

## 873 A Broader Impact

874 PoET-2, a multimodal, retrieval-augmented protein model, aims to improve the prediction of mu-  
 875 tational effects and enable controllable protein design by learning from sequence, structure, and  
 876 evolutionary family context. This work can accelerate beneficial applications such as designing novel  
 877 enzymes, therapeutics, and more stable proteins. PoET-2’s enhanced data efficiency in supervised  
 878 learning may also broaden access to advanced protein engineering, especially in data-limited settings.  
 879 While PoET-2 is a foundational research tool, advanced capabilities in understanding and designing  
 880 proteins could theoretically be misused, for example, in the development of dangerous drugs. We  
 881 expect that future work will serve to address and mitigate such concerns.

## 882 B PoET-2 Architecture

883 This section elaborates on PoET-2’s architecture, supplementing the description in the main text.

884 **Notation** To refer to a specific residue in a sequence-of-sequences  $x$ , we use the same notation  
 885 as in the main text i.e. we use  $x_j^{(i)}$  to denote the  $j$ th residue of the  $i$ th sequence in  $x$ . In general,  
 886 the superscript  $^{(i)}$  is used to refer to the  $i$ th sequence. For example, in the main text, we use  $D$   
 887 to refer to the matrix of pairwise discretized  $C\alpha$  distances of a protein. Thus, in the context of a  
 888 sequence-of-sequences, the notation  $D^{(i)}$  refers to the pairwise discretized  $C\alpha$  distances of the  $i$ th  
 889 sequence in the sequence-of-sequences.

### 890 B.1 Input Embedding

891 The encoder and decoders share a common input embedding space. This space fuses representations  
 892 derived from the input sequence ( $x_{\text{seq}}$ ), the local structure backbone coordinates ( $x_{\text{atomb}}$ ), and the  
 893 pLDDT scores ( $x_{\text{plddt}}$ ). Both  $x_{\text{atomb}}$  and  $x_{\text{plddt}}$  can contain entries that are masked, for instance, due to  
 894 missing structural information, padding, or masking specified by a user. For each of these features,  
 895 a corresponding binary mask (e.g.,  $x_{\text{atomb\_mask}}$ ,  $x_{\text{plddt\_mask}}$ ) is provided, where a value of 1 indicates  
 896 an observed or valid entry, and 0 indicates a masked or invalid entry. To process these potentially  
 897 masked inputs, the algorithm first applies the respective binary mask to the feature data by setting  
 898 the values at masked positions to zero. Subsequently, the binary mask itself is concatenated as an  
 899 additional feature channel to this modified data. These augmented representations for  $x_{\text{atomb}}$  and  
 900  $x_{\text{plddt}}$  are then linearly projected into the target embedding dimension. The sequence  $x_{\text{seq}}$  is embedded  
 901 directly. Finally, these three resulting embeddings are summed to form the single continuous latent  
 902 representation (Algorithm 1).

---

#### Algorithm 1 embed\_inputs – embeds a single sequence or a sequence-of-sequences

---

**Require:**  $x_{\text{inputs}} = \{x_{\text{seq}} \in \{1..28\}^{L_x}, x_{\text{plddt}} \in [0, 100]^{L_x}, x_{\text{plddt\_mask}} \in \{0, 1\}^{L_x}, x_{\text{atomb}} \in \mathbb{R}^{L_x \times 36}, x_{\text{atomb\_mask}} \in \{0, 1\}^{L_x \times 36}\}$

- ▷ Handle masks for  $x_{\text{plddt}}$  and  $x_{\text{atomb}}$  by applying the masks and concatenating along feature dimension
  - 1:  $z_{\text{plddt}} = \text{Concat}((x_{\text{plddt}} * x_{\text{plddt\_mask}}, x_{\text{plddt\_mask}}), \text{dim}=1)$  ▷  $[0, 100]^{L_x \times 2}$
  - 2:  $z_{\text{atomb}} = \text{Concat}((x_{\text{atomb}} * x_{\text{atomb\_mask}}, x_{\text{atomb\_mask}}), \text{dim}=1)$  ▷  $\mathbb{R}^{L_x \times 72}$
  - ▷ Embed inputs individually
  - 3:  $z_{\text{seq}} = \text{Embed}(x_{\text{seq}})$  ▷  $\mathbb{R}^{L_x \times d}$
  - 4:  $z_{\text{plddt}} = \text{Linear}(z_{\text{plddt}})$  ▷  $\mathbb{R}^{L_x \times d}$
  - 5:  $z_{\text{atomb}} = \text{Linear}(z_{\text{atomb}})$  ▷  $\mathbb{R}^{L_x \times d}$
  - 6:  $z_{\text{seqid}} = \text{Linear}(z_{\text{seqid}})$  ▷  $\mathbb{R}^{L_x \times d}$
  - 7: **return**  $z_{\text{seq}} + z_{\text{plddt}} + z_{\text{atomb}} + z_{\text{seqid}}$

---

### 903 B.2 Structure-based Attention Bias

904 See the main text (§3.2.3) for a description of the structure-based attention bias. Figure 3 visualizes  
 905 the application of the structure-based attention bias.

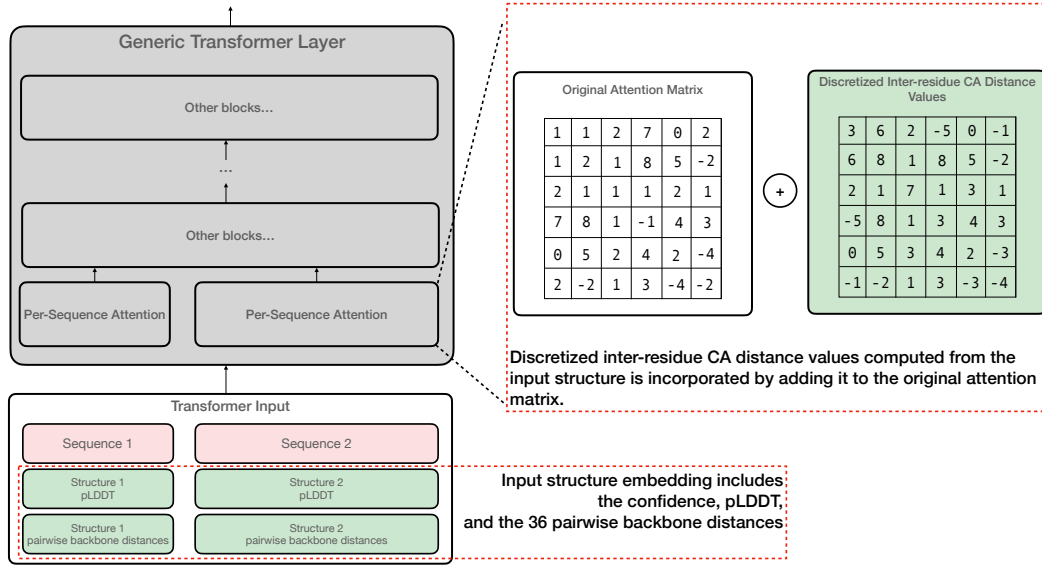


Figure 3: Structure-based attention bias

### B.3 Encoder Architecture

After transforming the raw inputs  $x_{\text{inputs}}$  into embeddings, the encoder (Algorithm 3) further transforms the embeddings by applying  $n_{\text{layers}}$  of protein order equivariant encoder layers (Algorithm 2). The encoder has two outputs: per residue embeddings of the prompt,  $h_{\text{encoder}}$ , that are used in the decoders, and per residue sequence logits,  $z_{\text{seq}}$ , that are used to compute the encoder MLM loss  $\mathcal{L}_{\text{MLM encoder}}$ .

### B.4 Decoder Architecture

The decoders each decode a single sequence  $y$ , conditioned on (1) the prompt embedding  $h_{\text{encoder}}$ , and (2) an optional query index  $q$  indicating which sequence in the prompt to use as the query, if any.

The transformations performed in each decoder are detailed in Algorithm 5; the CLM decoder and MLM decoder only differ in that the former uses a causal mask in the attention operations, and the latter does not. First, the input  $y$  is embedded. Next, if a query is present, the embedding of  $y$  and the embedding of the query (produced by the encoder) are averaged. Then,  $n_{\text{layers}}$  of decoder layers (Algorithm 4) that are equivariant to the protein order of the prompt are applied. The decoders each have a single output,  $z_{\text{seq}}$ , that is used to compute the corresponding loss ( $\mathcal{L}_{\text{CLM decoder}}$  for the CLM decoder and  $\mathcal{L}_{\text{MLM decoder}}$  for the MLM decoder).

---

**Algorithm 2** `encoder_layer` (for brevity, this algorithm describes only a single attention head, but can be extended to multiple attention heads in the normal fashion)

---

**Require:**  $\forall i \in \{1..N\}, j \in \{1..L_{x_i}\}$

- prompt embedding  $h_j^{(i)} \in \mathbb{R}^d$
- pairwise discretized C $\alpha$  distances  $D_{m,n}^{(i)} \in \{1..128\}, m \in \{1..L_{x_i}\}, n \in \{1..L_{x_i}\}$

▷ First, apply self-attention with structure-based attention bias to each sequence individually

- 1:  $f = \text{RMSNorm}(h)$  ▷  $\mathbb{R}^{L_x \times d}$
- 2:  $q_j^{(i)} = \text{RoPE}(\text{Linear}(f_j^{(i)}), j) \forall i, j$  ▷  $\mathbb{R}^d$
- 3:  $k_j^{(i)} = \text{RoPE}(\text{Linear}(f_j^{(i)}), j) \forall i, j$  ▷  $\mathbb{R}^d$
- 4:  $v = \text{Linear}(f)$  ▷  $\mathbb{R}^{L_x \times d}$
- ▷ Compute attention score with structure-based bias
- 5:  $A_{m,n}^{(i)} = q_m^{(i)T} k_n^{(i)} + \text{structure\_bias}(D_{m,n}^{(i)})$  ▷  $\mathbb{R}$
- 6:  $f^{(i)} = f^{(i)} + \text{softmax}(\frac{A^{(i)}}{\sqrt{d}})v^{(i)} \forall i$  ▷  $\mathbb{R}^{L_{x(i)} \times d}$
- ▷ Next, apply self-attention to all sequences together
- 7:  $g = \text{RMSNorm}(f)$  ▷  $\mathbb{R}^{L_x \times d}$
- 8:  $q_j^{(i)} = \text{RoPE}(\text{Linear}(g_j^{(i)}), j) \forall i, j$  ▷  $\mathbb{R}^d$
- 9:  $k_j^{(i)} = \text{RoPE}(\text{Linear}(g_j^{(i)}), j) \forall i, j$  ▷  $\mathbb{R}^d$
- 10:  $v = \text{Linear}(g)$  ▷  $\mathbb{R}^{L_x \times d}$
- 11:  $g = g + \text{Attention}(q, k, v)$  ▷  $\mathbb{R}^{L_x \times d}$
- ▷ Finally, the feedforward layer
- 12:  $g' = \text{RMSNorm}(g)$  ▷  $\mathbb{R}^{L_x \times d}$
- 13:  $h' = \text{SwiGLU}(g')$  ▷  $\mathbb{R}^{L_x \times \frac{8}{3}d}$
- 14:  $g' = g + \text{Linear}(h')$  ▷  $\mathbb{R}^{L_x \times d}$
- 15: **return**  $g'$

---



---

**Algorithm 3** `encoder` – encodes a prompt  $x$  composed of a sequence-of-sequences

---

**Require:**  $\forall i \in \{1..N\}$

- $x_{\text{inputs}} = \{x_{\text{seq}}, x_{\text{plddt}}, x_{\text{atomb}}\}$
- pairwise discretized C $\alpha$  distances  $D_{m,n}^{(i)} \in \{1..128\}, m \in \{1..L_{x_i}\}, n \in \{1..L_{x_i}\}$

- 1:  $h_{\text{encoder}} = \text{embed\_inputs}(x_{\text{inputs}})$  ▷  $\mathbb{R}^{L_x \times d}$
- 2: **for**  $l \in 1..n_{\text{layers}}$  **do**
- 3:      $h_{\text{encoder}} = \text{encoder\_layer}(h_{\text{encoder}}, D)$
- 4: **end for**
- 5:  $z_{\text{seq}} = \text{Linear}(h)$  ▷  $\mathbb{R}^{L_x \times 28}$
- 6: **return** prompt embedding  $h_{\text{encoder}}$ , sequence logits  $z_{\text{seq}}$

---

---

**Algorithm 4** decoder\_layer (for brevity, this algorithm describes only a single attention head, but can be extended to multiple attention heads in the normal fashion)

---

**Require:**

- decoder type  $T \in \{\text{CLM}, \text{MLM}\}$
- encoder prompt embeddings  $h_{\text{encoder},j}^{(i)} \in \mathbb{R}^d, i \in \{1..N\}, j \in \{1..L_{x_i}\}$
- decoder sequence embedding  $h_{\text{decoder},i} \in \mathbb{R}^d, i \in \{1..L_y\}$
- pairwise discretized C $\alpha$  distances  $D_{m,n} \in \{1..128\}, m \in \{1..L_y\}, n \in \{1..L_y\}$

---

▷ First, apply self-attention with structure-based attention bias

```

1:  $f = \text{RMSNorm}(h_{\text{decoder}})$  ▷  $\mathbb{R}^{L_y \times d}$ 
2:  $q_i = \text{RoPE}(\text{Linear}(f_i), i) \forall i$  ▷  $\mathbb{R}^d$ 
3:  $k_i = \text{RoPE}(\text{Linear}(f_i), i) \forall i$  ▷  $\mathbb{R}^d$ 
4:  $v = \text{Linear}(f)$  ▷  $\mathbb{R}^{L_y \times d}$ 
   ▷ Compute attention score with structure-based bias
5:  $A_{m,n} = q_m^T k_n + \text{structure\_bias}(D_{m,n})$  ▷  $\mathbb{R}$ 
6: if  $T == \text{CLM}$  then  $A = A + \text{CausalMask}(L_y)$  ▷  $\mathbb{R}^{L_y \times L_y}$ 
7: end if
8:  $f = f + \text{softmax}(\frac{A}{\sqrt{d}})v$  ▷  $\mathbb{R}^{L_y \times d}$ 
   ▷ Next, apply cross-attention to prompt embeddings
9:  $g = \text{RMSNorm}(f)$  ▷  $\mathbb{R}^{L_y \times d}$ 
10:  $q_i = \text{RoPE}(\text{Linear}(g_i), i) \forall i$  ▷  $\mathbb{R}^d$ 
11:  $k_j^{(i)} = \text{RoPE}(\text{Linear}(h_{\text{encoder},j}^{(i)}), j) \forall i, j$  ▷  $\mathbb{R}^d$ 
12:  $v = \text{Linear}(h_{\text{encoder}})$  ▷  $\mathbb{R}^{L_y \times d}$ 
13:  $g = g + \text{Attention}(q, k, v)$  ▷  $\mathbb{R}^{L_y \times d}$ 
   ▷ Finally, the feedforward layer
14:  $g' = \text{RMSNorm}(g)$  ▷  $\mathbb{R}^{L_y \times d}$ 
15:  $h'_{\text{decoder}} = \text{SwiGLU}(g')$  ▷  $\mathbb{R}^{L_y \times \frac{8}{3}d}$ 
16:  $g' = g + \text{Linear}(h'_{\text{decoder}})$  ▷  $\mathbb{R}^{L_y \times d}$ 
17: return  $g'$ 

```

---



---

**Algorithm 5** decoder – decodes a single sequence  $y$  conditioned on prompt embeddings  $h_{\text{encoder}}$

---

**Require:**

- decoder type  $T \in \{\text{CLM}, \text{MLM}\}$
- optional query index  $q \in \{0..N\}$
- encoder prompt embeddings  $h_{\text{encoder},j}^{(i)} \in \mathbb{R}^d, i \in \{1..N\}, j \in \{1..L_{x_i}\}$
- decoder inputs  $y_{\text{inputs}} = \{y_{\text{seq}}, y_{\text{plddt}}, y_{\text{atomb}}\}$
- pairwise discretized C $\alpha$  distances  $D_{m,n} \in \{1..128\}, m \in \{1..L_y\}, n \in \{1..L_y\}$

---

```

1:  $h_{\text{decoder}} = \text{embed\_inputs}(y_{\text{inputs}})$  ▷  $\mathbb{R}^{L_y \times d}$ 
   ▷ Embed the query if there is one
2: if  $q \neq 0$  then
3:    $h_{\text{decoder},i} = \frac{1}{2}(h_{\text{decoder},i} + h_{\text{encoder},i}^{(q)}) \forall i$  ▷  $\mathbb{R}^d$ 
4: end if
5: for  $l \in 1..n_{\text{layers}}$  do
6:    $h_{\text{decoder}} = \text{decoder\_layer}(T, h_{\text{encoder}}, h_{\text{decoder}}, D)$  ▷  $\mathbb{R}^{L_y \times d}$ 
7: end for
8:  $z_{\text{seq}} = \text{Linear}(h_{\text{decoder}})$  ▷  $\mathbb{R}^{L_y \times 28}$ 
9: return sequence logits  $z_{\text{seq}}$ 

```

---

## 922 B.5 Miscellaneous

923 This section details additional aspects of PoET-2’s architecture that are related to capabilities that are  
924 *not* utilized in this paper’s experiments.

### 925 B.5.1 3Di Token Prediction

926 In addition to predicting each protein’s amino acid sequence, PoET-2 is also trained to predict each  
927 protein’s 3Di structure token sequence [40] using the cross entropy loss. The 3Di structure tokens  
928 are only predicted when the predicted pLDDT of the residue is at least 70. The amino acid and 3Di  
929 structure token losses have equal weight i.e. the total loss is simply the sum of the two losses.

### 930 B.5.2 Conditioning on target homology

931 PoET-2 is also trained to generate sequences that must be within a specified sequence identity range  
932 of a protein in the prompt, referred to as the query protein. The query can be any protein in the  
933 prompt whose sequence is completely known (i.e. contains no unknown or masked amino acids).  
934 This generation mode is called "target homology". When using this generation mode, the structure of  
935 the generated protein does not have to contain any structural elements specified in the query protein.

936 The target homology generation mode is implemented with two modifications to the architecture  
937 discussed so far:

- 938 1. The input embedding is augmented with another feature,  $x_{\text{seqid}}$ . This feature represents  
939 the sequence identity range as two values  $\in [0, 1]$  indicating the range’s lower and upper  
940 bounds, and is repeated across all residues. Thus, it has shape  $L \times 2$ . If the target homology  
941 generation mode is inactive (e.g. as in the encoder) or not being used in the decoder, these  
942 two sequence identity values are both set to 0. To prepare  $x_{\text{seqid}}$  for input embedding, it is  
943 processed similarly to other features like  $x_{\text{plddt}}$  and  $x_{\text{atomb}}$ : any values at positions intended  
944 to be masked are set to zero, and then a corresponding binary mask (1 for observed/valid,  
945 0 for masked/invalid) is concatenated as a third channel to these two sequence identity  
946 values. This augmented 3-channel tensor for  $x_{\text{seqid}}$  is then linearly projected to the model’s  
947 hidden dimension and subsequently summed with the embeddings of other input features,  
948 as detailed in Algorithm [1].
- 949 2. Since the generated sequence does not necessarily have the same length as the query, we  
950 cannot simply combine the embeddings of the generated sequence and the embeddings  
951 of the query by summing the embeddings of all corresponding residues as in Line 3 of  
952 Algorithm [5] since the correspondence is unknown. Instead, we sum the embedding of  
953 each residue of the generated sequence with the embedding of the first residue of the query  
954 sequence. That is, when the target homology generation mode is used, we replace Line 3 of  
955 Algorithm [5] with  $h_{\text{decoder},i} = \frac{1}{2}(h_{\text{decoder},i} + h_{\text{encoder},1}^{(q)})\forall i$ .

### 956 B.5.3 Decoding queries of unknown length

957 Lastly, PoET-2 is trained to decode query sequences with contiguous segments of unknown length.  
958 These segments are represented with the gap token (-) mentioned in §3.2.1 the gap token indicates  
959 zero or more residues with unknown identity. For example, the query sequence \$MK-IP\* indicates  
960 that PoET-2 must generate a sequence that starts with the two amino acids M and K, then has 0 of  
961 more amino acids, and then ends with the amino acids I and P.

962 In order to decode such sequences, PoET-2 uses a special decoding scheme called the "insertion  
963 decoding scheme". The purpose of this decoding scheme is to align the residues of the query sequence  
964 and the generated sequence so that the embeddings of their corresponding residues can be summed in  
965 Line 3 of Algorithm [5].

966 In the standard decoding scheme, the alignment between the query sequence and the sequence being  
967 generated by the model may be ambiguous when gap tokens are present in the query. For example,  
968 suppose that the query sequence is \$-A. In a normal decoding scheme, if the model predicts that  
969 the token following the start token is the token A, it is ambiguous if that token should be aligned  
970 with the gap token to indicate an insertion, or aligned with the token A to indicate that there are  
971 no insertions. In order to address this ambiguity, we train the decoders to instead output the gap

972 token when generating a token that is unmasked in the query sequence. Continuing the example, if  
 973 the model wants to generate the token A as part of an insertion aligned with the gap token in the  
 974 query sequence, then the model should simply output the token A as usual. However, if the model  
 975 wants to generate the token A and have it be aligned with the token A at the third position in the  
 976 query sequence, then the model should output the gap token. In this case, the gap token in the query  
 977 sequence represents no insertions.

978 The insertion decoding scheme is visualized in Figure 4. Using the alignment provided by the insertion  
 979 decoding scheme, Line 3 of Algorithm 5 can then be modified to be  $h_{\text{decoder},i} = \frac{1}{2}(h_{\text{decoder},i} +$   
 980  $h_{\text{encoder,alignment}_i}^{(q)}) \forall i$ , where  $\text{alignment}_i$  indicates the index of the residue of the query that the  $i$ th  
 981 residue of the generated sequence is aligned to.

982 Although the insertion decoding scheme is primarily required for the CLM decoder, we apply the  
 983 insertion decoding scheme to the MLM decoder as well by adjusting the outputs tokens in a similar  
 984 manner i.e. since the MLM decoder is trained to unmask the token at the current position (as opposed  
 985 to next token prediction for the CLM decoder), if the token at the current position is unmasked, the  
 986 MLM decoder is trained to output the gap token.

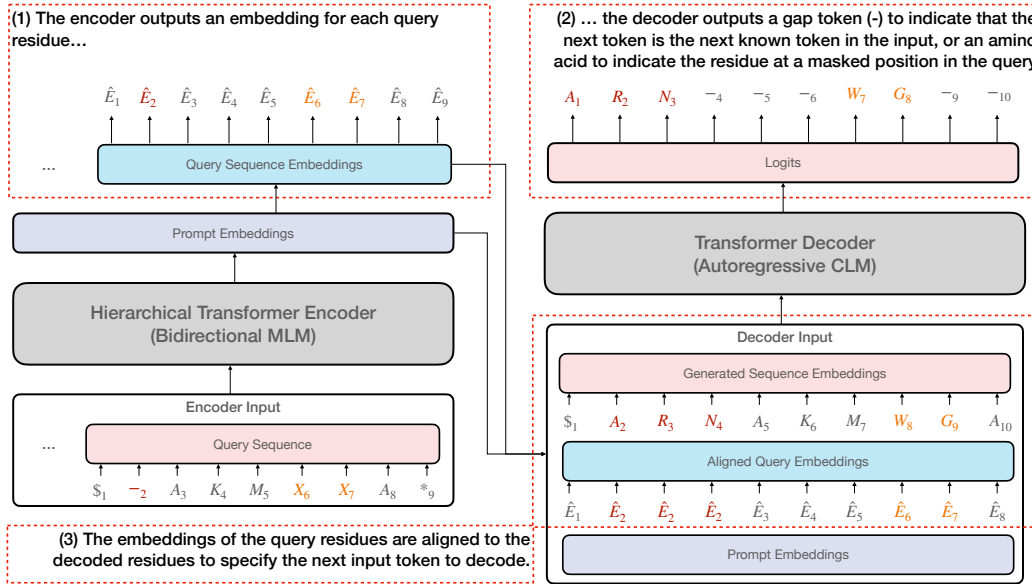


Figure 4: Visualization of insertion decoding scheme.

## 987 C PoET-2 Training Details

### 988 C.1 Training data

989 **Sequence data** PoET-2 is trained on sets of homologous sequences. Sets of homologous sequences  
 990 are found and sampled using the same procedure used by PoET-1 [11]. Summarizing briefly, the sets  
 991 of homologous sequences are found by using Diamond [38] to search UniRef50 in an all against all  
 992 search using the following command:

```
993 diamond blastp -q uniref50.fasta -d diamond/uniref50 -f 6 -header -k 200000
994 -max-hsps 1 -e 0.001 -p 96 -o output.tab
```

995 Sets of homologous sequences are sampled with weight proportional to the inverse of the size of the  
996 set.

997 The main differences between the training data for PoET-2 and PoET-1 are as follows:

- 998 • UniRef Version 2304 is used instead of UniRef Version 2103.
- 999 • All sets of homologous sequences are used, rather than only sets with at least 10 members.

1000 As a result of these differences, PoET-2 is trained on 62 million sets, as opposed to 29 million sets for  
1001 PoET-1.

1002 **Structure data** PoET-2 is trained only on predicted structures from AFDB [30, 31], and no  
1003 experimentally solved structures. Sequences in the training data are associated with structures in  
1004 AFDB using the UniRef100 sequence identifier. When the structure of a protein is used as input to  
1005 PoET-2, if there is a conflict between the sequence in UniRef and AFDB (e.g. due to changes in  
1006 UniRef), the sequence in AFDB is used. On the order of approximately half of sequences in the  
1007 training data can be associated with a predicted structure using this methodology.

## 1008 C.2 Noise schedule

1009 For sequence inputs,

- 1010 • Context sequence tokens are masked with a random masking rate chosen uniformly from  
1011 0%-30%.
- 1012 • Query sequence tokens are randomly masked with a random masking rate chosen uniformly  
1013 from 0%-100%.
- 1014 • Decoder sequence tokens are masked with a random masking rate chosen uniformly from  
1015 0%-30%.

1016 For structure inputs, the pLDDT and atomic backbone coordinates of the N, C $\alpha$ , and C atoms are  
1017 masked with a random masking rate chosen uniformly from 0%-100%.

1018 For both sequence and structure inputs, with probability

- 1019 • 50%, masking is performed randomly per residue.
- 1020 • 25%, random contiguous spans of length L are masked, where L is drawn from the distribu-  
1021 tion  $\text{Poisson}(3) + 1$ .
- 1022 • 25%, N random contiguous spans are masked, where N is drawn randomly from  
1023  $\text{Poisson}(2.5)$  half the time and from  $\text{Poisson}(13)$  the other half of the time.

## 1024 C.3 Optimizer and learning rate schedule

1025 PoET-2 is trained with the same optimizer and learning rate schedule as PoET-1 [1]. Namely, the  
1026 optimizer is Adafactor [41], and the learning rate schedule consists of a linear warmup over the first  
1027 4000 steps to a peak learning rate of  $1e-2$ , and then a square root decay over the remaining training  
1028 steps.

## 1029 C.4 Compute requirements

1030 PoET-2 is trained for 3 million steps on 8 x A100 GPUs with 40GB VRAM each. A batch size of  
1031 45056 tokens is used per GPU with gradient accumulation over two steps, for an effective batch size  
1032 of 90112 tokens per GPU. The total training time on this hardware is approximately 2.5 months.

## D Zero-shot variant effect prediction

### D.1 Prompt engineering

Recall that the prompt consists of two optional components, a context containing proteins from the protein family of interest, and a query containing explicit sequence and/or structure constraints. Our goal in prompt engineering is to design prompts that enable us to accurately predict the properties of variants of a WT sequence.

To determine the set of proteins to use in the context, we use the same method as PoET-1 [1]. We first identify proteins in the protein family by searching for sequence homologs of WT in UniRef100 [37] using the ColabFold MSA protocol [39]. We then select the proteins to include in the context of a prompt by sampling a representative subset of the sequence homologs using the method from Hopf et. al [42].

Building off of this approach, we employ two prompt engineering strategies to improve predictions:

- Following the approach of PoET-1 [1], we ensemble over different prompts, where the context of each prompt contains a different subsample of sequence homologs. The ensemble prediction is simply the average of the individual predictions. Furthermore, for each context, we use different values for the context length (i.e. number of tokens or amino acids in the context) and maximum similarity of a sequence in the context to the WT sequence. The exact values of these parameters used in the ensemble for DMS and clinical datasets are specified in the sections below.
  - Utilizing PoET-2’s multimodal capabilities, we explore two methods for incorporating structure in the prompt. The first method incorporates structure in the context by associating sequences in the context with their predicted structure in AFDB [30][31], if the sequence exists in AFDB. The second method incorporates structure by adding a query to the prompt that contains the structure (but not the sequence) of WT. The use of this "inverse-folding" query instructs PoET-2 to score the likelihood that a variant sequence will fold into the same structure as WT. Although it is not necessarily desirable for a variant to adopt the same structure as WT, the inverse-folding likelihood has been shown to be predictive of protein fitness, particularly for stability related properties [5].
- Not all methods of incorporating structure in the prompt are always helpful; the best method for doing so on the ProteinGym DMS and clinical substitutions benchmarks are ablated and identified in the following sections. Also, note that the query-based approach is not used for indel variants because indel variants have different lengths from WT and thus cannot adopt the same tertiary structure as WT.

#### D.1.1 Deep mutational scanning datasets

**Ensembling over context length and maximum similarity** Following PoET-1 [1], we ensemble over all combinations of values for context length  $\in \{6144, 12288, 24576\}$  and maximum similarity  $\in \{1.0, 0.95, 0.90, 0.70, 0.50\}$ , resulting in 15 combinations in total.

**Incorporating structure in the prompt** Table 5 shows the performance of various strategies for incorporating or not incorporating structure in the prompt, and the performance of ensembling different strategies. First, we analyze the effect of different strategies on the substitutions benchmark. When not incorporating structure at all (Strategy A), thus using the same prompting strategy as PoET-1, PoET-2 performs marginally better than PoET-1 ( $\Delta\rho = 0.005$ ; PoET-1 reported in Table 1). Both including the structure in the context (Strategy B), and in the query (Strategy C) improves performance, with the latter strategy offering a larger improvement. Interestingly, combining the two approaches (Strategy D) performs only about the same or slightly worse than Strategy C ( $\Delta\rho = -0.002$ ).

Strategy I, which ensembles all of the above strategies (A-D), improves performance further by  $\Delta\rho = 0.009$  vs Strategy C, the best individual strategy. However, we find that Strategy H, which only ensembles Strategies B and D and excludes the strategies that include only sequence in the context, performs similarly to Strategy I, with negligible performance loss. Therefore, we recommend the use of Strategy H, and use this strategy in performance comparisons with other models.



Table 5: Performance (Spearman’s  $\rho$ ) of different strategies for including structure in the prompt on the zero-shot DMS benchmarks.

Strategy	Prompt		Substitutions	Indels
	Context Modalities	Query		
A	Sequence	None	0.47534	0.55589
B	Sequence and Structure	None	0.48374	<b>0.56666</b>
C	Sequence	Structure of WT	<b>0.49128</b>	N/A
D	Sequence and Structure	Structure of WT	0.48927	N/A
E	Ensemble of A and B (Different contexts, No query)		0.48260	<b>0.56606</b>
F	Ensemble of C and D (Different contexts, With query)		0.49256	N/A
G	Ensemble of A and C (Sequence only context, Different queries)		0.49632	N/A
H	Ensemble of B and D (Sequence and structure context, Different queries)		0.49987	N/A
I	Ensemble of A, B, C, and D		<b>0.49989</b>	N/A

For indel variants, we observe a small improvement in performance by incorporating structure in the context (Strategy B) versus not (Strategy A). There is little or no benefit to ensembling these two strategies (Strategy E). Since a query cannot be used, the other strategies are not applicable. Therefore, we employ Strategy B when comparing PoET-2 to other models.

#### D.1.2 Clinical datasets

**Ensembling over context length and maximum similarity** Following PoET-1 [43], we use a context length of 49152 and ensemble over different values for maximum similarity  $\in \{1.0, 0.95, 0.90, 0.70, 0.50\}$ , resulting in 5 combinations in total. Note that the ensembling parameters for clinical datasets is less well studied than for DMS datasets, and there are likely better parameters for ensembling.

Table 6: Performance (AUROC) of different strategies for including structure in the prompt on the zero-shot clinical benchmarks.

Strategy	Prompt		Substitutions	Indels
	Context Modalities	Query		
A	Sequence	None	0.92544	0.94826
B	Sequence and Structure	None	<b>0.92624</b>	<b>0.95278</b>
C	Sequence	Structure of WT	0.91505	N/A
D	Sequence and Structure	Structure of WT	0.91292	N/A
E	Ensemble of A and B (Different contexts, No query)		<b>0.92789</b>	<b>0.95179</b>
F	Ensemble of C and D (Different contexts, With query)		0.91599	N/A
G	Ensemble of A and C (Sequence only context, Different queries)		0.92481	N/A
H	Ensemble of B and D (Sequence and structure context, Different queries)		0.92481	N/A
I	Ensemble of A, B, C, and D		0.92572	N/A

**Incorporating structure in the prompt** Table 6 shows the performance of various strategies for incorporating or not incorporating structure in the prompt, and the performance of ensembling

different strategies. First, we analyze the effect of different strategies on the substitutions benchmark. Incorporating structure in the context (Strategy B) offers a very minor and not statistically significant improvement versus not incorporating structure in the context (Strategy A). Incorporating structure via the query (Strategy C), however, has a negative effect. Therefore, we do not consider strategies that use a query further.

Ensembling Strategies A and B (Strategy E) has a small positive effect over not ensembling, although due to the small effect size, it is unclear if the effect is simply due to ensembling more prompts, or due to ensembling different prompt strategies. Nevertheless, since Strategy E performs best, we use it in comparisons with other models.

For indel variants, we observe similar trends among applicable strategies (those not using a query), with some improvement observed for incorporating structure in the context, but variations in performance being fairly minor. Although Strategy E slightly underperforms Strategy B by a non-statistically significant amount, we employ Strategy E in comparisons with other models for consistency with the strategy used for the substitutions benchmark.

## D.2 Length adjusted log likelihood (ratio)

Sequence likelihoods from autoregressive models trained with next-token prediction losses and teacher forcing can exhibit miscalibrated stop token probabilities that bias them towards shorter sequences. This likely arises because the loss function only operates at the token level – it does not strongly penalize an early stop token as long as the early stop token is predicted to be relatively unlikely compared to other tokens.

This bias towards shorter sequences can be problematic when scoring indel variants with likelihoods, as indel variants can differ in length from WT and each other. To compensate for this, when scoring indel variants, we apply an adjustment to the log likelihood that favors longer sequences over shorter sequences. We find that on a sample of random protein families from UniRef50, the log likelihood decreases roughly linearly with sequence length, with a slope of  $-1.96$  (Figure 5).

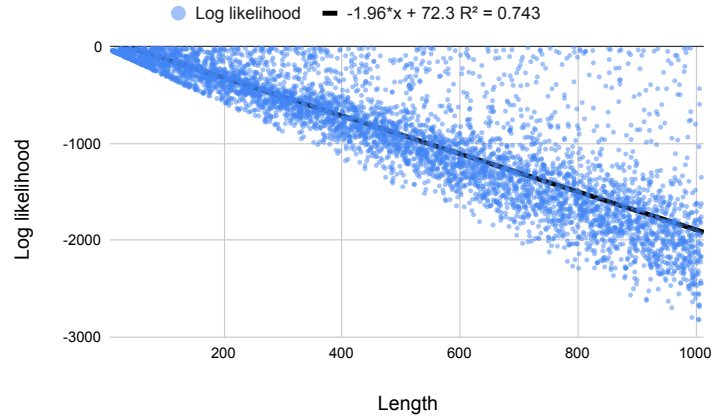


Figure 5: Plot of log likelihood vs length for random UniRef50 protein families.

Therefore, we compute the length adjusted log likelihood as follows:

$$\text{adjusted log likelihood} = \text{log likelihood} + \alpha \times \text{sequence length} \quad (5)$$

where  $\alpha = 1.96$  is the length adjustment factor. To compute the adjusted log likelihood ratio for scoring variants, we simply use the adjusted log likelihood instead of the regular log likelihood.

On the DMS indels benchmark, we find that using the length adjusted likelihood ratio with  $\alpha = 1.96$  improves performance (Figure 6). The length adjustment factor  $\alpha = 1.96$  is near optimal, with the empirical best adjustment factor being 2.1.

On the other hand, on the clinical indels benchmark, we find that using the length adjusted likelihood ratio with  $\alpha = 1.96$  slightly harms performance (Figure 7). However, the optimal adjustment factor is non-zero (between 0.90 and 1.20 depending on the benchmark version), indicating that some length adjustment is generally ideal for zero-shot fitness prediction, regardless of the specific task. Given that the relation between log likelihood and length is not completely linear, and that the length only explains  $\sim 75\%$  of the variance in the log likelihood (Figure 5), there may be better ways to adjust the log likelihood using factors other than the length e.g. factors that may be dependent on the specific protein family of interest. We leave exploration of this to future work.

As our experiments show that adjusting log likelihoods for length is generally useful, even if  $\alpha = 1.96$  is not necessarily optimal, we always use the length adjusted log likelihood when comparing the performance of PoET-2 to other models.

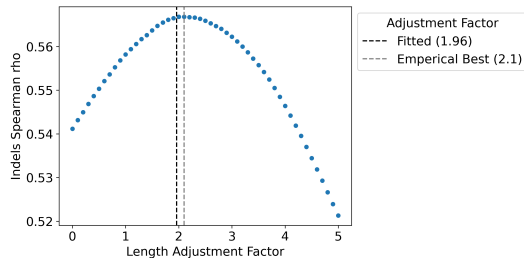


Figure 6: Plot of DMS indels performance ( $\rho$ ) vs length adjustment factor.

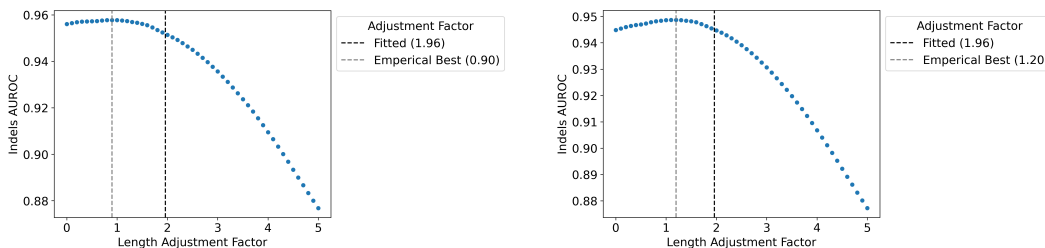


Figure 7: Plot of clinical indels performance (AUROC) versus length adjustment factor; results for ProteinGymV1 (left) and ProteinGymV1.3 (right).

### D.3 Model ensembles

We compute the zero-shot score for the ensemble model combining PoET-2 and VenusREM by computing a weighted average of the score from PoET-2 and the score from VenusREM:

$$\text{Ensemble Score} = w \times (\text{PoET-2 score}) + (1 - w) \times (\text{VenusREM score}) \quad (6)$$

where  $w \in [0, 1]$ . To select the weight  $w$ , we evaluate the performance of 21 values of  $w$  regularly spaced in the interval  $[0, 1]$  (inclusive; increments of 0.05) on ProteinNPT’s validation of set 8 datasets:

BLAT\_ECOLX\_Jacquier\_2013, CALM1\_HUMAN\_Weile\_2017, DYR\_ECOLI\_Thompson\_2019, DLG4\_RAT\_McLaughlin\_2012, P53\_HUMAN\_Giacomelli\_2018\_WT\_Nutlin, REV\_HV1H2\_Fernandes\_2016, RL40A\_YEAST\_Roscoe\_2013, TAT\_HV1BR\_Fernandes\_2016

On this validation set, we find that the optimal value of  $w$  is simply 0.5, corresponding to a simple average.

### D.4 Detailed results

The following tables detail results on the performance and standard error of models on the DMS zero-shot substitutions and indels benchmarks, broken by different metrics, and assay and protein subgroups.

- **Table 7** Overall performance (Spearman, AUC, MCC, NDCG, Recall) on substitutions benchmark, with standard error of difference to PoET-2.
- **Table 8** Overall performance (Spearman, AUC, MCC, NDCG, Recall) on substitutions benchmark, with standard error of difference to PoET-2 + VenusREM.
- **Table 9** Overall performance (Spearman, AUC, MCC, NDCG, Recall) on indels benchmark, with standard error of difference to PoET-2.
- **Table 10** Performance (Spearman) on substitutions benchmark broken down by assay type, with standard error of difference to PoET-2.
- **Table 11** Performance (Spearman) on substitutions benchmark broken down by MSA depth, with standard error of difference to PoET-2.
- **Table 12** Performance (Spearman) on substitutions benchmark broken down by taxonomy, with standard error of difference to PoET-2.
- **Table 13** Performance (Spearman) on substitutions benchmark broken down by mutation depth, with standard error of difference to PoET-2.

Table 7: Performance on zero-shot DMS substitutions benchmark. Standard error of difference to PoET-2 in parentheses.

Model	Metric				
	Spearman	AUC	MCC	NDCG	Recall
ESM-2	0.415 (0.013)	0.729 (0.007)	0.328 (0.010)	0.742 (0.007)	0.217 (0.007)
ESM C	0.407 (0.014)	0.727 (0.007)	0.323 (0.011)	0.742 (0.007)	0.213 (0.006)
ProGen2 M	0.379 (0.009)	0.711 (0.005)	0.299 (0.007)	0.747 (0.006)	0.203 (0.006)
ProGen2 XL	0.390 (0.009)	0.717 (0.005)	0.306 (0.008)	0.764 (0.004)	0.198 (0.005)
SaProt	0.457 (0.007)	0.751 (0.004)	0.358 (0.007)	0.764 (0.005)	0.232 (0.006)
ESM-3 Open	0.467 (0.005)	0.756 (0.003)	0.368 (0.005)	0.773 (0.004)	0.241 (0.006)
ProSST	0.508 (0.008)	0.777 (0.004)	0.399 (0.007)	0.752 (0.007)	0.235 (0.009)
MSA Transformer	0.431 (0.009)	0.736 (0.005)	0.338 (0.007)	0.774 (0.004)	0.225 (0.005)
TranceptEVE L	0.456 (0.006)	0.751 (0.003)	0.356 (0.005)	0.783 (0.003)	0.231 (0.005)
GEMME	0.455 (0.009)	0.749 (0.005)	0.352 (0.007)	0.773 (0.003)	0.212 (0.004)
PoET-1	0.470 (0.004)	0.759 (0.002)	0.367 (0.004)	0.779 (0.002)	0.226 (0.003)
S3F-MSA	0.496 (0.006)	0.771 (0.003)	0.388 (0.005)	0.788 (0.002)	0.244 (0.004)
VenusREM	0.519 (0.007)	0.783 (0.003)	0.405 (0.006)	0.766 (0.006)	0.243 (0.009)
PoET-2	0.500 (0.000)	0.773 (0.000)	0.391 (0.000)	0.786 (0.000)	0.238 (0.000)
PoET-2 + VenusREM	<b>0.543 (0.005)</b>	<b>0.796 (0.002)</b>	<b>0.423 (0.004)</b>	<b>0.791 (0.004)</b>	<b>0.254 (0.006)</b>

## D.5 Compute requirements

Inference with PoET-2 is performed on g5.xlarge instances from Amazon Web Services. The instances are equipped with A10G Nvidia GPUs with 24GB VRAM. For scoring sequences of average length (~350 amino acids), the inference throughput per prompt is approximately 125 sequences per second.

Table 8: Performance on zero-shot DMS substitutions benchmark. Standard error of difference to PoET-2 + VenusREM in parentheses.

Model	Metric				
	Spearman	AUC	MCC	NDCG	Recall
ESM-2	0.415 (0.014)	0.729 (0.007)	0.328 (0.011)	0.742 (0.007)	0.217 (0.007)
ESM C	0.407 (0.014)	0.727 (0.008)	0.323 (0.012)	0.742 (0.007)	0.213 (0.007)
ProGen2 M	0.379 (0.010)	0.711 (0.005)	0.299 (0.008)	0.747 (0.007)	0.203 (0.008)
ProGen2 XL	0.390 (0.010)	0.717 (0.005)	0.306 (0.009)	0.764 (0.006)	0.198 (0.007)
SaProt	0.457 (0.008)	0.751 (0.005)	0.358 (0.007)	0.764 (0.005)	0.232 (0.006)
ESM-3 Open	0.467 (0.007)	0.756 (0.004)	0.368 (0.006)	0.773 (0.004)	0.241 (0.006)
ProSST	0.508 (0.005)	0.777 (0.003)	0.399 (0.004)	0.752 (0.006)	0.235 (0.006)
MSA Transformer	0.431 (0.010)	0.736 (0.005)	0.338 (0.008)	0.774 (0.006)	0.225 (0.007)
TranceptEVE L	0.456 (0.007)	0.751 (0.004)	0.356 (0.006)	0.783 (0.004)	0.231 (0.006)
GEMME	0.455 (0.009)	0.749 (0.005)	0.352 (0.010)	0.773 (0.005)	0.212 (0.007)
PoET-1	0.470 (0.006)	0.759 (0.003)	0.367 (0.006)	0.779 (0.004)	0.226 (0.005)
S3F-MSA	0.496 (0.006)	0.771 (0.003)	0.388 (0.007)	0.788 (0.003)	0.244 (0.006)
VenusREM	0.519 (0.003)	0.783 (0.002)	0.405 (0.003)	0.766 (0.004)	0.243 (0.004)
PoET-2	0.500 (0.004)	0.773 (0.002)	0.391 (0.004)	0.786 (0.004)	0.238 (0.006)
PoET-2 + VenusREM	<b>0.543 (0.000)</b>	<b>0.796 (0.000)</b>	<b>0.423 (0.000)</b>	<b>0.791 (0.000)</b>	<b>0.254 (0.000)</b>

Table 9: Performance on zero-shot DMS indels benchmark. Standard error of difference to PoET-2 in parentheses.

Model	Metric				
	Spearman	AUC	MCC	NDCG	Recall
ProGen2 M	0.463 (0.037)	0.770 (0.021)	0.370 (0.031)	0.757 (0.018)	0.305 (0.017)
ProGen2 XL	0.427 (0.022)	0.747 (0.010)	0.323 (0.019)	0.749 (0.015)	0.297 (0.012)
TranceptEVE L	0.410 (0.020)	0.749 (0.011)	0.348 (0.020)	0.725 (0.013)	0.258 (0.014)
PoET-1	0.515 (0.006)	0.803 (0.005)	0.434 (0.011)	0.763 (0.006)	0.310 (0.010)
PoET-2	<b>0.567 (0.000)</b>	<b>0.831 (0.000)</b>	<b>0.478 (0.000)</b>	<b>0.795 (0.000)</b>	<b>0.340 (0.000)</b>

Table 10: Performance (Spearman  $\rho$ ) on zero-shot DMS substitutions benchmark broken down by assay type. Standard error of difference to PoET-2 in parentheses.

Model	Substitutions By Assay Type				
	Activity	Binding	Expression	Organismal Fitness	Stability
ESM-2	0.429 (0.028)	0.336 (0.043)	0.417 (0.030)	0.368 (0.024)	0.523 (0.016)
ESM C	0.426 (0.028)	0.313 (0.044)	0.408 (0.032)	0.360 (0.027)	0.526 (0.017)
ProGen2 M	0.396 (0.028)	0.291 (0.018)	0.434 (0.015)	0.379 (0.015)	0.396 (0.020)
ProGen2 XL	0.406 (0.021)	0.300 (0.028)	0.415 (0.023)	0.384 (0.014)	0.445 (0.012)
SaProt	0.461 (0.017)	0.380 (0.016)	0.488 (0.009)	0.366 (0.020)	0.592 (0.011)
ESM-3 Open	0.432 (0.013)	0.403 (0.013)	0.470 (0.008)	0.388 (0.012)	0.641 (0.011)
ProSST	0.480 (0.019)	0.444 (0.021)	0.532 (0.016)	0.430 (0.018)	<b>0.653 (0.011)</b>
MSA Transformer	0.477 (0.009)	0.324 (0.036)	0.447 (0.013)	0.416 (0.017)	0.492 (0.011)
TranceptEVE L	0.490 (0.014)	0.371 (0.018)	0.459 (0.012)	0.458 (0.007)	0.501 (0.011)
GEMME	0.485 (0.008)	0.380 (0.037)	0.440 (0.015)	0.450 (0.008)	0.519 (0.012)
PoET-1	0.498 (0.006)	0.391 (0.019)	0.466 (0.006)	0.474 (0.006)	0.519 (0.005)
S3F-MSA	0.506 (0.008)	0.437 (0.025)	0.480 (0.013)	0.476 (0.007)	0.582 (0.008)
VenusREM	0.499 (0.016)	0.452 (0.018)	0.535 (0.014)	0.459 (0.016)	0.651 (0.011)
PoET-2	0.508 (0.000)	0.423 (0.000)	0.503 (0.000)	0.482 (0.000)	0.582 (0.000)
PoET-2 + VenusREM	<b>0.538 (0.010)</b>	<b>0.475 (0.013)</b>	<b>0.552 (0.010)</b>	<b>0.505 (0.010)</b>	0.644 (0.007)

Table 11: Performance (Spearman  $\rho$ ) on zero-shot DMS substitutions benchmark broken down by MSA depth. Standard error of difference to PoET-2 in parentheses.

Model	Substitutions By MSA Depth		
	Low	Medium	High
ESM-2	0.340 (0.038)	0.410 (0.018)	0.513 (0.013)
ESM C	0.338 (0.040)	0.401 (0.020)	0.519 (0.011)
ProGen2 M	0.305 (0.031)	0.390 (0.016)	0.422 (0.016)
ProGen2 XL	0.322 (0.024)	0.411 (0.011)	0.442 (0.013)
SaProt	0.397 (0.027)	0.446 (0.014)	0.546 (0.011)
ESM-3 Open	0.402 (0.017)	0.465 (0.011)	0.575 (0.011)
ProSST	0.468 (0.029)	0.506 (0.013)	0.581 (0.013)
MSA Transformer	0.375 (0.024)	0.456 (0.011)	0.480 (0.012)
TranceptEVE L	0.434 (0.015)	0.473 (0.008)	0.491 (0.009)
GEMME	0.445 (0.017)	0.474 (0.008)	0.494 (0.009)
PoET-1	0.479 (0.008)	0.477 (0.006)	0.511 (0.005)
S3F-MSA	0.470 (0.017)	0.509 (0.005)	0.547 (0.007)
VenusREM	0.498 (0.023)	0.524 (0.011)	0.578 (0.013)
PoET-2	0.488 (0.000)	0.507 (0.000)	0.555 (0.000)
PoET-2 + VenusREM	<b>0.528 (0.016)</b>	<b>0.550 (0.007)</b>	<b>0.593 (0.008)</b>

Table 12: Performance (Spearman  $\rho$ ) on zero-shot DMS substitutions benchmark broken down by taxonomy. Standard error of difference to PoET-2 in parentheses.

Model	Substitutions By Taxonomy			
	Human	Other Eukaryote	Prokaryote	Virus
ESM-2	0.457 (0.011)	0.488 (0.031)	0.459 (0.019)	0.262 (0.043)
ESM C	0.467 (0.010)	0.482 (0.030)	0.442 (0.022)	0.245 (0.051)
ProGen2 M	0.412 (0.011)	0.418 (0.027)	0.355 (0.027)	0.334 (0.035)
ProGen2 XL	0.385 (0.012)	0.459 (0.017)	0.417 (0.017)	0.401 (0.024)
SaProt	0.478 (0.010)	0.530 (0.018)	0.515 (0.013)	0.323 (0.037)
ESM-3 Open	0.480 (0.008)	0.549 (0.015)	0.530 (0.016)	0.407 (0.028)
ProSST	0.518 (0.013)	0.577 (0.019)	0.550 (0.018)	0.449 (0.028)
MSA Transformer	0.439 (0.012)	0.517 (0.012)	0.445 (0.014)	0.419 (0.028)
TranceptEVE L	0.472 (0.007)	0.515 (0.014)	0.455 (0.013)	0.460 (0.020)
GEMME	0.468 (0.009)	0.519 (0.013)	0.467 (0.011)	0.471 (0.019)
PoET-1	0.481 (0.005)	0.543 (0.009)	0.464 (0.008)	0.491 (0.011)
S3F-MSA	0.501 (0.008)	0.561 (0.010)	0.521 (0.008)	0.502 (0.012)
VenusREM	0.530 (0.011)	0.586 (0.017)	0.550 (0.016)	0.489 (0.023)
PoET-2	0.506 (0.000)	0.569 (0.000)	0.507 (0.000)	0.528 (0.000)
PoET-2 + VenusREM	<b>0.548 (0.008)</b>	<b>0.604 (0.011)</b>	<b>0.562 (0.009)</b>	<b>0.551 (0.013)</b>

Table 13: Performance (Spearman  $\rho$ ) on zero-shot DMS substitutions benchmark broken down by mutation depth. Standard error of difference to PoET-2 in parentheses.

Model	Substitutions By Mutation Depth				
	1	2	3	4	5+
ESM-2	0.423 (0.011)	0.248 (0.021)	0.203 (0.077)	0.160 (0.077)	0.220 (0.073)
ESM C	0.416 (0.012)	0.257 (0.022)	0.189 (0.073)	0.150 (0.073)	0.217 (0.074)
ProGen2 M	0.372 (0.010)	0.131 (0.025)	0.149 (0.059)	0.131 (0.066)	0.178 (0.062)
ProGen2 XL	0.384 (0.008)	0.181 (0.023)	0.267 (0.046)	0.229 (0.046)	0.283 (0.047)
SaProt	0.459 (0.010)	0.312 (0.018)	0.271 (0.048)	0.268 (0.051)	0.337 (0.056)
ESM-3 Open	0.487 (0.009)	0.336 (0.019)	0.303 (0.047)	0.284 (0.046)	0.365 (0.050)
ProSST	0.520 (0.009)	0.393 (0.025)	0.316 (0.046)	0.274 (0.049)	0.334 (0.060)
MSA Transformer	0.427 (0.008)	0.220 (0.019)	0.358 (0.026)	0.365 (0.017)	0.401 (0.022)
TranceptEVE L	0.446 (0.006)	0.277 (0.014)	0.349 (0.041)	0.327 (0.039)	0.385 (0.046)
GEMME	0.447 (0.006)	0.275 (0.017)	0.329 (0.044)	0.338 (0.028)	0.419 (0.022)
PoET-1	0.466 (0.004)	0.298 (0.010)	0.412 (0.019)	0.393 (0.014)	0.421 (0.011)
S3F-MSA	0.499 (0.005)	0.332 (0.011)	0.377 (0.017)	0.343 (0.017)	0.387 (0.033)
VenusREM	0.534 (0.008)	0.395 (0.023)	0.352 (0.046)	0.320 (0.045)	0.372 (0.050)
PoET-2	0.506 (0.000)	0.357 (0.000)	<b>0.444 (0.000)</b>	<b>0.419 (0.000)</b>	<b>0.447 (0.000)</b>
PoET-2 + VenusREM	<b>0.556 (0.005)</b>	<b>0.402 (0.014)</b>	0.442 (0.023)	0.411 (0.020)	0.441 (0.028)

## E Supervised variant effect prediction

**Overview** As described in the main text, our supervised variant effect prediction methodology employs a Gaussian Process (GP) regression model to predict fitness scores. The GP is configured with a constant mean function and a product kernel. This kernel integrates information from two Matérn 5/2 sub-kernels, each operating on distinct features derived from PoET-2.

One sub-kernel operates on protein embeddings derived from the last layer of PoET-2’s MLM decoder. Given the high dimensionality of the full per-residue embeddings produced by PoET-2, a dimensionality reduction step is applied prior to their use in the GP. Specifically, we utilize Singular Value Decomposition (SVD) to project the full embeddings into a 1024-dimensional space. This dimensionality reduction strategy is conceptually similar to the PCA-based approach used by Bepler et al. [6] to improve the runtime of their learning algorithm, which is also a GP for protein fitness prediction. For each wild-type (WT) protein in an assay, the SVD transformation is fitted on a set of 1536 variants: this set comprises the WT sequence itself and a random sample of 1535 single and double substitution mutants of that WT. The number of variants used for fitting SVDs (1536) was chosen to be approximately 50% larger than the number of SVD components (1024). This was deemed a practical trade-off to provide a reasonable basis for the decomposition while managing the computational requirements of fitting SVD transformations for each of the numerous proteins in the ProteinGym benchmark; fitting the SVD on a larger or more diverse set of variants may improve performance.

The second Matérn 5/2 sub-kernel in the product utilizes the log likelihood ratios (LLRs) obtained from PoET-2’s CLM decoder, as detailed in the zero-shot methods section of the main text (§4.1.1). The final GP model thus learns from both the reduced-dimensionality MLM embeddings and the CLM-derived LLRs.

**Gaussian Process Hyperparameter Priors** To improve the stability and performance of GP training, particularly with small training datasets, we incorporate empirical priors on the GP hyperparameters. This process involves three steps: (1) We first fit individual GP models (with the architecture described above) to each of the 8 validation datasets specified by ProteinNPT [7] (listed in Appendix D.3). (2) From these 8 trained GPs, we extract the optimized hyperparameters and fit empirical distributions to them. Specifically, a Normal distribution is fitted to the learned constant mean values, while Gamma distributions are fitted to the other hyperparameters (i.e. the lengthscales of both Matérn kernels, the outputscale of the product kernel, and the likelihood noise term). (3) These fitted Normal and Gamma distributions then serve as priors for the respective hyperparameters when training GPs on the ProteinGym benchmark assays.

This prior-informed approach is particularly beneficial for assays with limited training data (e.g. fewer than ~50 data points), where the prior helps guide the optimization process. For larger training set sizes, the influence of the prior diminishes. When conducting ablation studies involving different foundation models to generate the input embeddings and LLRs, the procedure for deriving these priors (steps 1 and 2) is repeated to learn a separate, appropriate set of hyperparameter priors for each distinct foundation model.

### E.1 Prompt engineering

Similar to our approach to prompt engineering for zero-shot variant effect prediction (Appendix D.1), for supervised prediction, we also explore two prompt engineering methods.

**Ensembling over context length and maximum similarity** We ensemble over different values of context length  $\in \{6144, 12288, 24576, 49152, 98304\}$  and always use a maximum similarity value of 0.95, resulting in 5 combinations in total. We use a wide range of context lengths compared to zero-shot prediction because we observed in early experiments on ProteinNPT’s validation set of 8 datasets (Appendix D.3) that longer contexts lengths generally had a small positive impact on supervised prediction performance (in contrast, for PoET-1, it was observed that long context lengths could have a negative effect on zero-shot prediction [1]). Figure 8 shows the performance of the GP model with different values for the context length, and the performance of the ensemble model. We use a fixed value of 0.95 for maximum similarity because 0.95 is typically the best value for



zero-shot prediction, and did not explore other values in order to conserve compute. Therefore, it is most likely the case that there exists more optimal parameters for ensembling.

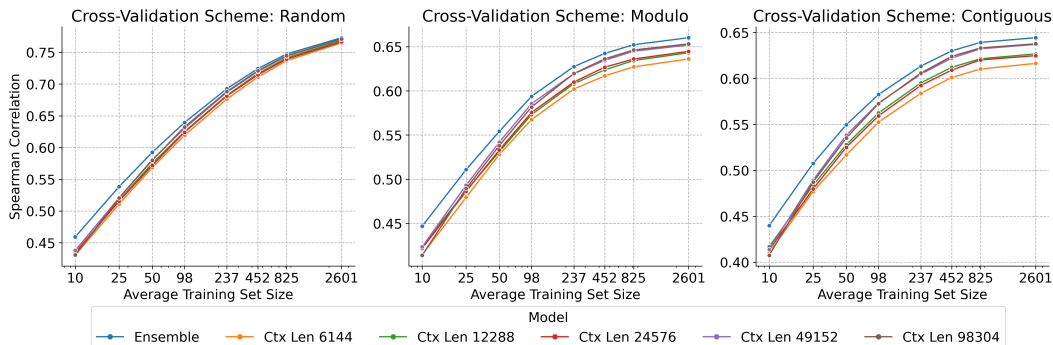


Figure 8: Performance on the supervised DMS substitutions benchmark as a function of training set size of (1) PoET-2 GP models using prompts with 5 different context lengths and (2) the ensemble GP model that ensembles the five models from (1).

**Incorporating structure in the prompt** Table 14 shows the performance of various strategies for incorporating structure in the prompt on an extended validation set of 30 assays from ProteinGym’s DMS supervised substitutions benchmark. We find that all strategies, both those that do and don’t incorporate structure in the prompt, perform about the same for supervised variant effect prediction, despite significant improvements for zero-shot variant effect prediction when using strategies that incorporate structure in the prompt. Therefore, for supervised variant effect prediction, we do not include structure in the prompt.

Table 14: Performance (Spearman’s  $\rho$ ) of different strategies for including structure in the prompt on a validation set of 30 datasets from the supervised DMS substitutions benchmark.

Prompt		Supervised Cross-Validation Scheme				Zero-shot
Context Modalities	Query	Rand.	Mod.	Contig.	Avg.	
Sequence	None	<b>0.70963</b>	<b>0.60398</b>	<b>0.49490</b>	<b>0.60284</b>	0.40711
Sequence and Structure	None	0.70689	0.59785	0.49312	0.59929	0.42554
Sequence	Structure of WT	0.70436	0.60179	0.49142	0.59919	0.42277
Sequence and Structure	Structure of WT	0.70103	0.58561	0.47449	0.58704	<b>0.43748</b>

The extended validation set consists of ProteinNPT’s 8 validation datasets (Appendix D.3), plus 22 randomly selected datasets. The full list of extended validation datasets is as follows:

BLAT\_ECOLX\_Jacquier\_2013, CALM1\_HUMAN\_Weile\_2017, DYR\_ECOLI\_Thompson\_2019, DLG4\_RAT\_McLaughlin\_2012, P53\_HUMAN\_Giacomelli\_2018\_WT\_Nutlin, REV\_HV1H2\_Fernandes\_2016, RL40A\_YEAST\_Roscoe\_2013, TAT\_HV1BR\_Fernandes\_2016, ACE2\_HUMAN\_Chan\_2020, BCHB\_CHLTE\_Tsuboyama\_2023\_2KRU, CAR11\_HUMAN\_Meitlis\_2020\_gof, CP2C9\_HUMAN\_Amorosi\_2021\_abundance, ENVZ\_ECOLI\_Ghose\_2023, F7YBW7\_MESOW\_Ding\_2023, GCN4\_YEAST\_Staller\_2018, GLPA\_HUMAN\_Elazar\_2016, HCP\_LAMBD\_Tsuboyama\_2023\_2L6Q, KCNH2\_HUMAN\_Kozek\_2020, LYAM1\_HUMAN\_Elazar\_2016, MBD11\_ARATH\_Tsuboyama\_2023\_6ACV, MTHR\_HUMAN\_Weile\_2021, OBSCN\_HUMAN\_Tsuboyama\_2023\_1V1C, OPSD\_HUMAN\_Wan\_2019, PA\_I34A1\_Wu\_2015, PSAE\_SYNP2\_Tsuboyama\_2023\_1PSE, PTEN\_HUMAN\_Matreyek\_2021, Q53Z42\_HUMAN\_McShan\_2019\_binding-TAPBPR, RNC\_ECOLI\_Weeks\_2023, SPG1\_STRSG\_Olson\_2014, TADBP\_HUMAN\_Bolognesi\_2019

## E.2 Gaussian Process kernels

Our primary Gaussian Process (GP) model for supervised variant effect prediction utilizes a product kernel combining two Matérn 5/2 sub-kernels: one operating on PoET-2 MLM embeddings and the

other on PoET-2 CLM log likelihood ratios (LLRs). While this product kernel is effective for single-site substitutions, its performance can be suboptimal when predicting the effects of multi-mutation variants due to the behavior of the LLR-based sub-kernel under distributional shift.

Specifically, the distribution of LLRs for multi-mutation variants often differs significantly from that of single-site mutations, with multi-mutation LLRs tending towards much lower or higher values. Consequently, if a GP is trained predominantly on single-site variants and then applied to predict multi-mutation variants, the LLRs of the test set can be markedly out-of-distribution (OOD) relative to the training data.

This OOD characteristic poses a challenge for the Matérn 5/2 kernel operating on LLRs. Stationary kernels like the Matérn assume that covariance is a function of the distance between inputs; when test LLRs are far outside the training distribution’s range, their covariance with the training data (as modeled by this kernel) diminishes significantly. Since our model employs a product kernel, if the LLR sub-kernel assigns low covariance to a test point, the overall covariance for that point will also be low, irrespective of the embedding-based sub-kernel. In such cases, the GP prediction tends to revert towards the prior mean, offering limited predictive power for these OOD multi-mutation variants.

We leave the exploration of more sophisticated methods for incorporating LLRs into the GP for multi-mutation contexts (e.g. using LLR transformations or non-stationary kernels) to future work. For the current work, a pragmatic approach to mitigate this issue when predicting the effects of multi-mutation variants is to utilize a GP model that relies solely on the embedding-based Matérn kernel, thereby omitting the LLR-based sub-kernel for these specific predictions. Even with this simplification, a PoET-2 based GP using only embeddings can achieve strong performance in predicting the effects of multi-mutation variants when trained on data from single-site or lower-order mutants, outperforming other state-of-the-art methods as discussed in Appendix E.5

### E.3 Detailed results

The following tables detail results on the performance and standard error of models on the DMS supervised benchmark, broken by different metrics, cross-validation schemes, and assay and protein subgroups.

- **Table 15** Performance (Spearman) broken down by cross-validation scheme, with standard error of difference to PoET-2.
- **Table 16** Performance (MSE) broken down by cross-validation scheme, with standard error of difference to PoET-2.
- **Table 17** Performance (average Spearman across cross-validation schemes) on substitutions benchmark broken down by assay type, with standard error of difference to PoET-2.
- **Table 18** Performance (average Spearman across cross-validation schemes) on substitutions benchmark broken down by MSA depth, with standard error of difference to PoET-2.
- **Table 19** Performance (average Spearman across cross-validation schemes) on substitutions benchmark broken down by taxonomy, with standard error of difference to PoET-2.

Table 15: Performance (Spearman  $\rho$ ) on supervised DMS substitutions benchmark. Standard error of difference to PoET-2 GP in parentheses.

Model	Spearman $\rho$ ( $\uparrow$ )			
	Random	Modulo	Contiguous	Average
ProteinNPT	0.741 (0.003)	0.588 (0.012)	0.529 (0.018)	0.619 (0.010)
Kermut	0.746 (0.004)	0.635 (0.008)	0.613 (0.009)	0.664 (0.006)
ESM-2 (650 M) GP	0.749 (0.002)	0.573 (0.009)	0.549 (0.010)	0.624 (0.007)
ESM C GP	0.747 (0.004)	0.605 (0.007)	0.573 (0.010)	0.642 (0.006)
PoET-2 GP	<b>0.773 (0.000)</b>	<b>0.661 (0.000)</b>	<b>0.645 (0.000)</b>	<b>0.693 (0.000)</b>

Table 16: Performance (MSE) on supervised DMS substitutions benchmark. Standard error of difference to PoET-2 GP in parentheses.

Model	MSE ( $\downarrow$ )			
	Random	Modulo	Contiguous	Average
ProteinNPT	0.441 (0.012)	0.765 (0.023)	0.856 (0.025)	0.687 (0.015)
Kermut	0.413 (0.004)	0.649 (0.010)	0.697 (0.010)	0.586 (0.007)
ESM-2 (650 M) GP	0.404 (0.004)	0.720 (0.011)	0.768 (0.011)	0.630 (0.008)
ESM C GP	0.398 (0.004)	0.660 (0.008)	0.716 (0.009)	0.592 (0.007)
PoET-2 GP	<b>0.370 (0.000)</b>	<b>0.602 (0.000)</b>	<b>0.647 (0.000)</b>	<b>0.540 (0.000)</b>

Table 17: Performance (Spearman  $\rho$ ) on supervised DMS substitutions benchmark broken down by assay type. Standard error of difference to PoET-2 GP in parentheses.

Model	Substitutions By Assay Type				
	Activity	Binding	Expression	Organismal Fitness	Stability
ProteinNPT	0.590 (0.007)	0.541 (0.045)	0.631 (0.011)	0.558 (0.010)	0.776 (0.005)
Kermut	0.606 (0.007)	0.627 (0.027)	0.680 (0.010)	0.584 (0.007)	0.825 (0.004)
ESM-2 (650 M) GP	0.569 (0.014)	0.577 (0.026)	0.633 (0.012)	0.545 (0.010)	0.795 (0.006)
ESM C GP	0.575 (0.015)	0.601 (0.021)	0.656 (0.013)	0.550 (0.013)	0.828 (0.006)
PoET-2 GP	<b>0.630 (0.000)</b>	<b>0.667 (0.000)</b>	<b>0.691 (0.000)</b>	<b>0.622 (0.000)</b>	<b>0.854 (0.000)</b>

Table 18: Performance (Spearman  $\rho$ ) on supervised DMS substitutions benchmark broken down by MSA depth. Standard error of difference to PoET-2 GP in parentheses.

Model	Substitutions By MSA Depth		
	Low	Medium	High
ProteinNPT	0.576 (0.016)	0.621 (0.010)	0.705 (0.006)
Kermut	0.619 (0.012)	0.658 (0.005)	0.743 (0.005)
ESM-2 (650 M) GP	0.561 (0.019)	0.618 (0.007)	0.721 (0.006)
ESM C GP	0.581 (0.019)	0.627 (0.010)	0.749 (0.005)
PoET-2 GP	<b>0.667 (0.000)</b>	<b>0.689 (0.000)</b>	<b>0.769 (0.000)</b>

Table 19: Performance (Spearman  $\rho$ ) on supervised DMS substitutions benchmark broken down by taxonomy. Standard error of difference to PoET-2 GP in parentheses.

Model	Substitutions By Taxonomy			
	Human	Other Eukaryote	Prokaryote	Virus
ProteinNPT	0.633 (0.007)	0.673 (0.006)	0.666 (0.012)	0.602 (0.019)
Kermut	0.671 (0.006)	0.712 (0.006)	0.707 (0.004)	0.628 (0.012)
ESM-2 (650 M) GP	0.649 (0.006)	0.668 (0.016)	0.673 (0.009)	0.552 (0.016)
ESM C GP	0.674 (0.005)	0.682 (0.017)	0.696 (0.007)	0.542 (0.023)
PoET-2 GP	<b>0.696 (0.000)</b>	<b>0.738 (0.000)</b>	<b>0.736 (0.000)</b>	<b>0.690 (0.000)</b>

## E.4 Compute requirements

- The computation of the SVD of embeddings from protein foundation models is performed on r6a.4xlarge instances from Amazon Web Services. These instances are equipped with 16 vCPUs and 128GB of RAM. The amount of RAM required to fit the SVD depends on the number of training samples, the length of the WT sequence, and the embedding dimension of the foundation model used. Generally we use 1536 training samples, as described at the start of this section. For some long sequence and models with very large embedding dimension, the number of samples may need to be decreased to fit within the available RAM.
- Embeddings and log likelihood ratios for supervised variant effect prediction are computed using the same computational resources as log likelihood ratios for zero-shot variant effect prediction (Appendix D.5)
- Gaussian process models are trained on g5.xlarge instances from Amazon Web Services. These instances are equipped with A10G Nvidia GPUs that have 24GB of VRAM.

## E.5 Prediction of mutational effects of multi-mutation variants

As discussed in Appendix E.2 when predicting multi-mutation variant effects from data on single-site or lower-order mutants, we utilize a Gaussian Process (GP) model with a kernel based solely on PoET-2 embeddings, omitting log-likelihood ratios. To evaluate this embedding-only PoET-2 GP in such challenging generalization scenarios, we benchmark it on the multi-mutation dataset introduced in the ProGen3 publication [44]. The ProGen3 authors report that their model outperforms Kermut [15] and matches ConFit [45] on this specific benchmark. Our PoET-2 based GP, however, surpasses all three aforementioned models (Table 20).

The ProGen3 multi-mutation benchmark comprises 8 datasets selected from ProteinGym. The selection criteria, as stated by the ProGen3 authors, is as follows: "We identify all assays in ProteinGym with at least 3 mutations, and we train on all variants at most k mutations from the wild type, where k is the smallest number required for the train split to exceed 500 sequences. To ensure that the train and test splits contain proteins of similar fitness, we require that the total variation distance between the train and test distributions of fitness scores be less than 1. These filters yield 8 assays that measure diverse functional attributes for a wide range of proteins."

Table 20 details the performance of PoET-2 GP alongside ProGen3, Kermut, and ConFit on this multi-mutation benchmark; PoET-2 GP outperforms all other models.

Table 20: Performance comparison on multi-mutation variant effect prediction benchmark.

Model	Spearman $\rho$ ( $\uparrow$ )
Kermut	0.628
ConFit	0.679
ProGen3	0.673
PoET-2 GP	<b>0.708</b>

## F Licenses

Existing assets used in this paper are licensed as follows:

- ProteinGym benchmark: MIT license
- UniRef protein database: CC BY 4.0 license
- AlphaFold database: CC BY 4.0 license
- ESM C [46] model: Cambrian Open License Agreement [47]

## G Additional details

### G.1 Statistical significance analysis

Statistical significance for all experiments is assessed by performing a non-parametric, two-sided bootstrap test with at least 10,000 samples. Bootstrap is performed using the same methodology as in ProteinGym [5].

## H Errata

The zero-shot variant effect prediction results section (§4.1.2) incorrectly states that the difference in performance between PoET-2 and PoET-1 on the DMS indels benchmark is not statistically significant. In fact, the difference is statistically significant with  $p < 1e - 5$ . This will be corrected in the final publication.

## References

- [1] 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](https://papers.nips.cc/paper_files/paper/2023/hash/f4366126eba252699b280e8f93c0ab2f-Abstract-Conference.html)
- [2] Joshua Meier, Roshan Rao, Robert Verkuil, Jason Liu, Tom Sercu, and Alex Rives. Language models enable zero-shot prediction of the effects of mutations on protein function. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 29287–29303. Curran Associates, Inc., 2021. URL <https://proceedings.neurips.cc/paper/2021/file/f51338d736f95dd42427296047067694-Paper.pdf>
- [3] Pascal Notin, Lood Van Niekerk, Aaron W Kollasch, Daniel Ritter, Yarin Gal, and Debora Susan Marks. TranceptEVE: Combining family-specific and family-agnostic models of protein sequences for improved fitness prediction. In *NeurIPS 2022 Workshop on Learning Meaningful Representations of Life*, 2022. URL <https://openreview.net/forum?id=170o9DcLmR1>
- [4] Pascal Notin, Mafalda Dias, Jonathan Frazer, Javier Marchena Hurtado, Aidan N Gomez, Debora Marks, and Yarin Gal. Tranception: Protein fitness prediction with autoregressive transformers and inference-time retrieval. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 16990–17017. PMLR, 17–23 Jul 2022. URL <https://proceedings.mlr.press/v162/notin22a.html>
- [5] 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. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 64331–64379. Curran Associates, Inc., 2023. URL [https://proceedings.neurips.cc/paper\\_files/paper/2023/file/cac723e5ff29f65e3fcbb0739ae91bee-Paper-Datasets\\_and\\_Benchmarks.pdf](https://proceedings.neurips.cc/paper_files/paper/2023/file/cac723e5ff29f65e3fcbb0739ae91bee-Paper-Datasets_and_Benchmarks.pdf)

- [6] Tristan Bepler and Bonnie Berger. Learning the protein language: Evolution, structure, and function. *Cell Systems*, 12(6):654–669.e3, 2021. ISSN 2405-4712. doi: <https://doi.org/10.1016/j.cels.2021.05.017>. URL <https://www.sciencedirect.com/science/article/pii/S2405471221002039>
- [7] Pascal Notin, Ruben Weitzman, Debora Marks, and Yarin Gal. ProteinNPT: Improving protein property prediction and design with non-parametric transformers. In A. Oh, T. Naudmann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 33529–33563. Curran Associates, Inc., 2023. URL [https://proceedings.neurips.cc/paper\\_files/paper/2023/file/6a4d5d85f7a52f062d23d98d544a5578-Paper-Conference.pdf](https://proceedings.neurips.cc/paper_files/paper/2023/file/6a4d5d85f7a52f062d23d98d544a5578-Paper-Conference.pdf)
- [8] Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, and Rob Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15):e2016239118, April 2021. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.2016239118. URL <https://pnas.org/doi/full/10.1073/pnas.2016239118>
- [9] Thomas Hayes, Roshan Rao, Halil Akin, Nicholas J. Sofroniew, Deniz Oktay, Zeming Lin, Robert Verkuil, Vincent Q. Tran, Jonathan Deaton, Marius Wiggert, Rohil Badkundri, Irhum Shafkat, Jun Gong, Alexander Derry, Raul S. Molina, Neil Thomas, Yousuf Khan, Chetan Mishra, Carolyn Kim, Liam J. Bartie, Matthew Nemeth, Patrick D. Hsu, Tom Sercu, Salvatore Candido, and Alexander Rives. Simulating 500 million years of evolution with a language model. July 2024. doi: 10.1101/2024.07.01.600583. URL <https://www.biorxiv.org/content/10.1101/2024.07.01.600583v1>
- [10] 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, 2023. doi: 10.1126/science.ade2574. URL <https://www.science.org/doi/abs/10.1126/science.ade2574>
- [11] Bo Chen, Xingyi Cheng, Yangli ao Geng, Shen Li, Xin Zeng, Boyan Wang, Jing Gong, Chiming Liu, Aohan Zeng, Yuxiao Dong, Jie Tang, and Le Song. xtrimopglm: Unified 100b-scale pre-trained transformer for deciphering the language of protein. *bioRxiv*, 2023. doi: 10.1101/2023.07.05.547496. URL <https://www.biorxiv.org/content/early/2023/07/14/2023.07.05.547496>
- [12] Eli N Weinstein, Alan Nawzad Amin, Jonathan Frazer, and Debora Susan Marks. Non-identifiability and the blessings of misspecification in models of molecular fitness. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL <https://openreview.net/forum?id=CwG-o0ind6t>
- [13] Mingchen Li, Yang Tan, Xinzhu Ma, Bozitao Zhong, Ziyi Zhou, Huiqun Yu, Wanli Ouyang, Liang Hong, Bingxin Zhou, and Pan Tan. Pross: Protein language modeling with quantized structure and disentangled attention. *bioRxiv*, 2024. doi: 10.1101/2024.04.15.589672. URL <https://www.biorxiv.org/content/early/2024/05/17/2024.04.15.589672.1>
- [14] Jin Su, Chenchen Han, Yuyang Zhou, Junjie Shan, Xibin Zhou, and Fajie Yuan. Saprot: Protein language modeling with structure-aware vocabulary. *bioRxiv*, 2023.
- [15] Peter Mørch Groth, Mads Herbert Kern, Lars Olsen, Jesper Salomon, and Wouter Boomsma. Kermut: Composite kernel regression for protein variant effects. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 29514–29565. Curran Associates, Inc., 2024. URL [https://proceedings.neurips.cc/paper\\_files/paper/2024/file/34547650b2ca69d91f3b3c3ae8b21962-Paper-Conference.pdf](https://proceedings.neurips.cc/paper_files/paper/2024/file/34547650b2ca69d91f3b3c3ae8b21962-Paper-Conference.pdf)



- [16] Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, Debsindhu Bhowmik, and Burkhard Rost. Prottrans: Toward understanding the language of life through self-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):7112–7127, October 2022. ISSN 1939-3539. doi: 10.1109/TPAMI.2021.3095381. URL <https://ieeexplore.ieee.org/document/9477085>.
- [17] Jonathan Frazer, Pascal Notin, Mafalda Dias, Aidan Gomez, Joseph K. Min, Kelly Brock, Yarin Gal, and Debora S. Marks. Disease variant prediction with deep generative models of evolutionary data. *Nature*, 599(7883):91–95, Nov 2021. ISSN 1476-4687. doi: 10.1038/s41586-021-04043-8. URL <https://doi.org/10.1038/s41586-021-04043-8>.
- [18] 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, Oct 2018. ISSN 1548-7105. doi: 10.1038/s41592-018-0138-4. URL <https://doi.org/10.1038/s41592-018-0138-4>.
- [19] Roshan M Rao, Jason Liu, Robert Verkuil, Joshua Meier, John Canny, Pieter Abbeel, Tom Sercu, and Alexander Rives. Msa transformer. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8844–8856. PMLR, 18–24 Jul 2021. URL <https://proceedings.mlr.press/v139/rao21a.html>.
- [20] Bowen Jing, Stephan Eismann, Patricia Suriana, Raphael J. L. Townshend, and Ron Dror. Learning from protein structure with geometric vector perceptrons, 2021. URL <https://arxiv.org/abs/2009.01411>.
- [21] J. Dauparas, I. Anishchenko, N. Bennett, H. Bai, R. J. Ragotte, L. F. Milles, B. I. M. Wicky, A. Courbet, R. J. de Haas, N. Bethel, P. J. Y. Leung, T. F. Huddy, S. Pellock, D. Fischer, F. Chan, B. Koepnick, H. Nguyen, A. Kang, B. Sankaran, A. K. Bera, N. P. King, and D. Baker. Robust deep learning-based protein sequence design using proteinmpnn. *Science*, 378(6615):49–56, 2022. doi: 10.1126/science.add2187. URL <https://www.science.org/doi/abs/10.1126/science.add2187>.
- [22] Chloe Hsu, Robert Verkuil, Jason Liu, Zeming Lin, Brian Hie, Tom Sercu, Adam Lerer, and Alexander Rives. Learning inverse folding from millions of predicted structures. *ICML*, 2022. doi: 10.1101/2022.04.10.487779. URL <https://www.biorxiv.org/content/early/2022/04/10/2022.04.10.487779>.
- [23] Zuobai Zhang, Pascal Notin, Yining Huang, Aurélie Lozano, Vijil Chenthamarakshan, Debora Marks, Payel Das, and Jian Tang. Multi-scale representation learning for protein fitness prediction, 2024. URL <https://arxiv.org/abs/2412.01108>.
- [24] Matsvei Tsishyn, Pauline Hermans, Fabrizio Pucci, and Marianne Rooman. Residue conservation and solvent accessibility are (almost) all you need for predicting mutational effects in proteins. *bioRxiv*, 2025. doi: 10.1101/2025.02.03.636212. URL <https://www.biorxiv.org/content/early/2025/02/04/2025.02.03.636212>.
- [25] Chloe Hsu, Hunter Nisonoff, Clara Fannjiang, and Jennifer Listgarten. Learning protein fitness models from evolutionary and assay-labeled data. *Nature Biotechnology*, 40(7):1114–1122, Jul 2022. ISSN 1546-1696. doi: 10.1038/s41587-021-01146-5. URL <https://doi.org/10.1038/s41587-021-01146-5>.
- [26] Kevin K. Yang, Zachary Wu, and Frances H. Arnold. Machine-learning-guided directed evolution for protein engineering. *Nature Methods*, 16(8):687–694, Aug 2019. ISSN 1548-7105. doi: 10.1038/s41592-019-0496-6. URL <https://doi.org/10.1038/s41592-019-0496-6>.
- [27] Tristan Bepler and Bonnie Berger. Learning protein sequence embeddings using information from structure. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=SygLehCqtm>.

- [28] 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*, 596(7873):583–589, Aug 2021. ISSN 1476-4687. doi: 10.1038/s41586-021-03819-2. URL <https://doi.org/10.1038/s41586-021-03819-2>
- [29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS’17*, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.
- [30] Mihaly Varadi, Stephen Anyango, Mandar Deshpande, Sreenath Nair, Cindy Natassia, Galabina Yordanova, David Yuan, Oana Stroe, Gemma Wood, Agata Laydon, Augustin Žídek, Tim Green, Kathryn Tunyasuvunakool, Stig Petersen, John Jumper, Ellen Clancy, Richard Green, Ankur Vora, Mira Lutfi, Michael Figurnov, Andrew Cowie, Nicole Hobbs, Pushmeet Kohli, Gerard Kleywegt, Ewan Birney, Demis Hassabis, and Sameer Velankar. Alphafold protein structure database: massively expanding the structural coverage of protein-sequence space with high-accuracy models. *Nucleic Acids Research*, 50(D1):D439–D444, 11 2021. ISSN 0305-1048. doi: 10.1093/nar/gkab1061. URL <https://doi.org/10.1093/nar/gkab1061>
- [31] Mihaly Varadi, Damian Bertoni, Paulyna Magana, Urmila Paramval, Ivanna Pidruchna, Malarvizhi Radhakrishnan, Maxim Tsenkov, Sreenath Nair, Milot Mirdita, Jingi Yeo, Oleg Kovalevskiy, Kathryn Tunyasuvunakool, Agata Laydon, Augustin Žídek, Hamish Tomlinson, Dhavanthi Hariharan, Josh Abrahamson, Tim Green, John Jumper, Ewan Birney, Martin Steinegger, Demis Hassabis, and Sameer Velankar. Alphafold protein structure database in 2024: providing structure coverage for over 214 million protein sequences. *Nucleic Acids Research*, 52(D1):D368–D375, 11 2023. ISSN 0305-1048. doi: 10.1093/nar/gkad1011. URL <https://doi.org/10.1093/nar/gkad1011>
- [32] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL <http://jmlr.org/papers/v21/20-074.html>
- [33] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2021. URL <https://arxiv.org/abs/2104.09864>
- [34] Noam Shazeer. Glu variants improve transformer, 2020. URL <https://arxiv.org/abs/2002.05202>
- [35] Biao Zhang and Rico Sennrich. Root mean square layer normalization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL [https://proceedings.neurips.cc/paper\\_files/paper/2019/file/1e8a19426224ca89e83cef47f1e7f53b-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2019/file/1e8a19426224ca89e83cef47f1e7f53b-Paper.pdf)
- [36] Timothy Fei Truong Jr. and Tristan Bepler. Poet: A foundation model for high accuracy protein property prediction, Sep 2024. URL <https://www.openprotein.ai/poet-foundation-model-for-high-accuracy-protein-property-prediction>
- [37] The UniProt Consortium. UniProt: the Universal Protein Knowledgebase in 2023. *Nucleic Acids Research*, 51(D1):D523–D531, 11 2022. ISSN 0305-1048. doi: 10.1093/nar/gkac1052. URL <https://doi.org/10.1093/nar/gkac1052>
- [38] Benjamin Buchfink, Klaus Reuter, and Hajk-Georg Drost. Sensitive protein alignments at tree-of-life scale using diamond. *Nature Methods*, 18(4):366–368, Apr 2021. ISSN 1548-7105. doi: 10.1038/s41592-021-01101-x. URL <https://doi.org/10.1038/s41592-021-01101-x>



- [39] Milot Mirdita, Konstantin Schütze, Yoshitaka Moriwaki, Lim Heo, Sergey Ovchinnikov, and Martin Steinegger. Colabfold: making protein folding accessible to all. *Nature Methods*, 19(6):679–682, Jun 2022. ISSN 1548-7105. doi: 10.1038/s41592-022-01488-1. URL <https://doi.org/10.1038/s41592-022-01488-1>.
- [40] Michel van Kempen, Stephanie S. Kim, Charlotte Tumescheit, Milot Mirdita, Jeongjae Lee, Cameron L. M. Gilchrist, Johannes Söding, and Martin Steinegger. Fast and accurate protein structure search with foldseek. *Nature Biotechnology*, 42(2):243–246, Feb 2024. ISSN 1546-1696. doi: 10.1038/s41587-023-01773-0. URL <https://doi.org/10.1038/s41587-023-01773-0>.
- [41] Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4596–4604. PMLR, 10–15 Jul 2018. URL <https://proceedings.mlr.press/v80/shazeer18a.html>.
- [42] 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, Feb 2017. ISSN 1546-1696. doi: 10.1038/nbt.3769. URL <https://doi.org/10.1038/nbt.3769>.
- [43] OpenProtein.AI. [new model] poet. URL <https://github.com/OATML-Markslab/ProteinGym/pull/28>. Accessed: 2025-5-15.
- [44] Aadyot Bhatnagar, Sarthak Jain, Joel Beazer, Samuel C. Curran, Alexander M. Hoffnagle, Kyle Ching, Michael Martyn, Stephen Nayfach, Jeffrey A. Ruffolo, and Ali Madani. Scaling unlocks broader generation and deeper functional understanding of proteins. *bioRxiv*, 2025. doi: 10.1101/2025.04.15.649055. URL <https://www.biorxiv.org/content/early/2025/04/16/2025.04.15.649055>.
- [45] Junming Zhao, Chao Zhang, and Yunan Luo. Contrastive fitness learning: Reprogramming protein language models for low-n learning of protein fitness landscape. *bioRxiv*, 2024. doi: 10.1101/2024.02.11.579859. URL <https://www.biorxiv.org/content/early/2024/02/12/2024.02.11.579859>.
- [46] ESM Team. ESM Cambrian: Revealing the mysteries of proteins with unsupervised learning. <https://evolutionaryscale.ai/blog/esm-cambrian>, 2024. EvolutionaryScale Website, December 4, 2024.
- [47] EvolutionaryScale. Cambrian open license agreement. URL <https://www.evolutionaryscale.ai/policies/cambrian-open-license-agreement>. Accessed: 2025-5-15.