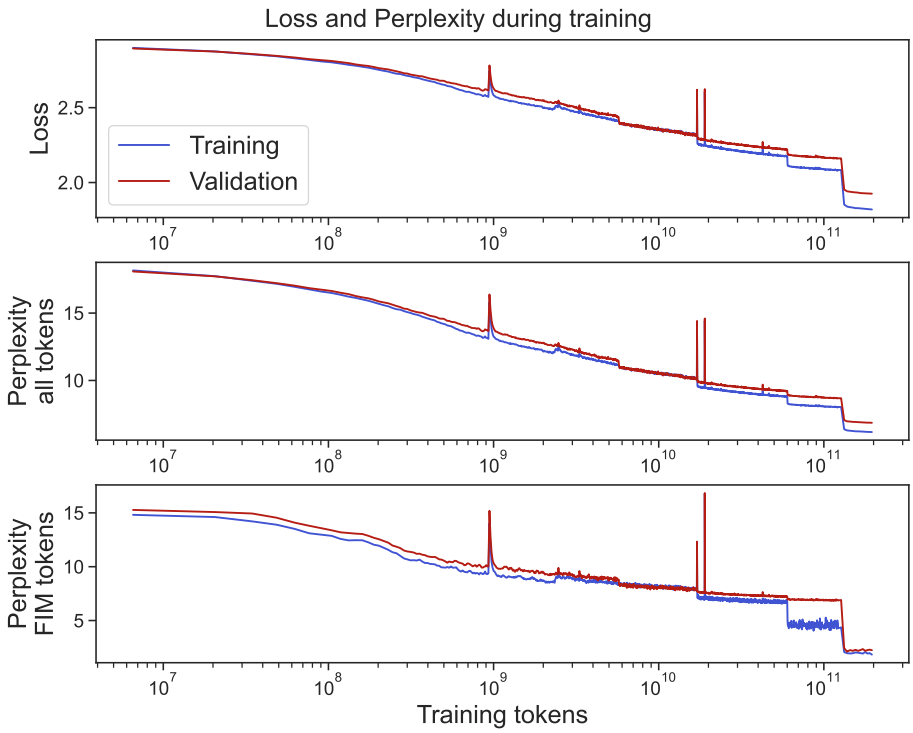

Fig. S1. Loss and perplexity during training. Cross entropy loss and perplexity computed for both the full non-masked sequences and the FIM tokens. We show them as a function of the number of tokens processed during the training of ProtMamba. They are computed on the training set and on a validation set of 192 held-out OpenProteinSet sequence clusters (see Section 2.3).

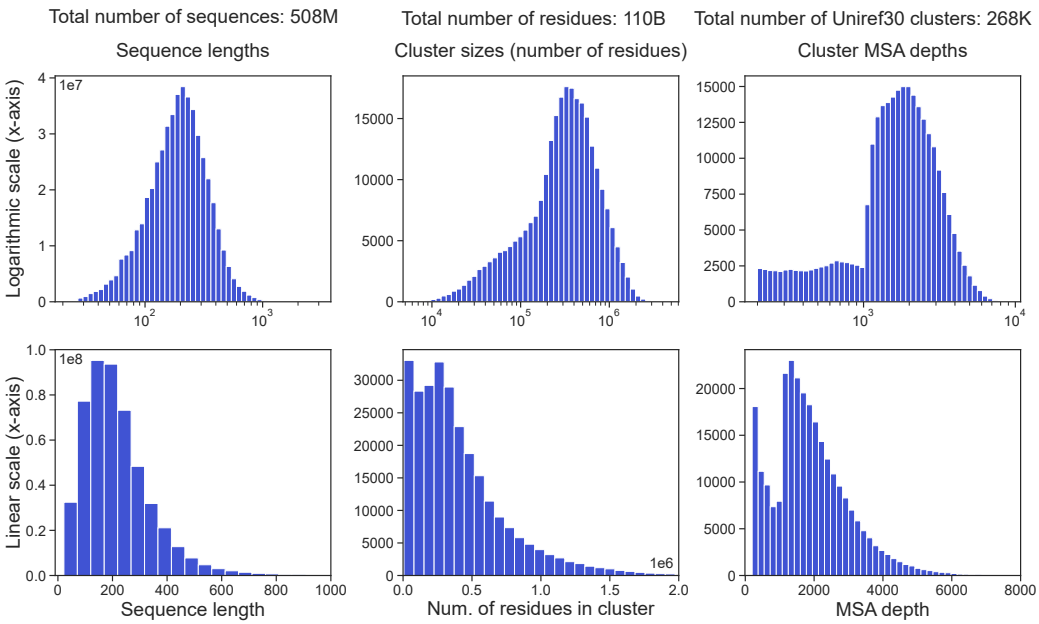

Fig. S2. Datasets statistics. We show the x axis both in log scale (first row) and in linear scale (second row) to have a better grasp of the distributions.

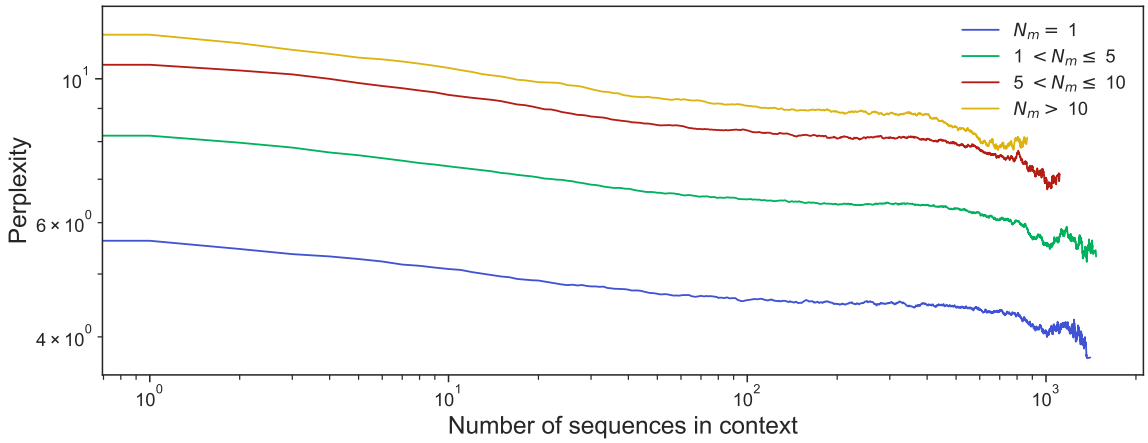

Fig. S3. Scaling of the FIM perplexity with the number of context sequences. Same as Figure 2, using logarithmic scales on both axes.

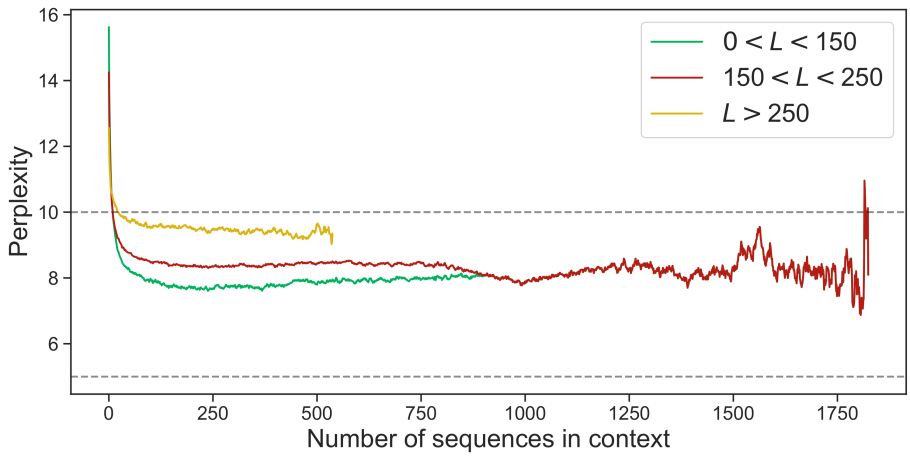

Fig. S4. Loss and perplexity of the full sequences vs. number of sequences in the context. Scaling of the per-sequence perplexity (i.e. the standard autoregressive perplexity of the full non-masked sequence) versus the number of context sequences. Results are averaged over all 500 clusters of the test set and 20 replicates for each cluster (differing by the random sampling of context sequences). Context sizes go up to 2^{17} amino acids. Sequence clusters are split according to the average length L of sequences in the cluster. We observe that clusters with shorter sequences reach lower perplexities.

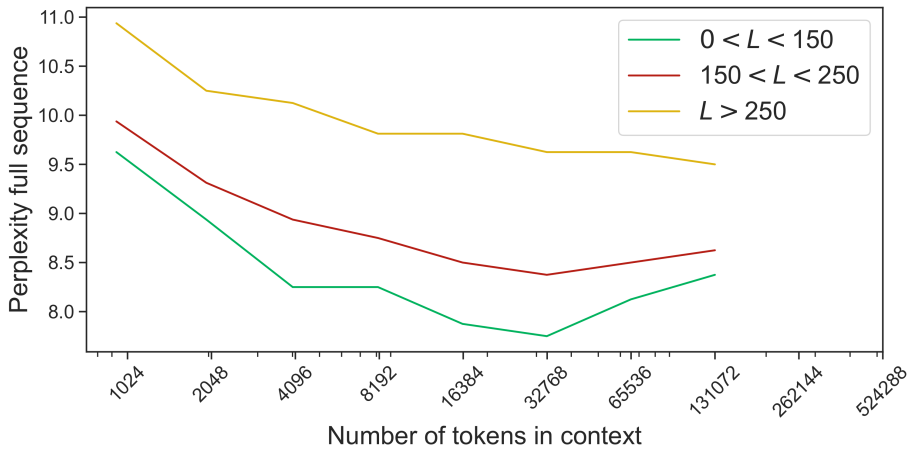

Fig. S5. Loss and perplexity of the full sequences vs. number of tokens in the context. Scaling of the per-sequence perplexity (i.e. the standard autoregressive perplexity of the full non-masked sequence) versus the size of the context (i.e. the number of preceeding tokens). Results are averaged over all 500 clusters of the test set and 20 replicates for each cluster (differing by the random sampling of context sequences). Context sizes go up to 2^{17} amino acids. Sequence clusters are split according to the average length L of sequences in the cluster. We observe that clusters with shorter sequences reach lower perplexities.

Cluster	Hamming	HMMER	pLDDT	pTM
A0A2H9MP70	0.45	−0.44	−0.77	−0.54
A0A135YUE9	0.76	−0.68	−0.61	−0.58
G4ZH78	0.23	−0.42	−0.47	−0.47
A0A1A8YWK1	0.79	−0.7	−0.84	−0.81
A0A0A0HZM8	0.46	−0.47	−0.69	−0.63
A0A091TDH7	0.31	−0.32	−0.46	−0.46
A0A2N1P554	0.18	−0.35	−0.37	−0.4
A0A1C5UJ41	0.81	−0.78	−0.81	−0.77
A0A194V424	0.66	−0.71	−0.7	−0.54
S7UZ45	0.56	−0.57	−0.74	−0.66
F2CV06	0.89	−0.85	−0.81	−0.79
A0A146ZGL6	0.72	−0.75	−0.56	−0.64
D8SD16	0.65	−0.74	−0.71	−0.61
A0A139IN77	0.18	−0.2	−0.15	−0.27
A0A1C6Q5J2	0.55	−0.22	−0.45	−0.27
I4B642	0.44	−0.52	−0.61	−0.52
A0A2X4BAY2	0.27	−0.63	−0.63	−0.54
A0A241VGM5	0.44	−0.59	−0.62	−0.55
A0A1S3G530	0.88	−0.72	−0.8	−0.74
Mean	0.54	−0.56	−0.62	−0.57

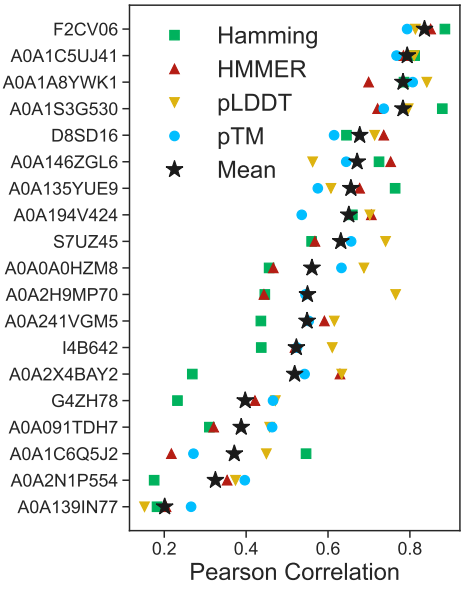

Fig. S6. Pearson correlation between ProtMamba perplexity and scores for generated sequences. For each of 19 test clusters, we used all the sequences generated by ProtMamba to compute the Pearson correlation between the model perplexity and the Hamming distance to the closest natural neighbor, the HMMER score, the pLDDT and pTM scores from ESMFold.

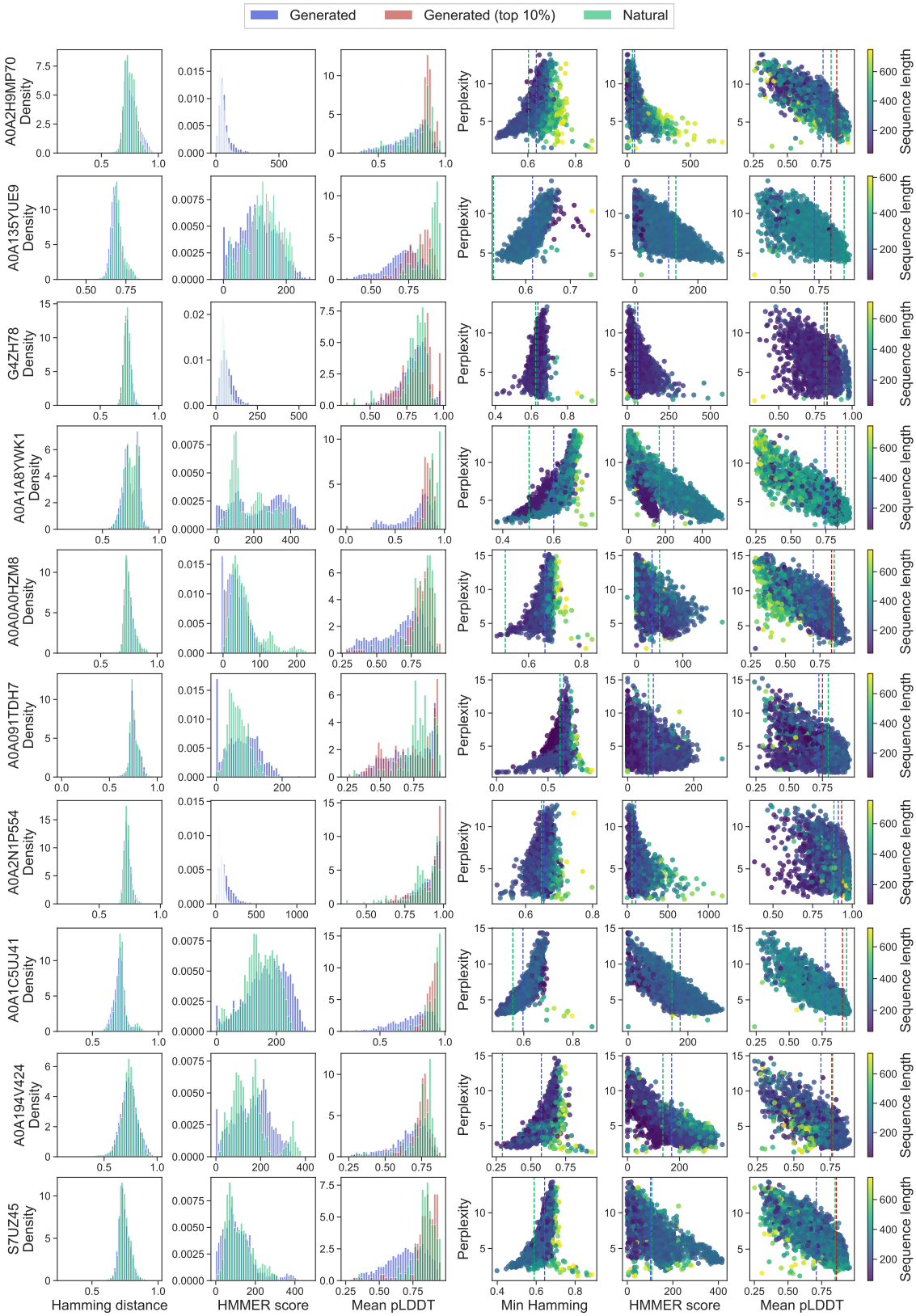

Fig. S7. Properties of generated sequences. Left panels: histograms of Hamming distances, HMMER scores and mean pLDDT scores from ESMFold of generated sequences for 10 example test clusters (10 rows). Right panels: scatter plots of ProtMamba perplexity versus the Hamming distance to the closest natural neighbor, the HMMER score and mean pLDDT score from ESMFold for all generated sequences from each of 10 example clusters (10 rows). Dashed vertical lines: median of the generated sequences (blue), median of the natural sequences (green) and pLDDT value of the reference structure of the cluster (red). The last one is shown only for the rightmost plot.

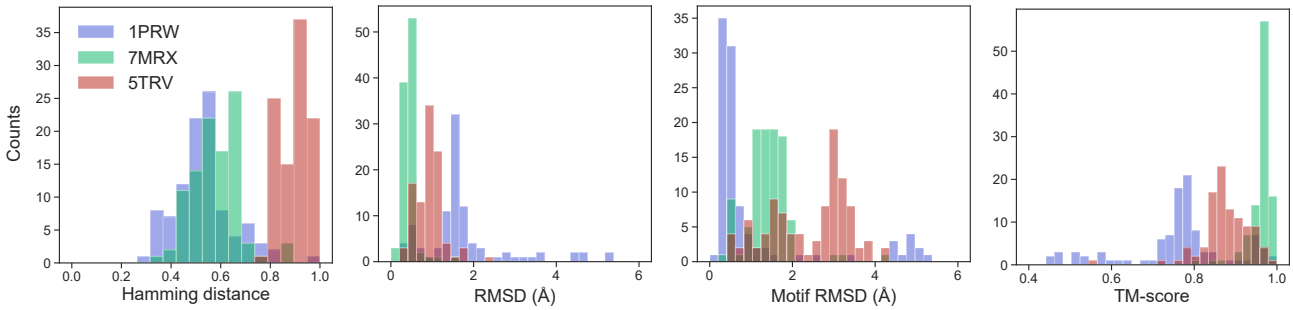

Fig. S8. Properties of inpainted functional motifs generated by ProtMamba. For each of three protein structures (same as in Figure 4), we show the distribution of four scores over 100 inpainted sequences, where the masked functional motif was generated by ProtMamba. The four scores considered are: Hamming distance, illustrating sequence divergence; RMSD between the ESMFold-predicted structure of the inpainted sequence and the original experimental structure, calculated over the whole protein; RMSD calculated over the inpainted motif; TM-score. Since the protein family associated to 1PRW includes multiple conformations (e.g. 1CLL, 1A29, 1CKK), we computed TM-score and RMSD against the two most common conformations (1PRW and 1CLL), and reported respectively the highest and lowest values. Over 90% of the generated sequences achieved a TM-score > 0.7 with at least one known conformation, indicating that ProtMamba reliably generates structurally plausible motifs even in the presence of different conformations.

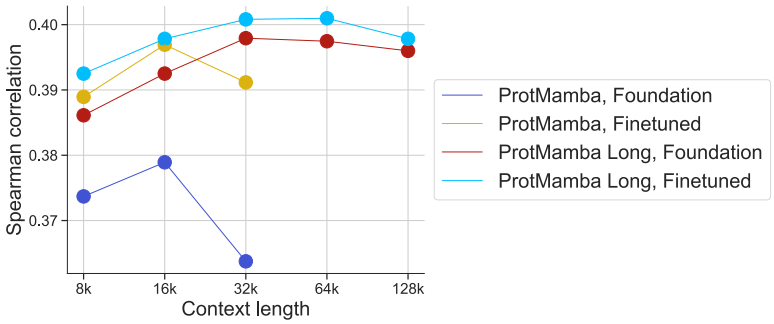

Fig. S9. Comparison of 4 ProtMamba variants on the ProteinGym benchmark. We show the predictive power for variant effect on the ProteinGym benchmark, via the Spearman correlation between predictions and experimental results, for "ProtMamba, Foundation" ($2^{15} = 32768$ tokens context seen in training), "ProtMamba, Fine-tuned" (fine-tuned on predicting only FIM tokens), ProtMamba Long, Foundation" ($2^{17} = 131072$ tokens context seen in second phase of training) and "ProtMamba Long, Fine-tuned" (fine-tuned on predicting only FIM tokens). We notice that models fine-tuned only on the FIM objective outperform the foundation models. ProtMamba Long is overall performing better than ProtMamba and its performance does not decrease as sharply as ProtMamba for longer context.

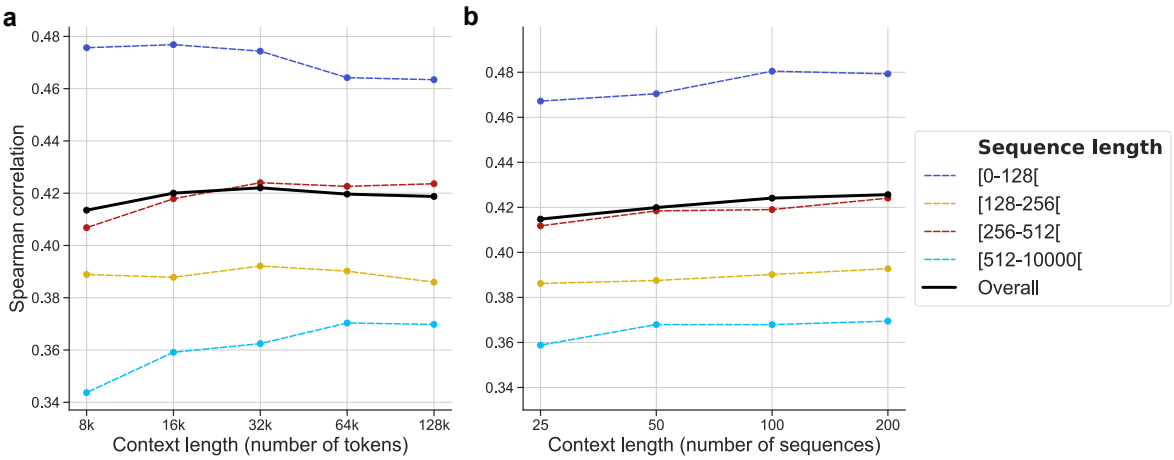

Fig. S10. Impact of context length on results on the ProteinGym benchmark. (a) We run ProtMamba Long on the ProteinGym dataset, building contexts of different sizes in terms of numbers of tokens (from 8,000 to 128,000). We see that the increase in performance is more important for long sequences, which highlights the benefit of long context to model long protein sequences. **(b)** We also run ProtMamba Long on the ProteinGym dataset, building contexts of different sizes in terms of numbers of sequences (from 25 to 200). Overall, we notice a rise in the Spearman correlation, showing that prediction benefits from longer context.

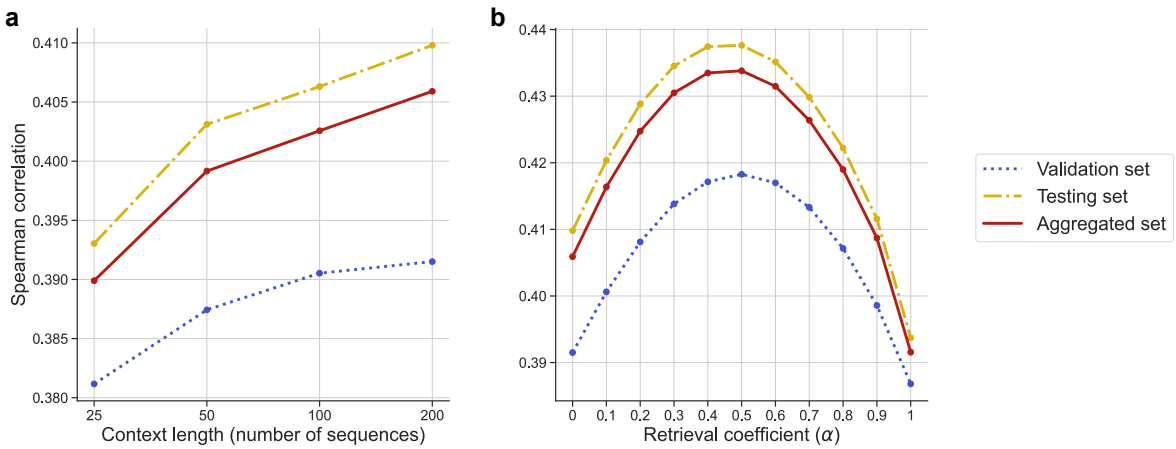

Fig. S11. Choice of context length and retrieval coefficient using a validation set. We randomly extracted a validation set of 20 datasets (see supplementary Section A) to select the best context length and retrieval coefficient. **(a)** The prediction improves with the context size in the validation set. This trend was later observed in the rest of the benchmark (testing set) too. **(b)** Retrieval requires mixing the fitness score $\mathcal{F}_m$ obtained from ProtMamba and the fitness score obtained from the independent-site model $\mathcal{F}_i$ through the retrieval fitness score $\mathcal{F}_r = \alpha\mathcal{F}_i + (1-\alpha)\mathcal{F}_m$. The best model on the validation set was obtained for a retrieval coefficient $\alpha = 0.5$, which was later verified on the rest of the dataset.

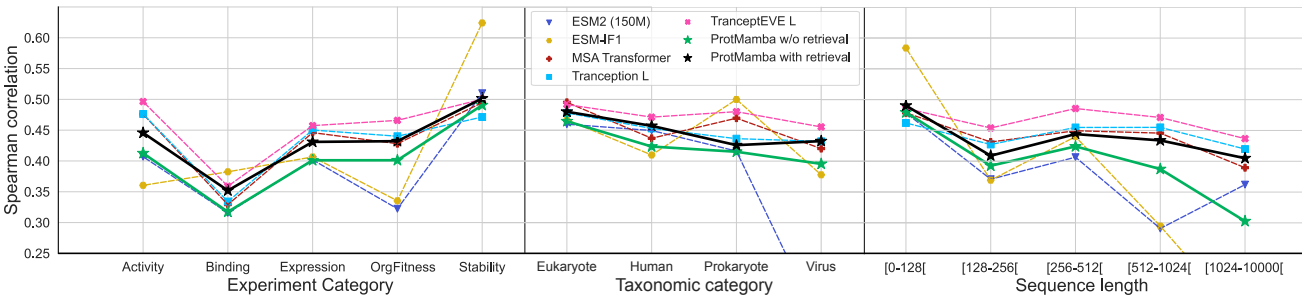

Fig. S12. Breakdown by categories of results on the ProteinGym benchmark. Results of ProtMamba Long and of existing specialized models on the ProteinGym benchmark, averaged over all datasets, are shown broken down by category of experiments (left), taxonomic category (middle) and wild-type sequence length (right). ProtMamba is fairly competitive with these models. We note that the inverse folding model ESM-IF1 outperforms sequence-based models for stability assessment, as expected (see left panel).

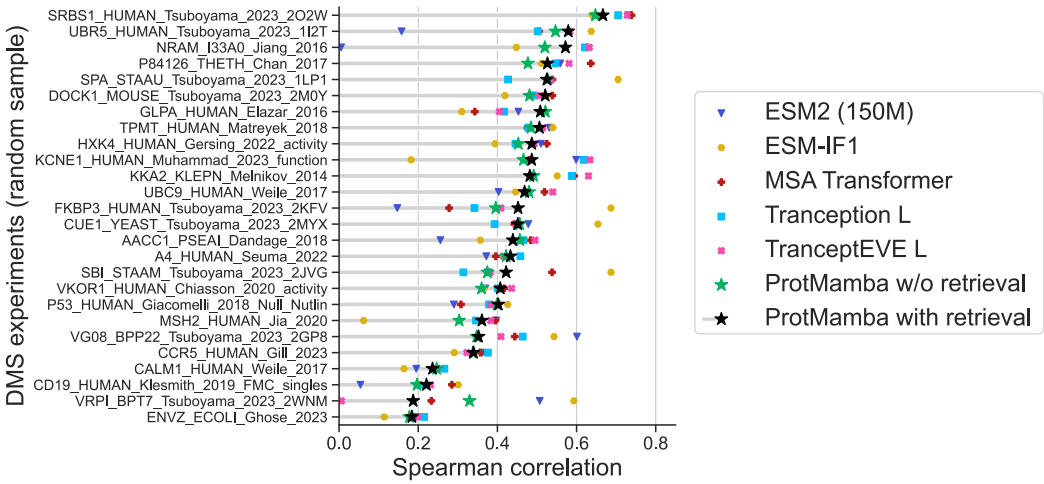

Fig. S13. Example results on the ProteinGym benchmark. Results of ProtMamba Long are shown on 25 randomly sampled deep mutational scan (DMS) experimental datasets from ProteinGym, and are compared to existing methods (see main text). The score shown is the Spearman correlation between predictions and experimental results.

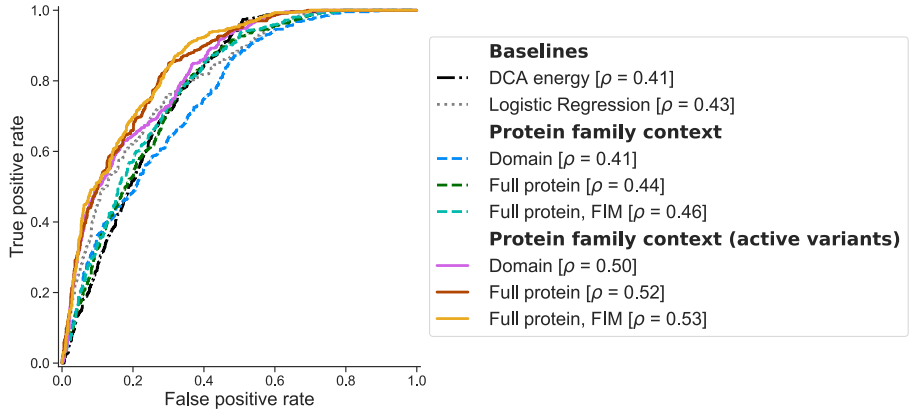

Fig. S14. Impact of various context construction methods on results on chorismate mutase activity. The ROC curve is shown for various context construction methods (see main text) for predicting active variants in the chorismate mutase dataset, and for baseline methods from [40]. Overall, we observe that restricting to active variants in context helps improving prediction quality (Spearman correlation ρ going from 0.41-0.46 to 0.50-0.53). Giving full proteins instead of restricting to the chorismate mutase domain also improves the results. Using FIM to condition the domain to score using the rest of the protein also improves performance. ProtMamba also outperforms the baselines provided in [40], namely the Potts or DCA energy and the logistic regression trained directly on amino-acid sequences.

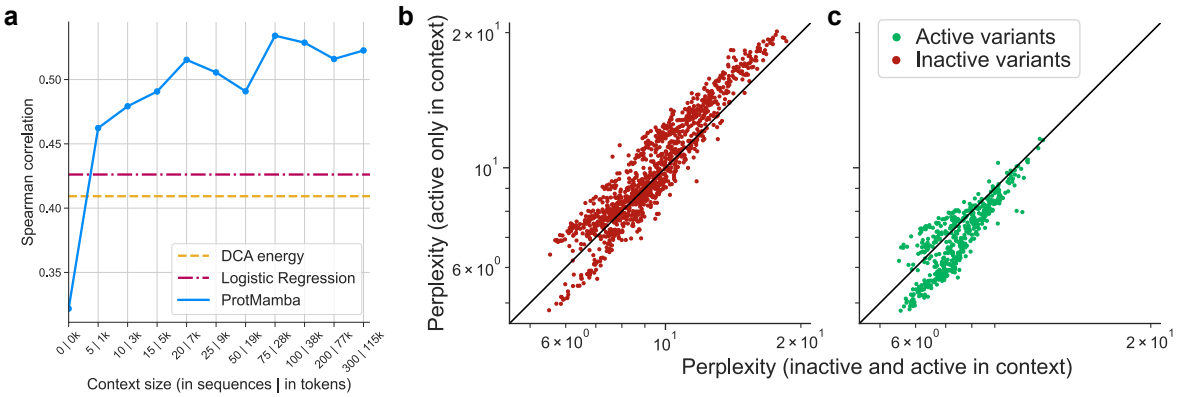

Fig. S15. (**a**) **Impact of context length on results on chorismate mutase activity.** Spearman correlation between experimental activity and predictions from ProtMamba is shown using a different number of active sequences in the context (using FIM and full active proteins as context to score variants using ProtMamba). The Spearman correlation quickly increases with the number of proteins sequences given in context, especially from 0 to 25 sequences (or 10,000 tokens) before slowly increasing with context size. (**b**) and (**c**) **Perplexity of generated variants when using only active variants in context (b) or using active and inactive variants in context (c)**. Inactive variants tend to have higher perplexity (implying lower fitness score) when the context contains only active variants (**b**) while active variants have lower perplexity (implying higher fitness score) when the context contains only active variants (**c**) .

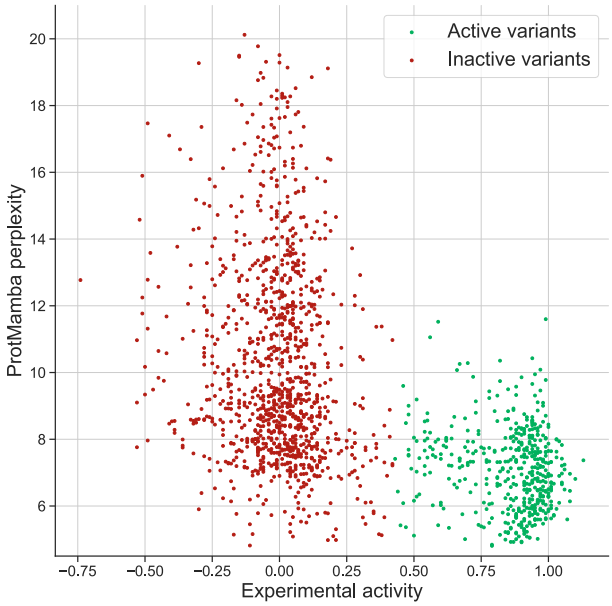

Fig. S16. ProtMamba captures chorismate mutase activity. Experimental activity of chorismate mutase enzyme variants from [40] is shown versus ProtMamba per-token perplexity, determined using FIM and full active proteins as context. The per-token perplexity is a good proxy of the activity. We obtain a Spearman correlation of 0.53 between this score and experimental activity, and it yields an AUC of 0.84 to discriminate active from inactive sequences.

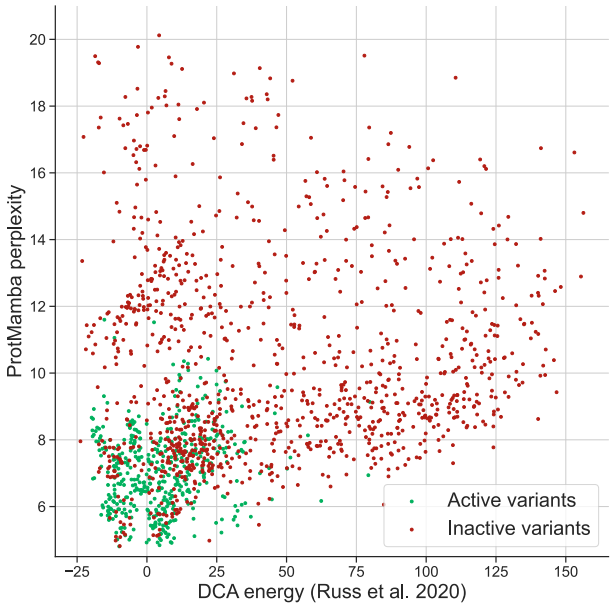

Fig. S17. ProtMamba perplexity versus DCA energy for chorismate mutase variants. ProtMamba perplexity is evaluated using full sequences, FIM and only active variants in the context, and is shown versus the Potts or DCA energy from [40]. Active variants are in green, while inactive variants are in red. We observe that most of the variants that are active have low perplexity, and that many inactive variants that were not discriminated as inactive by DCA are labelled as such by ProtMamba (bottom right part of the plot).