Science
AAAS

Supplementary Materials for

Simulating 500 million years of evolution with a language model

Thomas Hayes *et al.*

Corresponding author: Alexander Rives, arives@evolutionaryscale.ai

Science **387**, 850 (2025)
DOI: 10.1126/science.ads0018

The PDF file includes:

Other Supplementary Material for this manuscript includes the following:

Materials and Methods

A.1. ARCHITECTURE

A.1.1. Notation

In the following, we use L to denote the sequence length, d for the embedding dimension, {a..b} to denote the inclusive set of integers from a to b, and [a,b] an interval of real numbers. SE(3) is the special Euclidean group, which we use to denote frames (Appendix A.1.6.1).

A.1.2. Overview

ESM3 is all-to-all generative model that both conditions on and generates a variety of different tracks. As input, ESM3 is conditioned on various tracks as described in Appendix A.1.5.1, and as output, ESM3 generates predictions detailed in Appendix A.1.5.2.

The generative pipeline is as follows.

Tokenization First, raw inputs are tokenized as described in Appendix A.1.3. Structural inputs are tokenized via a VQ-VAE (Appendix A.1.7). Function keywords are tokenized by quantizing the TF-IDF transform of functional keywords with locality sensitive hashing (LSH), detailed in Appendix A.1.8.

Transformer Trunk A standard Transformer (62, 63) architecture processes the post-tokenized inputs. Geometric Attention (Algorithm 6 and Fig. S2) directly processes structural coordinates as input. Model outputs are logits over token space, and can be sampled to obtain outputs described in Appendix A.1.5.2. The overall architecture is diagrammed in Fig. S1.

Decoder Most tracks can be naively decoded into tokens detailed in Appendix A.1.3. Structure tokens must be decoded with a model—we use a 700M parameter transformer model to do this, trained post-hoc (Appendix A.1.7.2). The decoder uses sequence tokens and structure tokens to directly predict coordinates, pTM, and pLDDT (64). Function tokens are decoded using a small 3-layer transformer, trained post-hoc to invert the LSH quantization procedure (Appendix A.1.8.2.1).

A.1.3. Tokenization

All tracks are represented as a sequence of tokens, with tokens specified in each amino acid position. During tokenization, special beginning-of-sequence (BOS) and end-of sequence (EOS) tokens are prepended and appended.

Sequence Protein sequences are tokenized as the 20 canonical amino acids, plus BOS, EOS, mask, pad, unknown. We keep four non-standard amino acids as in Lin et al. (5), B - Asparagine, U - selenocysteine, Z - glutamic acid, and O - ornithine. This totals to 29 tokens.

Structure Structure tokenization is described in Appendix A.1.7.1. ESM3 uses a codebook size of 4096 with 4 special tokens - EOS, BOS, mask, and pad.

Secondary Structure Secondary structure is taken to be the canonical 8-class tokens (65), with unknown and mask, for a total of 10 tokens. The mask token is forced to be the 0-vector during embedding.

SASA The continuous values representing SASA are tokenized by discretization into a fixed set of 16 bins. SASA bin boundaries were chosen by computing SASA on 100 random structures and ensuring an equal number of residues belong in each bin. Unknown and mask are used for a total of 18 tokens. The mask token is forced to be the 0-vector during embedding.

Function annotations We tokenize function annotations as bags of keywords, described in Appendix A.1.8. Keywords are quantized using LSH into 8 tokens per residue, each of which can be one of 255 tokens. There are three special tokens, empty set, no-annotation, and mask. Again, the mask token is forced to be the 0vector during embedding.

Residue annotations InterPro annotations are tokenized as a multi-hot feature vector (1478 dimensions) over possible InterPro labels (36). Input annotations are limited to a maximum of 16. When annotations are not present, we enforce that the 0-vector is added.

A.1.4. ESM3 Inputs and Forward Pass

As mentioned above, ESM3 can take several tracks, all of which are optionally disabled via masking. In the following, we concisely denote the inputs to ESM3 as

$$\mathbf{x}_{\text{inputs}} = \begin{cases} x_{\text{seq}} \in \{0..29\}^L, x_{\text{structure}} \in \{0..4099\}^L, \\ x_{\text{ss8}} \in \{0..10\}^L, x_{\text{sasa}} \in \{0..18\}^L, \\ x_{\text{func}} \in \{0..258\}^{L\times 8}, x_{\text{res}} \in \{0,1\}^{L\times 1478}, \\ x_{\text{plddt}} \in [0,1]^L, x_{\text{avgplddt}} \in [0,1] \end{cases}$$

We now present the high level algorithm for a forward pass of ESM3:

Algorithm 1 `esm3_forward`

Input: $\mathbf{x}_{\text{inputs}}$

$z^{(0)}_{\text{embed}} =$ `encode_inputs` $(\mathbf{x}_{\text{inputs}})$ $\triangleright \mathbb{R}^{L\times d}$
for $\ell \in \{1..n_{\text{layers}}\}$ **do**
  $z^{(\ell)}_{\text{embed}} =$ `transformer_block` $(z^{(\ell-1)}_{\text{embed}})$
end for
for track in desired output tracks **do**
  $z_{\text{track}} =$ `regression_head` $(z^{(n_{\text{layers}})}_{\text{embed}})$
end for
return Track specific logits $z_{\text{track}} \in \mathbb{R}^{L\times c_{\text{track}}}$

In the next few sections, we detail each component.

A.1.5. Transformer

Our network is based on the transformer architecture (62), incorporating several subsequent improvements: We use Pre-LN instead of Post-LN (63), rotary embeddings (66) instead of absolute positional embeddings, and we replace ReLU non-linearity with SwiGLU (67). The hidden dimension is set to approximately , rounded to the nearest multiple of 256 for training efficiency. No biases are used in linear layers or layer norms, as suggested by PaLM (68). We have observed through the literature and in internal experiments that these architecture changes improve the stability and performance of models.

A core architectural modification we make is the insertion of the Geometric Attention sub-layer in the first block of the network only (Algorithm 2, line 3).

Algorithm 2 `transformer_block`

Input: $x \in \mathbb{R}^{L \times d}, T \in SE(3)^L$

1: $s = \sqrt{\frac{36}{n_{\text{layers}}}}$ ▷$\mathbb{R}$

2: $x = x + s \cdot \text{MultiHeadSelfAttention}(x)$ ▷$\mathbb{R}^{L \times d}$

3: $x = x + s \cdot$ `geometric_mha` (x, T) ▷$\mathbb{R}^{L \times d}$

4: $x = x + s \cdot \text{SwiGLUMLP}(x)$ ▷$\mathbb{R}^{L \times d}$

ESM3-small (1.4B) is a 48-layer network, while ESM3medium (7B) has 96 layers, and ESM3-large (98B) has 216 layers. We experimented with different width-to-depth ratios and observed higher returns for depth than width. Prior work also demonstrates that modalities like ours benefit more from deeper networks (69, 70). Detailed network specifications can be found in Table S1.

A.1.5.1. Embedding

There are 7 unique input tracks to ESM3: (a) sequence (amino acid tokens), (b) structure coordinates, (c) structure tokens, (d) 8-class secondary structure labels (SS8), (e) quantized solvent-accessible surface area (SASA) values, (f) function keyword tokens and (g) residue (InterPro) annotation binary features.

There are two additional tracks used during pretraining only: (h) per-residue confidence (pLDDT) and (i) averaged confidence (pLDDT). At inference time, these values are fixed, and these tracks are equivalent to adding a constant vector z_{plddt}.

Structure coordinates are parsed through the Geometric Attention and are not embedded.

For keyword-based function tokens, each of the eight integers per residue is converted to a "sub-embedding" (Algorithm 3 line 5), then concatenated to form the per-residue embedding (Algorithm 3 line 6). For InterPro residue annotations, the inputs are multi-hot. To create an embedding vector, we sum the embeddings for each of the "on" features (equivalent to the matrix-multiply on Algorithm 3 line 7).

The largest model, 98B, has an additional taxonomy track detailed in Appendix A.1.9.2, only enabled in the final 30K steps of pretraining.

The embeddings are all summed as input to the first layer in the network architecture.

Algorithm 3 `encode_inputs`

Input: $\mathbf{x}_{\text{inputs}} = \{x_{\text{seq}}, x_{\text{structure}}, x_{\text{ss8}}, x_{\text{sasa}}, x_{\text{func}}, x_{\text{res}}, x_{\text{plddt}}, x_{\text{avgplddt}}\}$

$z_{\text{seq}} = \text{embed}(x_{\text{seq}})$ ▷ $\mathbb{R}^{L \times d}$

$z_{\text{structure}} = \text{embed}(x_{\text{structure}})$ ▷ $\mathbb{R}^{L \times d}$

$z_{\text{ss8}} = \text{embed}(x_{\text{ss8}})$ ▷ $\mathbb{R}^{L \times d}$

$z_{\text{sasa}} = \text{embed}(x_{\text{sasa}})$ ▷ $\mathbb{R}^{L \times d}$

$h_{\text{func},i} = \text{embed}([x_{\text{func}}]_{:,i})$ ▷ $\mathbb{R}^{L \times \frac{d}{8}}$

$z_{\text{func}} = [h_{\text{func},1} \mid h_{\text{func},2} \mid \ldots \mid h_{\text{func},8}]$ ▷ $\mathbb{R}^{L \times d}$

$z_{\text{res}} = x_{\text{res}} W_{\text{res}}$ ▷ $\mathbb{R}^{L \times d}$

$z_{\text{plddt}} =$ `plddt_embed` $(x_{\text{plddt}}, x_{\text{avgplddt}})$ ▷ $\mathbb{R}^{L \times d}$

return $z_{\text{seq}} + z_{\text{plddt}} + z_{\text{structure}} + z_{\text{ss8}} + z_{\text{sasa}} + z_{\text{func}} + z_{\text{res}}$

A.1.5.2. Logits

We use a `regression_head` to take in d dimensional last layer hidden features and produce c_{track}-dimensional logits for each of the tracks, where c_{track} corresponds to the size of the vocabulary per track. Note that for the keyword function tokens, we produce $c_{\text{func}} \times 8$ logits, and softmax over each of the 8 independently when calculating the loss.

Algorithm 4 `regression_head`

Input: $x \in \mathbb{R}^{\cdots \times d}$

$z = \text{proj}_{\text{in}}(x)$

$z = \text{GeLU}(z)$

$z = \text{LayerNorm}(z)$

$z = \text{proj}_{\text{out}}(z)$

return z

Except for structure coordinates, we output predictions for each of the tracks detailed in Appendix A.1.5.1: (a) sequence, (b) structure tokens, (c) SS8, (d) quantized SASA, (e) function keyword tokens and (f) residue (InterPro) annotation binary features.

Except for the multi-hot residue annotations, all other tracks are predicted as a categorical distribution over possible tokens.

A.1.6. Geometric Attention

As mentioned in Appendix A.1.5.1, ESM3 processes structural information in two independent ways:

Geometric Attention Described in Algorithm 6, this leverages fine-grained 3D information via conditioning on atomic coordinates of backbone atoms. Coordinates are only used as model inputs.

Structure Tokens Described in Appendix A.1.7, structure tokens enable faster learning due to rich local neighborhood semantics being compressed into tokens. Structure tokens are generally used as model outputs.

Geometric attention enables high-throughput encoding of protein structures. Protein backbone structure can be represented by the relative distance and orientation of frames defined by each residue's backbone coordinates. Reasoning over the relative orientation of frames is important to capture the local backbone geometry when only partial structure is provided. Geometric attention is an $SE(3)$ invariant all-to-all attention mechanism which reasons over the relative distances and orientations of all defined frames in the input (Fig. S2). Because this attention operation can be realized using the same computational primitives as attention, it is readily scalable.

We first provide an overview of frames, and then describe how geometric attention uses them:

A.1.6.1. Frames

Frames are representations that encapsulate the positions and orientations of each residue in the protein. We use a formulation similar to Ingraham et al. (71). Each frame $T \in SE(3)$ consists of a rotation matrix $\mathbf{R} \in SO(3)$ and a translation vector $\mathbf{t} \in \mathbb{R}^3$.

Definition: A frame T_i for residue i is defined as:

$$T_i = \begin{bmatrix} \mathbf{R}_i & \mathbf{t}_i \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \in SE(3)$$

where $\mathbf{R}_i \in SO(3)$ and $\mathbf{t}_i \in \mathbb{R}^3$.

Rotation Matrix: The rotation matrix $\mathbf{R}_i$ for residue i is composed of three 3-dimensional vectors $[\hat{x}, \hat{e}_1, \hat{e}_2]$:

1. $\hat{x}$ and $\hat{e}_1$ are orthogonal unit vectors on the $N - C_\alpha - C$ plane.
2. $\hat{e}_2$ is a unit vector perpendicular to both $\hat{x}$ and $\hat{e}_1$.

This matrix rotates vectors to a local coordinate system where the $N - C_\alpha - C$ plane for the corresponding residue spans the xy plane.

Translation Vector: The translation vector $\mathbf{t}_i$ specifies the position of the residue's C_α.

Transformation: To transform a point $\mathbf{p} \in \mathbb{R}^3$ from the local frame of residue i to the global coordinate system, the following equation is used:

$$\mathbf{p}_{\text{global}} = T_i(\mathbf{p}) = \mathbf{R}_i\mathbf{p} + \mathbf{t}_i$$

Inverse Transformation: To transform a point $\mathbf{p}_{\text{global}} \in \mathbb{R}^3$ from the global coordinate system back to the local frame of residue i, the following equation is used:

$$\mathbf{p} = T_i^{-1}(\mathbf{p}_{\text{global}}) = \mathbf{R}_i^{-1}(\mathbf{p}_{\text{global}} - \mathbf{t}_i)$$

To create frames, all we require is a translation vector $\vec{t}$, and two vectors $\vec{x}$ and $\vec{y}$ defining the local xy plane *after* conversion to global coordinates, from which the frame T can be calculated with the standard Gram-Schmidt algorithm:

Algorithm 5 `gram_schmidt`

Input: $\vec{t} \in \mathbb{R}^{L\times 3}, \vec{x} \in \mathbb{R}^{L\times 3}, \vec{y} \in \mathbb{R}^{L\times 3}$

1: $\hat{x} = \frac{\vec{x}}{\|\vec{x}\|}$
2: $\vec{e}_1 = \vec{y} - (\hat{x} \cdot \vec{y})\hat{x}$
3: $\hat{e}_1 = \frac{\vec{e}_1}{\|\vec{e}_1\|}$
4: $\hat{e}_2 = \hat{x} \times \hat{e}_1$
5: $R = [\hat{x}, \hat{e}_1, \hat{e}_2]$ ▷$SO(3)^L$
6: $T = \begin{bmatrix} R & \vec{t} \\ 0_{1\times 3} & 1 \end{bmatrix}$ ▷$SE(3)^L$
7: **return** T

We construct frames such that the C_α is at the origin of the frame ($\vec{t}$), C on the negative x-axis ($-\vec{x}$), and N is on the xy-plane.

A.1.6.2. Geometric Self-Attention

Algorithm 6 details the Geometric Self-Attention layer. It can be efficiently implemented using similar ideas as FlashAttention (72). It is used twice in our system: in the VQ-VAE encoder for structure tokens (Appendix A.1.7.1), and in the first layer of ESM3.

Unlike regular self-attention, which only operates on per-residue embeddings, Geometric Attention incorporates the per-residue frames T to integrate geometric information in a rotation and translation invariant way. The process of forming the attention matrix A is as follows:

1. **QKV Projections:** Two sets of keys and queries (Q_r,K_r) and (Q_d,K_d), along with V , all with shapes $\in \mathrm{R}^{L\times h\times 3}$ are linearly projected from layer input X. L is the sequence length, h is the number of heads.
2. **Convert QKV to global frame:** Each of the queries, keys and values are initially assumed to be in the local frame of their corresponding residue.
   (a) **Convert to Global Rotational Frame:** We convert each of the vectors in Q_r,K_r,V from their local frame (where the xy plane is the N–C_α–C plane for each residue) to a *global* rotational frame (where the xy plane is aligned for all residues) by applying $\mathbf{R}_i$ (Algorithm 6, lines 3, 4).
   (b) **Convert to Global Distance Frame:** We convert each of the vectors in Q_d,K_d from their local frame to a *global* frame by applying T_i (Algorithm 6, lines 5, 6).
3. **Direction Attention:** The pairwise, per-head h rotational similarity R between keys i and queries j is calculated using the dot product $[R]_{i,j,h} = \frac{1}{\sqrt{3}}[q_r]_{i,h,:} \cdot [k_r]_{j,h,:}$. This is equivalent to the cosine distance between projected points.

4. **Distance Attention:** The pairwise, per-head h distance similarity D between keys i and queries j is computed using the L_2 norm of the difference $[D]_{i,j,h} = \frac{1}{\sqrt{3}} \|[q_r]_{i,h,:} - [k_r]_{j,h,:}\|_2$.
5. **Scale Factor:** R and D are scaled per-head with learned scalars $[\bar{w}_r]_h$ and $[\bar{w}_d]_h$, respectively, where $\bar{w}_r, \bar{w}_d \in \mathbb{R}^h$. We use the softplus function to transform weights into $[0, \infty)^h$. This scaling allows certain heads to specialize in attending via distance or direction attention.

Algorithm 6 `geometric_mha`

Input: $X \in \mathbb{R}^{L \times d}, T \in SE(3)^L$

$Q_r, K_r, Q_d, K_d, V = \text{Linear}(X)$ ▷$(\mathbb{R}^{L \times h \times 3})_{\times 5}$
$(\mathbf{R}_i, \mathbf{t}_i) = T_i$ ▷$(SO(3)^L, \mathbb{R}^{L \times 3})$
$[Q_r]_{i,h,:} = \mathbf{R}_i([Q_r]_{i,h,:})$ ▷$\mathbb{R}^{L \times h \times 3}$
$[K_r]_{i,h,:} = \mathbf{R}_i([K_r]_{i,h,:})$ ▷$\mathbb{R}^{L \times h \times 3}$
$[Q_d]_{i,h,:} = T_i([Q_d]_{i,h,:})$ ▷$\mathbb{R}^{L \times h \times 3}$
$[K_d]_{i,h,:} = T_i([K_d]_{i,h,:})$ ▷$\mathbb{R}^{L \times h \times 3}$
$[R]_{i,j,h} = \frac{1}{\sqrt{3}} [q_r]_{i,h,:} \cdot [k_r]_{j,h,:}$ ▷$\mathbb{R}^{L \times L \times h}$
$[D]_{i,j,h} = \frac{1}{\sqrt{3}} \|[q_d]_{i,h,:} - [k_d]_{j,h,:}\|_2$ ▷$\mathbb{R}^{L \times L \times h}$
$A = \text{softplus}(\bar{w}_r) R - \text{softplus}(\bar{w}_d) D$ ▷$\mathbb{R}^{L \times L \times h}$
$A = \text{softmax}_j(A)$
$[V]_{i,h,:} = \mathbf{R}_i([V]_{i,h,:})$
$O = A \cdot V$ ▷$\mathbb{R}^{L \times h \times 3}$
$[O]_{i,h,:} = \mathbf{R}_i^{-1}([O]_{i,h,:})$
$X = X + \text{Linear}(O)$ ▷$\mathbb{R}^{L \times d}$

In the Geometric Self-Attention layer of ESM3, partially or fully masked coordinates can be input. Undefined coordinates are handled by masking keys and zeroing the attention output wherever coordinates are missing.

A.1.7. Structure Tokenizer

Each residue is associated with one of 4,096 structure tokens (+4 special tokens), designed to provide a rich, learned representation of its local neighborhood. Prior works have shown structure tokens can be used for search (41), to improve representation learning (73), and for generation (16), but we trained our own tokenizer to optimize for structure generation. The tokens are generated with a VQ-VAE encoder, with a corresponding decoder to enable decoding of generated tokens back to 3D coordinates.

A.1.7.1. Encoder

The VQ-VAE encoder f_{enc} consists of two geometric attention blocks (Transformer blocks, but self-attention replaced with geometric_mha) with an embedding width of 1024 and 128 geometric heads per geometric attention layer.

The VQ-VAE encoder reasons over the backbone frames and the relative sequence position of residues in the local structure. Relative sequence positions are encoded through a learned positional embedding. Sequence positions are determined relative to the query residue (i.e., if the query residue has residue index 56, then the residue in index 58 has a +2 sequence position).

Relative sequence positions are clamped to +/- 32 before encoding, meaning long-range contacts share sequence positional embeddings. Relative sequence positional embeddings define the initial encoder state N, and has shape $L \times 16 \times d$ (Algorithm 7, line 4). Note that this means the input to the VQ-VAE encoder is purely structural: no sequence (amino acid), function or other information is used here. Furthermore, each neighborhood is processed completely independently; for each residue, the encoder only uses the information of its 16 nearest neighbors.

Geometric attention blocks operate similar to Transformer blocks in that they transform a state according to an attention operation (geometric_mha) and feedforward network (SwiGLU MLP). As such, the output has the same shape as the input. In this case, this means that the encoder outputs 16 latents per residue. However, we want to learn a single token, i.e., a single latent per residue, hence we take the embedding corresponding to the query residue position $N_{:,0,:}$.

The process of generating structure tokens (Algorithm 7) from the full 3D coordinates of the protein then is as follows:

1. **Local Neighborhood:** For each residue, obtain the indices $N_{\text{idx}} \in \{0..L-1\}^{L\times 16}$ of the 16 nearest residues (as measured by C_α distance). The first of the 16 neighbors is always the residue itself. We also obtain the frames for each residue in a local neighborhood with T_{knn}.

2. **Embed Neighbors:** Embed the relative distance in sequence space for each neighbor, $\Delta i = \text{clamp}(N_{\text{idx}} - i, -32, 32)$ to form $N \in \mathbb{R}^{L\times 16\times d}$.

3. **Encode:** Pass N through a shallow encoder f_{enc} consisting of 2 Transformer blocks, with regular multi-head self-attention swapped with `geometric_mha`. The attention is unmasked, all-to-all over the entire neighborhood.

4. **Quantize:** Extract the first element $N_{:,0,:}$ from the neighborhood, which corresponds to the residue itself. Project it linearly, and quantize by replacing with the nearest vector in a codebook. This yields the structure token per residue.

Algorithm 7 `structure_encode`

Input: $x_{C_\alpha} \in \mathbb{R}^{L\times 3}, T \in SE(3)^L$

1: $N_{\text{idx}} = \text{knn}(x_{C_\alpha})$ ▷$\{0..L-1\}^{L\times 16}$
2: $T_{\text{knn}} = T[N_{\text{idx}}]$ ▷$SE(3)^{L\times 16}$
3: $\Delta i = \text{clamp}(N_{\text{idx}} - i, -32, 32)$
4: $N = \text{embed}(\Delta i)$ ▷$\mathbb{R}^{L\times 16\times d}$
5: $N = f_{enc}(N, T_{\text{knn}})$ ▷$\mathbb{R}^{L\times 16\times d}$
6: $z = \text{Linear}(N_{:,0,:})$ ▷$\mathbb{R}^{L\times d'}$
7: $z =$ `quantize` (z) ▷$\{0..4095\}^L$

A.1.7.1.1. Codebook Learning

`quantize` transforms the L latents into L discrete tokens. Since the VQ-VAE was initially proposed (74), numerous approaches and tricks have been developed to address issues with poor codebook utilization and unstable training. We chose to learn the codebook as an exponential moving average of encoder outputs (74–76). To improve codebook utilization, unused codes are re-initialized to encoder outputs.

A.1.7.1.2. Parallel Encoding

To improve training and inference efficiency, we encode all local structure graphs within a protein in parallel. In practice, this means that given a batch of B proteins with average sequence length L, then the inputs to the structure encoder will have shape $BL \times 16 \times d$.

A.1.7.2. Decoder

While the encoder independently processes all local structures in parallel, the decoder f_{dec} attends over the entire set of L tokens to reconstruct the full structure. It is composed using a stack of bidirectional Transformer blocks with regular self-attention.

As discussed in Appendix A.1.7.3, the VQ-VAE is trained in two stages. In the first stage, a smaller decoder trunk consisting of 8 Transformer blocks with width 1024, rotary positional embeddings, and MLPs is trained to only predict backbone coordinates. In the second stage, the decoder weights are re-initialized and the network size is expanded to 30 layers, each with an embedding dimension of 1280 (~600M parameters) to predict all atom coordinates.

The exact steps to convert structure tokens back to 3D all-atom coordinates using the decoder is provided in Algorithm 8 and detailed as follows,

1. **Transformer:** We embed the structure tokens and pass them through a stack of Transformer blocks f_{dec} (regular self-attention + MLP sublayers, no geometric attention).

2. **Projection Head:** We use a projection head to regress 3 3-D vectors per residue: a translation vector $\vec{t}$, and 2 vectors $-\vec{x}$ and $\vec{y}$ that define the $N - C_\alpha - C$ plane per residue *after* it has been rotated into position. This head also predicts the unnormalized sine and cosine components of up to 7 side chain torsion angles.

3. **Calculate T:** We use `gram_schmidt` to convert $\vec{t}$, $\vec{x}$, and $\vec{y}$ into frames $T \in SE(3)^L$.

4. **Calculate T_{local}:** We normalize the sine and cosine components and convert them to frames $T_{\text{local}} \in SE(3)^{L\times 7}$ corresponding to rotations around the previous element on the side chain.

5. **Compose Frames:** We compose each element of T_{local} with its predecessors on a tree rooted at T to form $T_{\text{global}} \in SE(3)^{L\times 14}$, corresponding to the transformations needed for each heavy atom per residue in atom14 representation.

6. **Apply Frames:** We then apply the frame to the $\overrightarrow{X_{\text{ref}}} \in \mathbb{R}^{L\times 14\times 3}$ coordinates in a reference frame, to rotate and transform each residue into their final positions.

Algorithm 8 `structure_decode`

Input: $z \in \{0..4099\}^{L \times 16}$

$z = \text{embed}(z)$ ▷ $\mathbb{R}^{L \times d}$

$z = f_{dec}(z)$ ▷ $\mathbb{R}^{L \times d}$

$\vec{t}, \vec{x}, \vec{y}, \overline{\sin\theta}, \overline{\cos\theta} = \text{proj}(z)$ ▷ $(\mathbb{R}^{L \times 3})_{\times 3}, (\mathbb{R}^{L \times 7})_{\times 2}$

$T = \texttt{gram\_schmidt}\,(\vec{t}, -\vec{x}, \vec{y})$ ▷ $SE(3)^L$

$\sin\theta = \frac{\overline{\sin\theta}}{\sqrt{\overline{\sin\theta}^2 + \overline{\cos\theta}^2}}$ ▷ $[-1, 1]^{L \times 7}$

$\cos\theta = \frac{\overline{\cos\theta}}{\sqrt{\overline{\sin\theta}^2 + \overline{\cos\theta}^2}}$ ▷ $[-1, 1]^{L \times 7}$

$T_{\text{local}} = \texttt{rot\_frames}\,(\sin\theta, \cos\theta)$ ▷ $SE(3)^{L \times 7}$

$T_{\text{global}} = \texttt{compose}\,(T_{\text{local}}, T)$ ▷ $SE(3)^{L \times 14}$

$\vec{X} = T_{\text{global}}(\vec{X_{ref}})$ ▷ $\mathbb{R}^{L \times 14 \times 3}$

A.1.7.3. Training

When using a VQ-VAE to learn discrete representations which maximize reconstruction quality, it is common to train in the autoencoder in two stages (77). In the first stage, the encoder and codebook is learned with a relatively small and efficient decoder. In the second stage, the encoder and codebook are frozen and a larger or otherwise more computationally expensive decoder is trained to maximize reconstruction quality. We follow this two-stage training approach for the structure tokenizer. The VQ-VAE is fully trained before ESM3 pretraining, with the encoder and decoder kept frozen during ESM3 training.

A.1.7.3.1. Stage 1.

The VQ-VAE is trained for 90k steps on a dataset of single chain proteins from the PDB, AFDB, and ESMAtlas. We use the AdamW optimizer (Loshchilov et al. 2017) with learning rate annealed from 4e-4 according to a cosine decay schedule. Proteins are cropped to a maximum sequence length of 512. Five losses are used to supervise this stage of training. The geometric distance and geometric direction losses are responsible for supervising reconstruction of high quality backbone structures.

Additionally, binned distance and direction classification losses are used to bootstrap structure prediction but are ultimately immaterial to reconstruction. We have found that these structure prediction losses formulated as classification tasks improve convergence early in training. To produce these pairwise logits, we use a `pairwise_proj_head`, that takes $x \in \mathbb{R}^{L \times d}$ and returns logits $z \in \mathbb{R}^{L \times L \times d'}$. It works as follows:

Algorithm 9 `pairwise_proj_head`

Input: $x \in \mathbb{R}^{L \times d}$

$q, k = \text{proj}(x), \text{proj}(x)$

$\text{prod}_{i,j,:}, \text{diff}_{i,j,:} = q_{j,:} \odot k_{i,:}, q_{j,:} - k_{i,:}$

$z =$ `regression_head` $([\text{prod} \mid \text{diff}])$ $\triangleright \mathbb{R}^{L \times L \times d'}$

return z

Finally, an inverse folding token prediction loss (i.e., a cross-entropy loss between predicted sequence and ground truth sequence) is an auxiliary loss used to encourage the learned representations to contain information pertinent to sequence-related tasks.

The five losses are covered in detailed as follows:

1. **Backbone Distance Loss:** Compute the pairwise L_2 distance matrix for the predicted and true coordinates of the 3 backbone atoms (N, C_α, C). Let $D_{\text{pred}}, D \in \mathbb{R}^{3L \times 3L}$. Compute $\left(D_{\text{pred}} - D\right)^2$, clamp the maximum error to $(5\ \text{Å})^2$, and take the mean.

Algorithm 10 `backbone_distance_loss`

Input: $\hat{X} \in \mathbb{R}^{L \times 3 \times 3}, X \in \mathbb{R}^{L \times 3 \times 3}$

$\hat{Z}, Z = \text{flatten}(\hat{X}), \text{flatten}(X)$ $\triangleright \mathbb{R}^{3L \times 3}, \mathbb{R}^{3L \times 3}$

$[D_{\text{pred}}]_{i,j} = \|[\hat{Z}]_{i,:} - [\hat{Z}]_{j,:}\|_2^2$ $\triangleright \mathbb{R}^{3L \times 3L}$

$[D]_{i,j} = \|[Z]_{i,:} - [Z]_{j,:}\|_2^2$ $\triangleright \mathbb{R}^{3L \times 3L}$

$E = (D_{\text{pred}} - D)^2$

$E = \min(E, 25)$

$l = \text{mean}_{i,j}(E)$ $\triangleright \mathbb{R}$

return l

2. **Backbone Direction Loss:** Compute six vectors for both predicted and ground truth coordinates for each residue:

   a. $N \rightarrow C_\alpha$

   b. $C_\alpha \rightarrow C$

   c. $C \rightarrow N_{\text{next}}$

   d. $\mathbf{n}_{C_\alpha} = -(N \rightarrow C_\alpha) \times (C_\alpha \rightarrow C)$

   e. $\mathbf{n}_N = \left(C_{\text{prev}} \rightarrow N\right) \times (N \rightarrow C_\alpha)$

   f. $\mathbf{n}_C = (C_\alpha \rightarrow C) \times (C \rightarrow N_{\text{next}})$

   Compute the pairwise dot product, forming $D_{\text{pred}}, D \in \mathbb{R}^{6L \times 6L}$. Compute $\left(D_{\text{pred}} - D\right)^2$, clamp the maximum error to 20, and take the mean.

   In algorithm form (with `compute_vectors` computing the six vectors described above):

Algorithm 11 `backbone_direction_loss`

Input: $\hat{X} \in \mathbb{R}^{L\times 3\times 3}, X \in \mathbb{R}^{L\times 3\times 3}$

$\hat{V} =$ `compute_vectors` $(\hat{X})$ ▷$\mathbb{R}^{6L\times 3}$

$V =$ `compute_vectors` (X) ▷$\mathbb{R}^{6L\times 3}$

$[D_{\text{pred}}]_{i,j} = [\hat{V}]_{i,:} \cdot [\hat{V}]_{j,:}$ ▷$\mathbb{R}^{6L\times 6L}$

$[D]_{i,j} = [V]_{i,:} \cdot [V]_{j,:}$ ▷$\mathbb{R}^{6L\times 6L}$

$E = (D_{\text{pred}} - D)^2$

$E = \min(E, 20)$

$l = \text{mean}_{i,j}(E)$ ▷$\mathbb{R}$

return l

3. **Binned Direction Classification Loss:** This loss captures a coarser similarity between ground truth and predicted orientations to stabilize early training. It uses the last layer representations of the decoder, not the predicted coordinates. The process is as follows:

    a. **Unit vectors:** Compute three vectors per residue from ground truth coordinates: $C_\alpha \to C$, $C_\alpha \to N$, and $\mathbf{n}_{C_\alpha} = (C_\alpha \to C) \times (C_\alpha \to N)$, and normalize them to unit length.

    b. **Dot Products:** Compute pairwise dot products between each pair of vectors for all residues, forming $D \in [-1,1]^{L\times L\times 6}$. Bin the dot products into 16 evenly spaced bins in $[-1,1]$, forming classification labels $y \in \{0..15\}^{L\times L}$.

    c. **Pairwise Logits:** Pass the final layer representations of the decoder $h \in \mathbb{R}^{L\times d}$ through a `pairwise_proj_head` to obtain logits $z \in \mathbb{R}^{L\times L\times 6\times 16}$.

    d. **Cross Entropy:** Calculate cross-entropy loss using the labels y from the ground truth structure and the logits z, and average over all $L \times L \times 6$ values.

4. **Binned Distance Classification Loss:** Similar to the Binned Direction Classification Loss, this loss bins the true distances between residues (specifically, their C_β) to get ground truth targets and computes a cross-entropy between these targets and pairwise logits. In detail:

    a. **Calculate C_β:** Given the ground truth N, C_α, and C coordinates, we compute the location of C_β:

        i. Obtain the three vectors $N \to C_\alpha$, $C_\alpha \to C$, and $\mathbf{n} = (N \to C_\alpha) \times (C_\alpha \to C)$.

        ii. Define the following scalars:

$$\begin{aligned} a &= -0.58273431 \\ b &= 0.56802827 \\ c &= -0.54067466 \end{aligned}$$

        iii. Compute the location of C_β using the formula:

$$C_\beta = a\mathbf{n} + b(N \to C_\alpha) + c(C_\alpha \to C) + C_\alpha \qquad (1)$$

b. **Pairwise C_β distances:** Compute an $L \times L$ pairwise distance matrix of the C_β, and bin them into one of 64 bins, with lower bounds $[0, 2.3125^2, (2.3125 + 0.3075)^2, \ldots, 21.6875^2]$, forming the labels $y \in \{0..63\}^{L \times L}$.

c. **Pairwise logits:** Pass the final layer representations of the decoder $h \in \mathbb{R}^{L \times d}$ through a `pairwise_proj_head` to obtain the logits $z \in \mathbb{R}^{L \times L \times 64}$.

d. **Cross Entropy:** Calculate the cross-entropy using the labels y computed from the ground truth structure and the logits z, then average over all $L \times L$ values.

5. **Inverse Folding Loss:** Pass final layer representations of the decoder through a regression head to produce logits z. Using ground truth residues as labels y, compute cross-entropy for the classification task of predicting residues from final layer representations.

A.1.7.3.2. Stage 2.

In the second stage of VQ-VAE training, the encoder and codebook are frozen and a new, deeper decoder is trained. This second stage of training has multiple purposes. First, a larger decoder improves reconstruction quality. Second, augmented structure tokens from ESM3 are added to enable learning pAE and pLDDT heads. Third, we add sequence conditioning and train with all-atom geometric losses to be able to decode all-atom protein structures. Fourth, we extend the context length of the decoder to be able to decode multimers and larger single chain proteins.

Training data for stage 2 consists of predicted structures in AFDB and ESMAtlas, as well as single chains, multimers, and antibody-antigen complexes from the PDB. Sequence conditioning was added to the decoder via learned embeddings which are summed with structure token embeddings at the input to the decoder stack.

The structure token decoder was trained in three stages: 2A, 2B, 2C detailed in Table S2. The purpose of stage 2A is to efficiently learn decoding of all-atom structures. Enhanced training efficiency is achieved by keeping a short context length and omitting the pAE and pLDDT losses, which are both memory-consuming and can be in competition with strong reconstruction quality. In stage 2B, we add the pAE and pLDDT losses. These structure confidence heads cannot be calibrated unless structure tokens are augmented such that ESM3-predicted structure tokens are within the training distribution. To this end, for stages 2B and 2C we replace ground truth structure tokens with ESM3-predicted structure tokens 50% of the time. In stage 2C, we extend context length to 2048 and upsample experimental structures relative to predicted structures.

1. **All-atom Distance Loss:** We generalize the Backbone Distance Loss to all atoms by computing a pairwise L_2 distance matrix for all 14 atoms in the atom14 representation of each residue. This results in $D_{\text{pred}}, D \in \mathbb{R}^{14L \times 14L}$. The rest of the computation follows as before: $\left(D_{\text{pred}} - D\right)^2$, clamping to $(5\ \text{Å})^2$, and taking the mean, while masking invalid pairs (where any atom14 representations are "empty").

2. **All-atom Direction Loss:** We extend the Backbone Direction Loss to all heavy atoms:

    a. Identify all covalent bonds between heavy atoms within each residue.

    b. For each atom with at least two covalent bonds, compute a normal (z-axis) vector to the plane spanned by its first two covalent bonds. This results in a set of z-axis and bond vectors.

    c. Compute all-to-all pairwise dot products between z-axis and bond vectors, forming $D_{\text{pred}}, D \in \mathbb{R}^{n \times n}$. Compute $\left(D_{\text{pred}} - D\right)^2$, clamp the max to 20, and take the mean.

3. **pLDDT Head:** Uses a Regression Head with 50 output classes (each capturing 0.02 units from 0 to 100). Predicted structures are compared to ground truth to calculate per-residue pLDDT values, which are supervised with cross-entropy loss.

4. **pAE Head:** Use a Pairwise Projection Head to produce 64 logits per residue pair $\in \mathbb{R}^{L \times L \times d}$, converting to probabilities p via softmax. Each probability corresponds to a bin representing 0.5 Å of positional error, with centers $[0.25, 0.75, \ldots, 31.25, 31.75]$.

    Computing Loss:

    a. Compute the pairwise distances between residues in both the predicted and ground truth structures, resulting in distance matrices D_{pred} and $D \in \mathbb{R}^{L \times L}$.

    b. Calculate the differences $\left(D_{\text{pred}} - D\right)$.

    c. Bin these differences into 64 bins, generating classification targets for the logits.

    d. Compute the loss using cross-entropy between these targets and the logits.

    Computing pAE: Multiply probabilities by bin centers and sum to obtain the expected positional error per residue pair, with values $\in [0.25, 31.75]$.

    Computing pTM: Additionally, the pairwise logits are used to compute the pTM (Predicted Template Modeling) score, as follows:

    a. Compute f_d for sequence length L as:

$$
\begin{aligned}
d_0 &= 1.24 \cdot (\max(L, 19) - 15)^{\frac{1}{3}} - 1.8 \\
f_d &= \frac{1}{1 + \left(\frac{\text{bins}}{d_0}\right)^2}
\end{aligned}
$$

    b. Compute pTM using previously computed probabilities p:

$$\text{pTM} = \max_i \left[\frac{1}{L} \sum_j \left(\sum_{\text{bin}} [p]_{i,j,\text{bin}} [f_d]_{\text{bin}} \right) \right]$$

A.1.7.4. Evaluation

We evaluate the reconstruction quality of the structure tokenizer after stage 1 and stage 2 of training using a set of CAMEO, CASP14, and CASP15 proteins taken after the training cutoff date (Fig. S3). Both decoders consistently reach backbone RMSD < 1Å, LDDT-CA > 0.98. The re-training of the structure token decoder results in substantial improvements in reconstruction quality across all test sets.

The stage 2 decoder, trained with an all-atom reconstruction loss and a sequence input, achieves strong all-atom reconstruction as well (Fig. S3C). We also visualize a random sample of backbone reconstructions on the CAMEO test set (Fig. S4A). Looking at the proteins with worse reconstruction quality, we find that long regions with few tertiary contacts, disordered regions, and unresolved coordinates can lead to inaccurate global orientation of structural elements, while local structure reconstruction remains largely error-free (Fig. S4B). This behavior can be explained by the fact that the tokenizer relies on tertiary contacts to resolve the global orientation of a residue.

We also investigate the vocabulary learned by the structure tokenizer by visualizing the local neighborhoods which map to the same learned structure token. We find that many structure tokens encode semantically coherent sets of local neighborhoods (Fig. S5A). However, a number of tokens appear to represent multiple local neighborhoods (Fig. S5B). While the meaning of a single token may be ambiguous, the high-fidelity reconstruction quality from the decoder suggests that it is able to disambiguate given surrounding context in the full set of structure tokens.

Fig. S6 indicates that pLDDT and pTM have good predictive power. We assess the calibration of the structure confidence heads on the CAMEO test set using structure tokens predicted by ESM3 7B. Most predictions for pLDDT lie along the diagonal, though there is a small bias towards more confident predictions. As pTM is a pessimistic estimator of the TM-score, we find that pTM is biased downwards. Anecdotally, we also find that pLDDT can be poorly calibrated for some generated sequences, particularly in alpha helical regions where it can be an overestimate.

A.1.8. Function Tokenization

ESM3 processes annotations of functional characteristics of proteins through two tracks: function tokens, and residue annotations. Both support input conditioning and output heads for generation. Appendix A.1.5.1 outlines how tokens are processed into the network. We further describe the creation of the tokens in this section.

A.1.8.1. Function Tokens

Function tokens are a dense semantic representation of functional characteristics of proteins derived from free-text descriptions of the InterPro and Gene Ontology (GO) terms at each

residue. At training time, function tokens are produced from each protein's InterPro annotations by a multi-step process illustrated in Fig. S7. At a high level:

1. For each residue, we gather free-text for each InterPro annotation via annotation term names, associated GO terms per annotation (via InterPro2GO mapping), and all ancestor GO terms. All residues with the same InterPro domain are assigned the same set of GO terms. We parse the free-text into counts from a vocabulary of 68,103 keywords. The vocabulary is composed of unigram and bigrams extracted from the free-text of all valid InterPro annotations (and their associated GO/ancestor GO terms) in our training datasets.

2. The keywords are converted to a sparse TF-IDF vector per InterPro annotation. During training, we also produce a corrupted version by dropping keywords at the protein level (i.e. the same keywords have their counts set to 0 across all residues) at a 15% probability per keyword.

3. To create a vector per residue from the per annotation vectors, we max pool the TF-IDF vectors for the annotations per residue. During training, we apply further corruption by dropping annotations at the protein level (i.e. the same annotations are removed from the max pool across all residues) at a 15% probability per annotation.

4. We then quantize each residue's vector (a highly sparse vector with float entries) into a discrete representation suitable for input to the language model as tokens by applying a fixed series of 8 locality sensitive hashes (LSH), each with 8 hyperplanes.

The result is a sequence of 8 tokens each ranging in value from 0 to 255 per-residue. We reserve a special token <none> to represent positions with an empty set of InterPro annotations. For proteins that lack any functional annotations, the tokens are filled with the <pad> token which has an embedding value fixed to all zeros. At test time, we can produce per residue vectors using the process described, or directly creating a TF-IDF vector with keywords.

During pretraining we use the corrupted versions of the function tokens at input, predicting the un-corrupted version function tokens at positions which have been masked. 90% of the time, the entire input is replaced with <mask> . The other 10% of the time, we replace all 8 tokens of selected residue with a <mask>, with the per-residue selection probability sampled from a cosine masking schedule per protein. The model has an output head which predicts each of the 8 function tokens in positions with <mask> as input, and is trained with a categorical cross entropy loss.

Function tokenization offers several key advantages as compared to simpler approaches (e.g., using a dedicated InterPro tag vocabulary). Encoding functional annotations with a generic functional keyword vocabulary supports flexible prompting of the model with combinations of keywords.

Function tokenization can also be viewed through the lens of data compression. This choice of representation reduces the input/output space from all possible InterPro combinations which would naively be represented by 35k bits, to a reduced space of 8 tokens x 8 bits / token = 64

bits. This also affords significant memory saving during pretraining by eliminating the need to perform multi-class multi-label binary classification.

A.1.8.2. Function Prediction

ESM3 is trained to predict all 8 function tokens, each spanning 256 possible values. To extract interpretable predictions of protein function from ESM3, we decode the predicted function tokens into function keywords using a separately trained function token decoder.

A.1.8.2.1. Function Token Decoder

We train a 3-layer transformer model to learn the inverse map of the function tokenization process. The model takes as input the 8 function tokens representing the locality sensitive hash of function keywords. It outputs for each residue the binary-classification predictions predicting the presence of each function keyword, as well as predicting InterPro annotations from which the keywords originate. We find that unpacking the 8-bit LSH tokens into single-bit tokens improves training dynamics of the function token decoder. We train the function token decoder offline using combinations of InterPro tags from the UniRef annotated proteins. Since the function token vocabulary is fixed, the decoder is applied identically across different ESM3 model sizes.

A.1.8.2.2. Evaluation

To evaluate ESM3's performance in predicting protein function, we compute Average Precision, a standard measure of information retrieval, using a validation set of proteins from UniRef and their associated InterProScan function annotations. We present results in Fig. S8.

A.1.8.3. Residue Annotations Track

Residue annotations label a protein's sites of functional residues with a vocabulary of 1474 multi-hot labels emitted by InterProScan. To gather this data, we run InterProScan with databases (SFLD, CDD, PIR) on all cluster members in our UniRef and Mgnify datasets (seq-id 90 clustered). We take all unique residue annotation descriptions that occur in more than 1k proteins across all of UniRef90 and MGnify90, and deduplicate labels by punctuation and case insensitivity. We join these annotations into our UniRef, MGnify, AFDB, and ESMAtlas datasets for training.

As introduced in Appendix A.1.5.1, ESM3 has a track dedicated to processing residue annotations that supports input conditioning, and an output head for prediction and generation. The residue annotation labels for a protein are tokenized into a sequence of token-sets in length equal to the protein. At each position there is an unordered set of tokens representing the residue annotations present at that position. The tokens are input to ESM3 first through an embedding lookup followed by a sum over embeddings. The permutation invariance of the sum retains that the labels are represented to an unordered set as a model. The per-position embedding sums are then added onto the per-position sequence embedding before input into the first transformer block. Positions with no residue annotations are represented by a <pad> token which has an embedding fixed to zeros.

The residue annotations track has an output head which outputs a set of binary classification logits predicting for each position the presence or absence of each residue annotation in the vocabulary. We apply a masking procedure to partially/fully mask residue annotation labels, and train the output head with a binary cross-entropy loss function to reconstruct the full residue annotation. In pretraining, with 90% probability all residue annotations are masked, and otherwise we independently sample positions to mask with a square root schedule.

A.1.9. Other Tracks

A.1.9.1. Confidence Tracks

As mentioned in Appendix A.1.5.1, ESM3 has two additional tracks that are only used during pretraining, and only used as input (we do not have output heads predicting these values). The first is a per-residue pLDDT: for ground truth PDB structures, these values are all 1; for AlphaFoldDB/ESMFold structures, we use the provided pLDDT. We also provide an averaged pLDDT across all the residues when structure is provided (1 otherwise), with the average calculated before any tokens are masked.

This information allows the model to distinguish between gold-standard structures and computationally predicted ones; at inference time, we set these to 1 throughout, with the goal of producing structures better than the computational predictions used to pretrain the model. The embedding itself is straightforward, with the pLDDT values first having a radial basis function, followed by a Linear layer applied to them:

Algorithm 12 `rbf`

Input: $x \in \mathbb{R}^{\cdots \times L}$, $a \in \mathbb{R}$, $b \in \mathbb{R}$, $n \in \mathbb{Z}^+$

$\Delta = \frac{b-a}{n-1}$ ▷ $\mathbb{R}$

$c = [a, a+\Delta, a+2\Delta, \ldots, a+(n-2)\Delta, b]$ ▷ $\mathbb{R}^n$

$\sigma = \frac{b-a}{n}$ ▷ $\mathbb{R}$

$[z]_{\ldots,i,j} = \frac{1}{\sigma}([x]_{\ldots,i} - [c]_j)$ ▷ $\mathbb{R}^{\cdots \times L \times n}$

return $\exp(-z^2)$ ▷ $\mathbb{R}^{\cdots \times L \times n}$

Algorithm 13 `plddt_embed`

Input: $x_{\text{plddt}} \in [0,1]^L$, $x_{\text{avgplddt}} \in [0,1]$

$\text{rbf}_{\text{plddt}} =$ `rbf` $(x_{\text{plddt}}, 0.0, 1.0, 16)$ ▷ $\mathbb{R}^{L \times 16}$

$\text{rbf}_{\text{avgplddt}} =$ `rbf` $(x_{\text{avgplddt}}, 0.0, 1.0, 16)$ ▷ $\mathbb{R}^{16}$

$z_{\text{avgplddt}} = \text{Linear}(\text{rbf}_{\text{avgplddt}})$ ▷ $\mathbb{R}^d$

$z_{\text{plddt}} = \text{Linear}(\text{rbf}_{\text{plddt}})$ ▷ $\mathbb{R}^{L \times d}$

$[z_{\text{plddt}}]_{i,:} = [z_{\text{plddt}}]_{i,:} + z_{\text{avgplddt}}$ ▷ $\mathbb{R}^{L \times d}$

return z_{plddt}

A.1.9.2. Taxonomy Track

The final 30,000 steps in the pretraining of the 98B variant of ESM3 includes a track for processing the taxonomic and species classification of the organism from which the protein sequence originates. For each protein, the taxonomic and species classifications are concatenated to create a full taxonomic lineage. The list of terms is then tokenized using a vocabulary

comprised of the top 40,000 taxonomic terms in the UniRef training dataset. At input, learned embeddings (dimension 768) for each term in the lineage are summed and projected by a learned linear projection to a single embedding of d_{model}. This single embedding is then repeated across the length of the sequence and summed with the embeddings of all other tracks. The linear projection is zero-initialized at the start of this stage of training to preserve model behavior, enabling continuation of pretraining with no degradation in performance.

In pretraining we apply random corruption to the taxonomic lineages and train ESM3 to reconstruct the full lineage by predicting dropped terms with a shallow MLP head trained on the final layer's representations. To corrupt the taxonomic lineage, we either drop all terms (probability 25%) or drop a set of the most specific terms of the lineage of size chosen uniformly at random from 1 to the length of the lineage (probability 25%). We also independently drop any taxonomic term with probability 10%. The output head outputs binary classification logits over the full set of 40,000 taxonomic lineage terms, and is trained to predict the missing terms via a binary-cross entropy loss.

A.1.10. ESM3 Inference

Since ESM3 is a bidirectional transformer capable of representing arbitrary factorizations of the joint probability in any order or combination of tracks, we have significant flexibility during inference. We can generate sequence, structure, or function conditioned on any combination of other tracks. We also have a choice of how much compute to apply during generation.

The usual inference strategy is to fix a prompt (which may be a combination of any of the tracks, either fully or partially specified) and choose a track for generation (which may have been partially specified). When predicting the tokens for the generation track, a number of strategies are possible. Here we discuss two notable strategies. Argmax decoding, which predicts all tokens in the generation track in a single forward pass of the model; this computation is $O(L^2)$ in the length of the protein and is extremely efficient. Iterative decoding, on the other hand, samples tokens one position at a time, conditioning subsequent predictions on those already sampled. The runtime for iterative decoding is $O(L^3)$ in the length of the protein.

Additionally, the number of decoding steps can be chosen at runtime. Argmax decoding is equivalent to decoding in one step, while iterative decoding is equivalent to decoding in L steps. It is possible to select any number of decoding steps between these two extremes to find an optimal tradeoff between computation and accuracy for a particular use case. See 1.3.4 for a case study on structure prediction, in which the generation track is the structure tokens track.

When using iterative decoding, ESM3 further allows flexibility in choosing the next position to decode. We choose the position based off of the logits output of ESM3, and for the results of this paper utilize two strategies: entropy decoding, which chooses the position with the lowest entropy after softmax, or max probability decoding, which chooses the position with the maximum probability. To generate k tokens in one pass, we rank by either entropy or max probability and take the top k positions.

In the following algorithm, assume a single forward pass of ESM3 is a function f of a prompt x, and that we can access the logits of a specific token track through a subscript; e.g. sequence

logits would be $f_{\text{sequence}}(x) \in \mathbb{R}^{L\times 32}$. Furthermore, denote $\pi(\cdot; z)$ as the probability distribution induced by the logits z, including an implicit softmax, and $T \in \mathbb{R}^{L}$ a temperature schedule.

Algorithm 14 `generate` from track

Input: $x_{\text{prompt}}, n_{\text{decode}} \in \{1..L\}, T \in \mathbb{R}^{n_{\text{decode}}}$

$k = L/n_{\text{decode}}$ ▷ # steps to decode at each step
for $s \in \{1..n_{\text{decode}}\}$ **do**
  $z_{\text{logits}} =$ `esm3_forward` (x_{prompt}) ▷$z \in \mathbb{R}^{L\times c_{\text{track}}}$
  $\{p_1, \ldots, p_k\} = \text{CHOOSEPOSITIONS}(z)$
  for $i \in \{p_1, \ldots, p_k\}$ in parallel **do**
    $x_i \sim \pi(x; z_i/T_s)$ ▷Sample i with temp T_s
    $x_{\text{prompt}} = \{x_{\text{prompt}}, x_i\}$ ▷Update prompt
  end for
end for

A.2. TRAINING ESM3

A.2.1. Pretraining Data

A.2.1.1. Sequence Databases

UniRef release 2023 02 is downloaded and parsed from the official UniRef website (78). MGnify90 version 2023 02 is downloaded and parsed from MGnify (38). All nonrestricted studies available in JGI on July 31st, 2023 are downloaded and concatenated into the JGI dataset (79). OAS, which includes over a billion antibody sequences from 80 studies, is downloaded and clustered at 95% sequence identity (39).

A.2.1.2. Clustering

In all cases, data is clustered with mmseqs2 (55), with flags `–kmer-per-seq 100 –cluster-mode 2 –cov-mode 1 -c 0.8 –min-seq-id <seqid>`.

During training we re-weight the data distribution using clustering. We first sample a random cluster at the outer level, and then sample a random member of the cluster. To deduplicate cluster members, we perform a first step of clustering at a high level of sequence identity (e.g. 90%), taking one representative per cluster. We then cluster these deduplicated sequences at a lower level of sequence identity (e.g. 70%). We limit the number of samples within a cluster to 20.

A.2.1.3. Inverse Folding

As data augmentation we train a 200M parameter inverse folding model and use it to create additional training examples.

The inverse folding model uses geometric attention for structure conditioning and an output projection head for the sequence logits, similar to ESM3. Unlike ESM3, the transformer stack alternates between blocks with geometric attention and standard attention. The model is trained on the sequence and structure pairs in PDB, AlphaFold-DB, and ESMAtlas, with the single

training task of (and loss computed on) predicting sequence at the output given structure at the input. Model architecture and training methodology is otherwise substantially similar to ESM3.

This model is used to generate additional sequences corresponding to each structure in the training data for ESM3 (5 sequences per structure for ESMAtlas and AlphaFoldDB, 64 sequences per structure for the PDB). When training ESM3, with 50% probability the original sequence and structure pair is presented to the model as a training example. The other 50% of the time one of the inverse folded sequences is paired with the structure as the training example seen by ESM3.

A.2.1.4. Functional Labels

Functional labels are obtained from InterPro (36) and InterProScan (80), both version 95.0. All annotations for UniProtKB were downloaded from the InterPro website via the 'protein2ipr.dat.gz' file. InterProScan was applied to the entirety of MGnify90 with flags `–goterms –iprlookup –pathways –disable-precalc`. The resultant values are taken as ground truth functional labels for model training.

A.2.1.5. Structural Data

We use all PDB chains, clustered by unique PDB ID and entity ID within the PDB structure. We filter to all structures deposited before May 1, 2020, determined by X-ray crystallography, and better than 9Å resolution (40).

AlphaFoldDB is downloaded as the v4 version specified on their website (4). We notice that structures with high pLDDT are disproportionately alpha helices. Therefore, we ensure globularity by measuring the number of long range (>12 sequence distance) contacts in the chain. If this value is $< 0.5L$ with an L length protein, we omit it from our training set. We also filter out all proteins < 0.7 pLDDT.

ESMAtlas is downloaded as version v0 and v2023_02. Similarly, we use a < 0.7 pLDDT filter. We use a 0.7 pTM cutoff as well to enforce globularity. High pTM structures tends to be more compact.

Structural data also includes any functional labels that exist for the corresponding sequence.

A.2.1.6. Solvent Accessible Surface Area and Secondary Structure

For solvent accessibility surface area, we use the Shrake-Rupley rolling probe algorithm as implemented in biotite (81). This generates a set of real numbers, or a nan value when structural coordinates are not provided. Similarly, SS8 labels are generated using the mkdssp tool (82) and taken as ground truth labels. SS8 and SASA tracks were generated for the AlphaFoldDB and ESMAtlas datasets.

A.2.1.7. Purging of Validation Sequences

We keep track of validation set performance on a set of held out sequences from each training set: UniRef, MGnify, and JGI. In order to properly hold out a sufficiently diverse set of

validation proteins, we first sample 25000 proteins from each set. Then we use mmseqs easy-search to filter out proteins from this set with a 70% sequence identity threshold. We choose the set of proteins from our training set to be the "query" set, and the set of validation proteins as our "target" set for mmseqs. We use the flags --alignment-mode 3 -c 0.8 {cov-mode 0 --max-seqs 300 --max-accept 3 --start-sens 2 -s 7 --sens-steps 3. This query is designed such that early stopping in mmseqs will not affect if we find a hit in the "query" training set.

Train purges are run to generate a list of blacklisted UniRef, MGnify, and JGI IDs, which are removed from the training set.

A.2.1.8. Token Counts

The dataset counts in Table S3 are computed after limiting the large clusters to 20. The number of tokens are computed by multiplying the number of sequences with the average length of the dataset.

In order to compute the approximate number of sequences and tokens seen during training, we first compute the number of times the dataset is repeated at the cluster level. Given the number of repeats, we know the expected number of unique samples seen when sampling with replacement is $n\left(1-\left(1-\frac{1}{n}\right)^{k}\right)$ with a cluster of size n and k items selected. Computing this on the size of each cluster and number of dataset repeats results in the approximate number of tokens presented in Table S4. Our largest model is trained on all of this data, while our smaller models use a portion of it depending on the model's token budget.

A.2.2. Pretraining Tasks

A.2.2.1. Noise Schedule

In the masked generative framework, corruption is applied to each input to the model. To enable generation, the amount of noise applied to an input is sampled from a distribution with probability mass on all values between 0 and 1.

We select various noise schedules for different tracks with several goals in mind. First, ESM3 should see all combinations of tracks as input and output, enabling it to generate and predict based on arbitrary inputs. Second, ESM3 should maintain a balance of strong representation learning and high quality generations. Third, the type of inputs provided should be representative of what users would like to prompt the model with.

In initial experimentation, we found that a fixed 15% noise schedule led to poor generation results, while a linear noise schedule where probability of each mask rate was constant led to good generation but poor representation learning results. We find a good trade-off between representation learning and generation by sampling the noise schedule from a mixture distribution. 80% of the time, the mask rate is sampled from a $\beta(3,9)$ distribution with mean mask rate 25%. 20% of the time, the mask rate is sampled from a uniform distribution, resulting in an average overall mask rate of 30%.

The noise schedules applied to each input are listed in Table S6. For the structure coordinate track, we also modify the masking to be applied as span dropping 50% of the time. This ensures that the model sees contiguous regions of masked and provided coordinates, which better mimics the types of inputs users may provide.

A.2.2.2. Track Dropout

Along with applying noise to each track, we want to ensure ESM3 is able to perform well when some tracks are not provided at all (e.g. to perform structure prediction when no structure is provided as input). We enable this by wholly dropping out some tracks with varying probabilities, listed in Table S6.

A.2.2.3. Loss Weights

Losses for each track are combined with weights listed in Table S6. Losses are also normalized on each rank by their relative number of loss contributing tokens, which helps to reduce gradient noise for tracks which are more sparsely present in pretraining data (e.g., residue annotations).

A.2.2.4. Structure Noise

We apply gaussian noise with standard deviation 0.1 to all coordinates the model takes as input.

A.2.2.5. Tertiary Coordination Sampling

An interesting use case of generative protein models involves conditioning on key structural information, such as an active site, and generating the sequence and structure of a protein that contains this information. It is possible to define an tertiary coordination task as 3 residues which are mutually in contact in structure space ($C\alpha$–$C\alpha$ distance < 6Å), but are distant in sequence space (≥ 10 positions apart) (26). Training on this conditioning may enable the model to better perform the type of tertiary coordination required for active site sampling.

While this task will be sampled with some probability under the standard noise schedules, we also manually sample the task with 5% probability whenever a structure is available. If the task is sampled and a valid tertiary coordination triplet is found, the structure coordinates for that triplet are shown to the model. For each residue in the triplet, the adjacent residues are also independently shown with 50% probability, which leads to a total size of between 3 and 9 residues. All other structure coordinates are masked. Normal masking is applied to the other tracks.

A.2.2.6. Tertiary Interface Sampling

Predicting and generating binding interfaces is another important task for generative protein models. To help with this capability, we add computational data augmentation that simulates the binding interface task.

We define a tertiary interface as one involving a long range contact ($C\alpha$–$C\alpha$ distance < 8Å ,≥ 24 sequence positions). When this task is sampled (5% probability whenever a structure is present) and a long range contact is found, then the chain is split into two chains, each containing one

side of the contact interface. Suppose the contacting positions are given by the indices i, j. Then the first chain will contain residues between $[\text{RANDINT}(1, i-3), \text{RANDINT}(i+3, j-15)]$, while the second chain will contain residues between $[\text{RANDINT}(i+15, j-3), \text{RANDINT}(j+15, L)]$. This ensures there is always a residue gap between the two pseudochains. A chainbreak token "—" is inserted to represent the residue gap.

A.2.2.7. Residue Gap Augmentation

To encourage the model to learn to represent residue gaps using the chainbreak token, we introduce a task which randomly splits a single chain into multiple subchains.

First, a number of chains to sample is sampled from a geometric distribution with probability 0.9, up to a maximum of 9 possible chains. If the number of chains sampled is 1, no additional transformations are applied. A minimum separation of 10 residues between chains is defined. Sequence lengths of the chains along with gaps are sampled from a dirichlet distribution to maintain identically distributed sequence lengths for each chain. This transformation is applied to all samples.

A.2.2.8. Geometric Attention Masking

In the case that multiple chains are provided to the model from either the interface sampling or residue gap augmentation tasks, we mask the geometric attention layer to prevent the model from attending to cross-chain coordinates. This simulates tasks where the structure of individual chains is known, but the interface is unknown.

A.2.3. Training Details

A.2.3.1. Hyperparameters

We train all models using AdamW optimizer (83), with the following hyperparameters: $\beta_1 = 0.9$, $\beta_2 = 0.95$. We use a weight decay of 0.01 and gradient clipping of 1.0. We employ 5K to 20K warmup steps until reaching the maximum learning rate, and utilize a cosine decay scheduler to decay LR to 10% of the maximum learning rate by the end of training.

A.2.3.2. Infrastructure

Our training codebase uses Pytorch. We use Pytorch's FSDP (84) implementation for data parallelism. We also use custom components from the TransformerEngine (85) library.

We have made several optimizations to increase the training speed of our models. For multi-head attention uses, we use the memory efficient implementation from the xformers library (86). We also save activations that are expensive to compute during training when necessary. We employ mixed precision training, utilizing FP8, BF16, and FP32 as needed based on accuracy requirements and kernel availability throughout our network.

A.2.3.3. Stability

Scaling ESM3 to 98 billion parameters with its novel architecture, multi-modal inputs, and low precision computation requirements poses significant training stability challenges. Our model is significantly deeper than its NLP counterparts, and literature has shown that deeper networks are harder to train due to attention collapse (87).

We observed training instability early in the architectural innovation phase, which we addressed through several changes. We apply layer normalization to the query and key vectors within the attention mechanism (88). We observe longer warm up helps (89). Another source of instability is the masking rate in pretraining tasks. We found that a very high masking rate is more likely to cause training divergences than a lower one, especially early in the training. Choosing a masking schedule biased towards lower mask rates improved both performance and training stability. Interestingly, the introduction of conditioning from other modalities also improves training stability, perhaps suggesting that stability is related to the degree of under-specification of a task.

An incorrectly set learning rate is another source of instability. To ensure the right balance between learning effectiveness and stability, we optimized the learning rate on smaller models and scaled it according to best practices as outlined in (90, 91). We find empirically that the initialization has a small effect on model stability, and the majority of stabilization can be gained from simply scaling the learning rate at the appropriate rate. By applying the rules in both width-μP and depth-μP, we can simply scale the learning rate inversely proportional to the square root of the number of parameters, and find this results in stable training.

Following these modifications, we successfully trained our 98-billion-parameter model without any issues related to training instability.

A.2.3.4. Staged Training

We stage training to alter dataset composition, train on longer contexts that would be too expensive for the entire pretraining, or introduce features such as the taxonomy track (A.1.9.2).

A.3. MODEL EVALUATIONS

ESM3 is both a generative model and a representation learning model that can be adapted for predictive tasks. In this section, we present benchmarking results for both capabilities.

A.3.1. Models

ESM3 models are trained at three scales—1.4B, 7B, and 98B parameters—on approximately 75B, 560B, and 1.8T training tokens, respectively.

The ESM3 1.4B model, trained on 75B tokens and noted for its small size and speed, allows rapid iteration both during training and at inference. Optimal model size and number of training tokens are studied by extrapolating from a series of smaller runs, given a training compute budget, model architecture, and dataset characteristics (22, 24). After determining compute optimality for training, a variety of factors such as release frequency, amount of inference, ease of use, and usage patterns are also taken into account to determine the ideal number of tokens on which to train the model. To enable efficient inference for the benefit of the research community, we have trained two additional versions of ESM3 1.4B, named 1.4B Overtrained

and 1.4B Open, which are trained on 300B tokens, far beyond their compute optimality for training.

A.3.2. Evaluation Data

In the following benchmarks for this section, unless otherwise noted, models are evaluated on a test set of 902 proteins whose structures are temporarily held out from the ESM3 training set. The proteins were sourced from the Continuous Automated Model EvaluatiOn (CAMEO) targets released from May 1, 2020 through Aug 1, 2023 (92). A summary of the datasets used in each model evaluation is provided in Table S7.

For contact and structure prediction evaluations, we also evaluate on the CASP14 (71 proteins) and CASP15 (70 proteins) structure prediction benchmarks (93, 94). The CASP14 and CASP15 sets are obtained directly from the organizers and include both free modeling and template-based modeling single domain targets. Both CASP sets are temporally held out from the PDB training set, which has a cutoff date of May 1, 2020.

A.3.3. Representation Learning

The contact prediction model is a multilayer perceptron (MLP) head that operates independently over the representations of each amino acid pair, outputting the probability of contact between them. We use LoRA (95) for finetuning, which is a common alternative to full weight finetuning that uses much less memory while attaining strong performance. LoRA is applied to the base model for finetuning, and the MLP along with the LoRA weights are trained end-to-end using the cross-entropy loss with respect to the ground truth contact prediction map. For the ground truth, all residues at least 6 positions apart in the sequence and within an 8Å Cα-Cα distance are labeled as a contact. All models are trained with LoRA rank 4, batch size 64 and a learning rate of 1e-3 for 10k steps on a mix of sequence and structure data from PDB, AlphaFold-DB, ESMAtlas, and OAS Predicted Structures. Data are sampled in a ratio of 1:3:3:0.03 from these datasets.

Table S8 shows the performance on each structural test set through the metric of precision at L (P@L), which evaluates the precision of the top-L most confident predictions, where L is the length of the protein. The smallest ESM3 model, with 1.4B parameters, achieves a P@L of 0.76 ± 0.02 on the CAMEO test set, which is higher than the 3B parameter ESM2 model (0.75 ± 0.02). Furthermore, it trains on an order of magnitude less compute during pretraining (6.72×10^{20} FLOPs vs. 1.8×10^{22} FLOPs), demonstrating the benefits of multimodal pretraining.

A.3.4. Structure Prediction

ESM3 can directly predict protein structures without additional finetuning by first predicting structure tokens, then decoding these tokens into coordinates. When predicting structure tokens, we follow the strategy outlined in Appendix A.1.10 and test both single-pass argmax decoding and full iterative decoding.

Iterative decoding and single-pass decoding have equivalent performance for structure prediction. We hypothesize that this is because ESM3 has a latent representation of structure, so whether structure tokens are decoded in a single pass or iteratively does not meaningfully change

the sampling trajectory. Furthermore, ESM3 was trained with dropout of the structure tokens track, which mimics single-pass decoding (see Appendix A.2.2).

The fidelity of the representation of structure in the hidden states of the language model increases with scale with the ESM3 1.4B having P@L of 0.76, ESM3 7B having 0.82, and ESM3 98B having 0.85 respectively (CAMEO; Table S8). By comparison, ESM2 has 0.75 P@L on CAMEO (Table S8). A similar improvement with scale in the ESM3 models is observed when evaluating the decoded structure tokens, with ESM3 1.4B having LDDT-CA of 0.777, ESM3 7B having 0.848, and ESM3 98B having 0.879 respectively (single pass inference, CAMEO, Table S9). Structure prediction scaling curves as a function of training compute, are shown in Fig. S10. Note that ESMFold which incorporates representations from ESM2 3B has 0.861 LDDT. The performance of ESMFold on this benchmark reflects the effect of fine-tuning for structure prediction using a specialized structure prediction head, while the ESM3 models are evaluated without fine-tuning directly on their decoded structure token outputs.

A.3.5. Conditional Likelihood

The conditional likelihood of an output given a prompt serves as a proxy for the generative capabilities of a model. Fig. S11 and Table S10 evaluate the performance of ESM3 as a conditional generative model, using its negative log-likelihood (NLL) on the test set. For each track—sequence, structure, function, SASA, and secondary structure—NLL is evaluated both unconditionally and conditioned on each of the other tracks.

Unlike, for example, an autoregressive model, ESM3 is a generative model over masking patterns, so is trained to predict tokens given any masking pattern. The NLL of a sample under ESM3 is given by $\frac{1}{L!}\sum_{o\in\mathbb{O}}\frac{1}{L}\sum_{i=1}^{L}\log p\left(x_{o_i}|x_{o_1},\ldots,x_{o_{i-1}}\right)$, where O is the set of all decoding orders with normalization constant $Z=\frac{1}{L!}$. This computation is intractable (as the set of all decoding orders is exponential in length of a protein), but can be approximated by sampling a single decoding order o for each x in our dataset. At each step, teacher forcing is used to replace the masked token with the ground truth token and report the mean NLL over the output tokens.

We observe a number of interesting trends in these results. As expected, the unconditional NLL (Fig. S11, black lines) is always higher than conditional, and conditioning on full 3D structure reduces the loss on secondary structure prediction to nearly zero (1.4B: 0.24, 7B: 0.19, 98B: 0.16). Conditioning on sequence results in a lower structure prediction loss than conditioning on secondary structure (98B; sequence: 3.13, secondary structure: 3.37). We observe a clear log-linear relationship between pretraining FLOPs and NLL for the sequence track, under each form of conditioning.

To better understand the effect of function conditioning, we plot the sequence token NLL gap under function conditioning across the range of mask rates (Fig. S12). We observe that the NLL gap is larger at higher mask rates (e.g., 0.20 at a 90% mask rate) but is negligible at lower mask rates. These results suggest that at low mask rates, there is sufficient information in the sequence for the model to infer the remaining tokens without needing additional information from the function track. We further analyze the function-conditioned NLL gap by InterPro tags, finding that tags for more common domains show larger NLL gaps. For instance, IPR015426 has a mean

NLL gap of 0.59, comparable to the gap between sequence and structure conditioned sequence NLL in the 7B model (0.61).

A.3.6. Unconditional Generation

To assess the model's unconditional generation capability, we sampled 16 proteins for each length L sampled uniformly every 4 values between 64 and 1024 (i.e. {64, 68, 72, ..., 1024}) using ESM3 98B with a constant temperature of 0.7. Structures for each sequence were predicted using ESM3 7B, and the resulting structures have high confidence under ESM3, even for sequences over 1000 amino acids in length, as shown in Fig. S14A. The distribution of pTM and pLDDT are shown in Fig. S14B. ESM3 generates more high-quality structures than ESM2, which was trained using a simple MLM objective over sequence only with a fixed mask rate.

We use pTM for evaluating structure predictions from ESM3 in addition to pLDDT. This is because pLDDT can be miscalibrated for generated structures and can overestimate the confidence of a prediction. pLDDT is biased towards local structural confidence, which can result in pathologies such as very long alpha helices with high pLDDT at all positions. pTM is a more global measure of structural confidence, and is more robust to these pathologies. Fig. S13 shows that pTM and pLDDT correlation drops for generated sequences (Pearson r: natural = 0.85, generation = 0.79), and a clear pattern of high pLDDT (> 0.8) but low pTM (< 0.6) emerges.

To assess whether ESM3 is biased towards particular secondary structures, we use DSSP to predict the three-class secondary structure of the high-confidence (pTM > 0.8, mean pLDDT > 0.8) generations and measure the percentage of residues that form alpha helices and beta sheets. When compared to a background distribution computed over the PDB, we find that ESM3 closely matches the secondary structure distribution of known proteins (Fig. S14D), unlike other methods which preferentially generate helical structures (15, 26, 28). Finally, to confirm that the structures predicted with high confidence by ESM3 are designable, we inverse folded and re-folded each using ESM3 7B. The majority of generations successfully re-folded with TM-score of greater than 0.8 to the hallucinated structures, demonstrating that ESM3 has high self-consistency for its own high-confidence designs (Fig. S14C).

To visualize the distribution of unconditional generations, we generate a larger dataset by sampling 100 protein lengths randomly from the PDB and generating 1,024 sequences for each, again using ESM3 98B with a constant temperature of 0.7. We compute sequence embeddings by extracting the final layer outputs produced by running ESM3 7B with sequence inputs only. Protein-level embeddings are computed by averaging over all positions in the sequence to produce a 2560-dim embedding. We then project these embeddings into two dimensions using a UMAP projection (96) fit on a background distribution of 50,000 randomly sampled sequences from UniProt with minimum distance 0.1 and number of neighbors 25. Examples are selected by computing structural clusters with Foldseek-cluster (using default parameters) and sampling the example with highest ESM3 pTM from each cluster. A subset of these cluster representatives are shown in Fig. 1E. Sequence similarity to the training set was computed using mmseqs2 (55) with the following parameters: `--cov-mode 2 -c 0.8 -s 6.0`. Proteins generated unconditionally are similar—but not identical—to proteins found in the training set (Fig. S16) and have high coverage of the training set (Fig. 1E), demonstrating that the model has properly fit the training

distribution and does not exhibit mode collapse. We observe a cluster of generations with very high sequence identity to the training set; these correspond to antibody sequences, with the framework regions accounting for the high sequence identity.

To explore alternative ways of generating proteins, we assess the quality of proteins generated by a chain-of-thought (CoT) procedure in which ESM3 7B generates the secondary structure (SS8 tokens), then the 3-D backbone coordinates (structure tokens), followed by the amino acid sequence (sequence tokens) (Fig. S15). We compare the quality of amino acid sequences generated from this CoT procedure with the above method of unconditionally directly generating amino acid sequences. We find that the CoT procedure generates sequences that have higher confidence ESM3predicted structures than the directly-generated sequences as measured by pTM and mean pLDDT (Fig. S15A). Compared to high-confidence (pTM > 0.8, mean pLDDT > 0.8) directly-generated sequences, the high-confidence subset of CoT-generated sequences are also more designable: the CoT-generated sequences have predicted structures whose inverse folded, then re-refolded structures have higher TMscore to the originally predicted structure (Fig. S15C). The CoT-generated sequences show a small bias towards higher alpha and beta proportion compared to those generated directly (Fig. S15D).

A.3.7. Prompt Construction

Prompts for ESM3 specify inputs for each track in every position along the sequence length of the protein, which is determined by the positions of BOS and EOS tokens. ESM3 can scaffold motifs or hotspots with unknown relative positions by shuffling their positions in the prompt and generating multiple samples.

A summary of prompt variation and folding methods for each design task is provided in S14.

A.3.8. Prompt-following Evaluations

To evaluate ESM's ability to follow prompts, we use a set of held-out proteins as described in Appendix A.3.2. The test set is further filtered to remove proteins with length greater than 1024, which removes 7 proteins from the test set. To construct prompts for the structure coordinate, secondary structure, and SASA tracks, we sample a random span of length 15% of the original protein length. The model is then shown the corresponding track for the randomly sampled span, and is tasked with generating the sequence for the entire protein. For example, for the structure track, for a protein of length 100, we may sample a random span of 15 residues from residue 20-35. The model would then have to generate a protein sequence of length 100 conditioned on structure coordinate conditioning from residues 20-35 derived from the original test protein. This same procedure is applied for the secondary structure and SASA tracks. For the function track, we form the prompt by tokenizing the keywords from the InterProScan annotations associated with each sequence. The ESM3 7B model is used for all generations with a temperature of 0.7 and L decoding steps (where L is the length of the sequence). The model generates 64 sequences per prompt, which we use to compute pass@64.

To evaluate the generations, we use ESMFold to fold the sequences generated by ESM3. For the structure coordinate, secondary structure, and SASA tracks, alignment metrics (backbone cRMSD, 3-class secondary structure accuracy, and SASA Spearman ρ) can be calculated using

the relevant span in the ESMFold-predicted structure and the original template protein. Continuing the previous example for the structure track, we would compute the RMSD between residues 20-35 in the ESMFold structure predicted of the ESM3-generated sequence and residues 20-35 of the original test protein. For the function annotation track, we run InterProScan (36) on each generated sequence and extract function keywords from the emitted annotations. We report function keyword recovery at the protein level, computing the proportion of all function keywords in the prompt which appear anywhere in the function keywords from the InterProScan annotations of the generation.

A.3.9. Steerable Design

To test the ability of ESM3 to generalize beyond its training distribution under prompting, we evaluate two prompting scenarios. First, we identify proteins which were deposited in the PDB after our training cutoff (December 2020) and choose eight with TM < 0.7 to any structure in our training dataset (PDB IDs: 2JVN chain A, 2KAF chain A, 2L8K chain A, 2MJM chain A, 7ZUO chain A, 8EXF chain B). Using DSSP, we compute the residue-level SS8 and SASA for each of these proteins to prompt ESM3, masking all other tracks. We fold generated sequences with ESM3 7B. We show in Fig. S16A that the generated proteins are diverse, globular, and closely follow the SS8 and SASA prompts while having no close sequence or structure neighbors in the training set.

Second, we classify the residue-level secondary structure for a set of eight symmetric protein backbones using DSSP. These proteins were previously designed using ESMFold (5, 97) and have varying secondary structure (alpha and beta) and varying symmetries (5-fold and 8-fold). Again, ESM3 is able to design these proteins successfully with high confidence (pTM > 0.8, pLDDT > 0.8) and low sequence similarity to the training set Fig. S16B. The structural similarity is moderate for these designs due to the high structural conservation of the protomer units in each design. All designs are generated using a constant temperature of 0.7 with L/2 decoding steps, where L is the protein length. Again, we fold generated sequences with ESM3 7B. We sample 256 sequences for each prompt and filter generations by pTM (> 0.8), pLDDT (> 0.8), and accuracy in satisfying the SS8 prompts (SS8 accuracy > 0.8). Final examples were selected from these filtered designs by visual inspection. Sequence similarity to the training set was computed using the same procedure as the unconditional generations, and structure similarity was computed using Foldseek (41) in TM-score mode (`alignment-type 1`) with sensitivity `-s 7.5`.

A.3.10. Composing Prompts

ESM3 is able to compose multimodal prompts across its input tracks—sequence, structure, SS8, SASA, and function keywords—to generate proteins with novel characteristics. To demonstrate this, we augment the standard functional motif scaffolding task (i.e., partial structure and sequence prompts) with additional conditioning to specify the type of scaffold for ESM3 to design. The functional sites comprise a combination of ligand binding sites coordinated by residues remote in sequence and those defined by short local motifs. For each motif, the coordinates and amino acid identities of all residues from the reference PDB structures are input to the model, with random shuffling and augmentation of the gaps between each active site. See Appendix A.4.5 for a description of this augmentation procedure and the specifications of the

ligand-binding sites chosen. In addition to these sites, we also create a set of 12 partial sequence and structure prompts derived from conserved functional motifs (Table S11). These motifs are defined using a combination of the benchmark dataset in Watson et al. (26) and conserved sequence patterns from the Prosite database (98).

The scaffold conditioning is defined using either SS8 tokens (to specify secondary structure composition) or function keywords defined by InterPro accession numbers (to specify a particular fold). For each combination of functional site and scaffold prompt, we sample between 256 and 2048 times to generate proteins with diverse and novel characteristics. All designs were generated with the 7B-parameter model, a constant temperature of 0.7, and $L/2$ decoding steps for a protein of length L.

Secondary structure prompting. We generated proteins under four main classes of secondary structure composition: mostly alpha helices, mostly beta sheets, and mixed alpha-beta proteins (split into alpha/beta, alpha/beta/alpha, and beta/alpha/beta topologies). For each generation, we prompt the model with a random set of SS8 spans up to a total length L, with mask tokens in between. For example, an all-alpha SS8 prompt for a protein of length L=20 might look like __HHHH___HHHHH____HH and a beta-alpha-beta prompt might look like __EEE___HHHHH____EE_, where H is a residue within an alpha helix and E is a residue in a beta strand. We then combine this with the augmented partial structure and sequence tracks given by a functional site motif. To increase the diversity of the scaffolds and maximize the probability of generating physically realizable prompt combinations, we generate between 256 and 1024 designs for each combination of SS8 and functional site motif. For each generation, we uniformly sample a random length L between 150 and 400. Then, we produce a set of secondary structure spans with length 5-20 residues, each separated by a gap of 3-10 residues, such that the total length adds up to L. Finally, to avoid incompatibility between the partial structure and secondary structure constraints, we also mask the SS8 tokens at positions where structure is specified by the functional site prompt. Secondary structure–prompted designs were assessed by running DSSP on the designed sequence and measuring the fraction of prompted residues which were assigned the correct secondary structure. Success was determined by a pTM > 0.8, all-atom cRMSD < 1.5Å for the functional site, and SS8 accuracy > 0.8.

Keyword prompting. To prompt the model to generate proteins with a specific fold, we extracted the set of InterPro tags associated with a set of proteins from the test set for which ESM3 achieved keyword recovery of greater than 80% (Fig. 2A). These tags were then converted into keywords and used to prompt the model in combination with the partial sequence and structure constraints. The list of prompts and function tags is given in Table S12. Keyword-prompted designs were assessed using a self-consistency evaluation, i.e. whether the model successfully predicts any of the prompted InterPro accessions for the designed sequence. Success was determined by a pTM > 0.8, all-atom cRMSD < 2.0, and number of InterPro accessions recovered > 0.

We assess novelty of each motif-scaffold combinations by measuring the TM-score between the generated scaffold and the chain from which the motif is derived (Table S13). This confirms that the model is not retrieving the original motif scaffold, particularly for secondary structure–prompted scaffolds where we do not provide any explicit instructions to produce diverse designs. For the motifs derived from ligand binding residues (magnesium, serotonin, calcium, zinc,

protease inhibitor 017, and Mcl-1 inhibitor YLT), we additionally use Foldseek to search the PDB for any other proteins which share that motif (as defined by BioLiP (99)), as a more stringent evaluation of novelty. For all but zinc-binding and magnesium-binding motifs, Foldseek finds no significant hits at an E-value threshold of 1.0. The hits discovered for zinc and magnesium have only modest TMscore (0.76 and 0.64), demonstrating that the model still finds novel scaffolding solutions for these ligands. To assess whether the generated scaffolds are likely to be designable, we measure a self-consistency TM-score (scTM) under orthogonal computational models by inverse-folding the designed structure with ESM-IF1 (42) (using a temperature of 0.5) and re-folding with ESMFold (5). We report the best scTM over 8 inverse folding designs in Table S13.

A.3.11. Multimodal Editing Examples

First, we describe the procedure for generating the protein compression example shown in Fig. 2D. A series of prompts of length 150 were constructed. The sequence and structure of the catalytic triad of trypsin (PDB 1Y3V) (H57, D102, S195) were placed in the prompt using the following procedure: three random residue numbers between 20 and 130 were sampled such that the minimum pairwise difference in position between each of the residues was no less than 20. Then, H57 from the template trypsin was placed at the lowest sampled number, D102 at the second lowest, and S195 at the largest number, thus respecting the left-to-right ordering of the catalytic triad in the template trypsin. 128 prompts were generated by this procedure. Each of these prompts was combined with a function keyword prompt derived from the template protein, specifically InterPro (36) tags IPR001254 (serine proteases, trypsin domain) and IPR009003 (peptidase S1, PA clan), to arrive at a final set of 128 prompts. The base ESM 7B model was then prompted to generate the sequence of the remaining 147 residues of the protein conditioned on the randomly placed catalytic triad sequence and structure coordinates and function keywords. $L = 150$ decoding steps were used with a temperature of 0.7, with 32 generations per prompt. Generations were then filtered by all-atom RMSD at the prompt positions, ESM3 pTM, and InterPro Scan keyword outputs, with the generation shown in Fig. 2D selected finally by visual inspection.

Generation quality was measured using ESMFold (5) pTM of the generated sequence, in addition to self-consistency. For self-consistency, we inverse fold the ESM3-predicted structure of the generation with ESM-IF1 (42) 8 times and re-fold with ESMFold, reporting the mean and std of the TM-scores between the 8 ESMFold-predicted structures and the ESM3-predicted structure. To perform a blast search of the sequence, we use a standard Protein Blast search (54). We set the max target sequences parameter to 5000 and sort results by sequence length and sequence identity, selecting the first sequence that is a serine protease. This yields the reference WP 260327207 which is 164 residues long and shares 33% sequence identity with the generation.

We showcase two further examples of protein editing. First, ESM3 is prompted to bury an exposed helix in a protein with an alternating alpha-beta sandwich fold. The prompt is constructed as follows: the prompt is of the same length as the template protein (PDB 1LBS). We identify a buried helix (mean SASA 0.32 Å^2) between residues 106-116 of the template protein. Structure coordinates from this region are placed in the prompt at the same residue indices, to prompt ESM3 to generate the same helix. This is composed with a SASA prompt of

40.0 for each of the 11 helix residues, prompting ESM3 to place this helix on the surface of the protein. Finally, we prompt with the secondary structure of 5 central beta strands surrounding the buried helix, residues 33-36, 62-65, 99-103, 125-130, and 179-182. ESM3 7B is then used to generate 512 protein sequences conditioned on this prompt using $\frac{L}{2}$ decoding steps and a temperature of 0.7. Designs are filtered by ESM3 pTM and adherence to the SASA prompt. The final generation is chosen by visual inspection. The generation is evaluated as described above (ESMFold pTM 0.71, scTM mean 0.82, std 0.045). Examining the generation, ESM3 is able to satisfy the input constraints: the generated protein maintains the structure of the helix (backbone cRMSD 0.18 Å) and the alternating alpha-beta fold (both the generation and the template have 7 strands alternating with helices), while exposing the helix motif to the surface (mean SASA 28.35 Å^2). Furthermore, the generation is structurally distinct: a Foldseek search (41) of AlphaFold-DB, ESMAtlas, and PDB in TM-align mode reveals no hit with TM-score greater than 0.76.

We also use ESM3 to generate an idealized TIM Barrel with 11-fold symmetry. This generation is undertaken in two steps. First, we derive a secondary structure and function keyword prompt from a reference TIM Barrel (PDB 5EKY). The secondary structure of the reference protein is computed using DSSP and then idealized to construct a prompt for ESM3. To construct the secondary structure prompt, the length of each helix and strand is fixed at 7 residues. Each helix and strand region is then separated by 3 mask tokens, with a mask token appended to the N and C termini of the prompt as well. This yields a secondary structure prompt of total length 159, which is combined with a function keyword prompt derived from the reference protein: keywords are derived from IPR013785 (aldolase-type TIM barrel) and IPR000887 (KDPG/KHG aldolase). ESM3 7B is then used to generate 256 samples with L decoding steps and a temperature of 0.7. The design shown is chosen by filtering by ESM3 pTM and visual inspection. In the second step, the secondary structure prompt from the first step is expanded to contain 11 helix-strand subunits, for a total prompt length of 225 residues (4 mask tokens are now appended to the N and C termini, rather than just 1). ESM3 7B is then used to generate 256 samples with L decoding steps and a temperature of 0.7, with generations filtered by ESM3 pTM and visual inspection. The generation is evaluated as described above (ESMFold pTM 0.69, scTM mean 0.97, std 0.011). The generation is structurally distinct: a Foldseek search (41) of AlphaFold-DB, ESMAtlas, and PDB in TM-align mode reveals no hit with TM-score greater than 0.61.

A.4. ALIGNMENT

A.4.1. Algorithm

Since the introduction of RLHF (44) there have been a number of algorithms developed to tune large models trained via unsupervised learning to better follow instructions and generally align their generations to user preferences (45, 46, 100, 101). We use IRPO (Iterative Reasoning Preference Optimization) due to its simplicity in implementation and good performance. The IRPO loss combines supervised finetuning with contrastive learning from preference pairs. IRPO operates on a dataset $D \sim (y_w, y_l, x)$ consisting of prompt x and a pair of completions y_w (preferred) and y_l (not preferred). It also operates on two separate models: the reference model π_{ref} and the current model π_θ. The reference model π_{ref} is the fixed base model of the same scale, and the current model π_θ is the model being optimized.

$$\mathcal{L}_{\text{IRPO}}(\pi_\theta; \pi_{\text{ref}}) = \mathcal{L}_{\text{NLL}} + \alpha\mathcal{L}_{\text{DPO}} = \\ -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\frac{\log \pi_\theta(y_w|x)}{|y_w|+|x|} + \right. \\ \left. \alpha \log \sigma\left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right] \quad (2)$$

The IRPO loss contains two terms. The L_{NLL} term maximizes the log likelihood of the preferred example normalized by the length of the sequence, providing signal to reinforce the good generations from the model. The L_{DPO} term is the contrastive preference tuning term, which increases the difference in log likelihoods between the preferred and not preferred examples while staying close to the reference model (45). The use of the reference model serves as a regularizer to prevent overfitting to the preference dataset, which can often be small. There are two hyperparameters, α and β. α weights the relative importance of the supervised with the preference loss and the β parameter controls how close we stay to the reference model: the higher the beta, the closer we stay. We minimize this loss with respect to the current model parameters θ.

ESM3 is a multi-modal model so the prompt can be any combination of the input tracks of (partial) sequence, structure, and function and the generation y can be any of the output tracks. In our experiments we always generate the amino-acid sequence so this will be our running example from now on. Since an amino-acid sequence y can be generated from prompt x in many multi-step ways computing the full likelihood $\pi(y|x)$ would involve integrating over all possible multi-step decoding paths. Since this is intractable, we use a surrogate that mirrors pretraining, shown in Eq. (3) and described below.

$$\log \pi(y|x) \approx \mathbb{E}_m\left[\sum_{i\in m} \log p(y_i|y_{\setminus m}, x)\right] \quad (3)$$

To approximate the likelihood of a generation y from prompt x, we mask y with a mask sampled from a linear noise schedule, prompt ESM3 with $\{y_{\setminus m}, x\}$, and compute the cross-entropy of ESM3 logits with the masked positions of y. During training, the same mask is used to compute the likelihoods for the reference policy vs current policy, as well as for the preferred sample vs non preferred sample.

A.4.2. Preference Tuning Intuition

Rearranging the DPO term of the loss function gives some insight into how it finetunes the model for the preference pairs.

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = \\ \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[-\log \sigma\left(-\beta z_\theta(x, y_l, y_w)\right)\right] \quad (4)$$

where

$$z_\theta(x, y_l, y_w) = \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)} - \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)}$$
$$= \log \frac{\pi_{\text{ref}}(y_w|x)}{\pi_{\text{ref}}(y_l|x)} - \log \frac{\pi_\theta(y_w|x)}{\pi_\theta(y_l|x)}$$

The function $f(z) = -\log\sigma(-\beta z) = \log(1+\exp(\beta z))$ is the softplus function, and is an approximation of the hinge function; in other words $f(z) = \beta z$ when $z >> 0$ and $f(z) = 0$ when $z \ll 0$. Because of this property, there are two cases. In the case where

$$\log \frac{\pi_{\text{ref}}(y_w|x)}{\pi_{\text{ref}}(y_l|x)} >> \log \frac{\pi_\theta(y_w|x)}{\pi_\theta(y_l|x)} \quad (5)$$

$f(z)$ is in the linear regime, so the loss function is simply maximizing the likelihood ratio $\log \frac{\pi_\theta(y_w|x)}{\pi_\theta(y_l|x)}$. In the case where

$$\log \frac{\pi_{\text{ref}}(y_w|x)}{\pi_{\text{ref}}(y_l|x)} \ll \log \frac{\pi_\theta(y_w|x)}{\pi_\theta(y_l|x)} \quad (6)$$

the loss has saturated. This ensures that we do not deviate too far from the reference model.

These dynamics also hold true in the case of ESM3 finetuning. Although we use a surrogate instead of the true likelihood, the loss will increase the surrogate of the preferred pair over the non-preferred pair until the current model deviates too much from the reference model.

A.4.3. Evaluation Metrics

Possibly the most important part of preference tuning is how the generations are separated into preference pairs. The desired objectives for a generation are quality and correctness. Quality refers to the viability of the sequence to be a stable protein. Correctness refers to the extent to which it follows the given prompt; also called prompt consistency. This section only deals with structure coordinate prompts, so prompt consistency can be measured via constrained site RMSD (cRMSD), which is the RMSD between the backbone coordinates of the prompt and the corresponding coordinates in the predicted structure of the generated sequence. Sequence quality can be measured via predicted-TM (pTM) of a structure predictor on the generated sequence.

As with any metric, especially one which is really a surrogate such as a structure predictor, there is a risk of over optimizing: the model keeps improving the specific metric, e.g. in our case pTM but the actual property of interest, the viability of the sequence to be a stable protein, stops correlating with the metric (102). Using orthogonal models to rank our training dataset vs to perform evaluation helps mitigate this.

To create the training datasets, generations are evaluated according to backbone cRMSD and pTM of ESM3 7B to maintain a consistent structure predictor across all datasets. After the preference tuning phase, the generations from the tuned models are evaluated with ESMFold backbone cRMSD and pTM as an orthogonal model. Training on ESM3-scored preference pairs while evaluating using ESMFold provides a more objective evaluation.

A.4.4. Training Dataset

All ESM3 model scales are trained with the IRPO loss (Eq. 2) on their respective preconstructed training datasets consisting of structure coordinate prompts and generations. The datasets have 16 generations each for 30,000 prompts from the respective ESM3 model. Preference selection is determined via a threshold of metrics. A sample is considered "good" if it has ESM3 7B pTM > 0.8 and backbone cRMSD to its structure prompt < 1.5Å.

Each "good" sample is paired with a "bad" sample to create a preference pair. We found that enforcing a gap between metrics of paired generations improves results, so to qualify as a "bad" sample the generation must have a delta $\text{pTM} = \text{pTM}_{\text{good}} - \text{pTM}_{\text{bad}} >= 0.2$ and delta backbone $\text{cRMSD} = \text{cRMSD}_{\text{good}} - \text{cRMSD}_{\text{bad}} < -2$Å. Each prompt can have multiple preference pairs, and prompts with no valid preference pair are discarded.

The structure prompts are composed of a variety of proteins adapted from our pretraining pipeline. 50% of the prompts are synthetic active sites, while the other 50% are structure coordinates randomly masked with a noise schedule. All of the structure prompts are derived from PDB structures with a temporal cutoff of before May 1st, 2020.

The synthetic active sites are derived by finding sequences from PDB with coordinating residues. For these structures, the amino acid identities are included in the prompt.

The remaining structure track prompts are masked according to a cosine noise schedule. 50% of the noise scheduled prompts are masked in completely random positions, and the other 50% are masked according to an autocorrelation mechanism that prefers sequentially masked positions.

Each model's training dataset consists of generations of its own reference model. For each prompt, we generate samples from the corresponding ESM3 model scale using iterative decoding with $L/4$ steps, where L is the length of the prompt. We anneal the temperature from 1.0 to 0.5 over the decoding steps.

A.4.5. Evaluation Dataset: Tertiary Coordination

Tertiary coordination tasks require the generation of proteins which satisfy challenging tertiary interaction constraints. The model is prompted with the sequence and coordinates of a set of residues which are near in 3D space, but distant in sequence. To evaluate performance on these tasks, we curate a dataset of 46 proteins with ligand binding sites from the Biolip dataset (99). All selected proteins were deposited in the PDB after the training set cutoff date (2020-12-01). The coordinating residues shown to the model are given by the ligand binding sites defined in the Biolip dataset (Table S15).

ESM3 is prompted with the sequence and coordinates of the residues for a particular ligand binding site. We ask ESM3 to generate novel structures by applying multiple transformations to the prompt. The total sequence length is sampled evenly to be 150, 250, or 350 residues (regardless of the original sequence length). Next, we define a contiguous span of coordinating residues to be prompt residues with fewer than 5 sequence positions between them. The order and the distance between contiguous spans of residues is shuffled. Together, this ensures that,

for example, the original protein will no longer satisfy the prompt. We consider a generation a success if backbone cRMSD < 1.5Å and pTM > 0.8.

We construct a total of 1024 prompts for each ligand and generate a completion for each prompt with the model we are evaluating. We report Pass@128, which is an estimate for the fraction of ligands with at least one successful completion after 128 prompts per ligand. We estimate this using an unbiased estimator (Chen et al. (103), Page 3) using the success rate over 1024 prompts. We visualize randomly selected successful generations for both the base model and finetuned model in Fig. S19.

We evaluate RFDiffusion on the tertiary coordination benchmark following a similar methodology. We use the RFDiffusion model finetuned for active site scaffolding. For each ligand, we prompt the model with the same 1024 prompts as ESM3 (maintaining identical protein lengths and active site positions) and generate one structure per prompt using the default parameters provided in the RFDiffusion repository. To generate sequences, we inverse fold each structure once with ProteinMPNN using temperature 0.1, yielding 1024 sequences per ligand. This approach ensures that RFDiffusion and ESM3 are afforded the same sampling budget. These sequences are then scored using the same approach as for ESM3-generated sequences. Pass@128 and the fraction of tasks solved with at least 8, 16, and 32 distinct solutions (clustered at TM > 0.8) are reported in Table S16.

A.4.6. Supervised Finetuning

To judge the value of preference tuning, we also train a supervised finetuning (SFT) baseline where we finetune the model to increase likelihood of the high quality samples without the preference tuning loss. The 1.4B, 7B, and 98B models solve 14.2%, 33.7%, and 44.6% of tertiary coordination tasks at 128 generations, respectively, which improves upon the base models but is much lower than their corresponding preference tuned versions.

A.4.7. Training Hyperparameters

Each IRPO model is trained for 1000 steps using RMSProp. The learning rates are 1e-5, 1e-5, and 5e-6 for the 1.4B, 7B, and 98B, respectively, annealed using a cosine schedule after a 150 step warmup. Gradient norms are clipped to 1.0.

For all IRPO runs $\beta = 0.05$ and $\alpha = 0.8$. The SFT baseline uses the same hyperparameters, but with $\alpha = 0.0$ to disregard the preference tuning term.

A.5. GFP

ESM3 generates a dim distant GFP, B8, and a bright distant protein, esmGFP. Details are provided below on computational methods, experimental protocols, results, and post-experiment analyses.

A.5.1. Generation and Selection

The base ESM3 7B model generates candidate GFP designs for laboratory testing using a single prompt and a chain of thought over sequence and structure tokens. Candidates are filtered and

ranked by metrics at several steps in the process. Experiment 1 tests candidates across a range of sequence identity to a template, yielding multiple GFPs including dim hit B8. Experiment 2 consists of designs starting a chain of thought from the sequence of B8, yielding numerous bright GFPs including C10 which we term esmGFP. This section details the computational protocol that generated and selected candidate GFP designs for Experiments 1 and 2, shown in Fig. 4B. Protocols, metrics, and selection conventions are separately introduced and then synthesized in descriptions of the two experiments, at the end of the section.

A.5.1.1. Model

All candidate GFP designs were created using the base ESM3 7B model with no finetuning. Throughout generation, the model is prevented from decoding cysteine residues.

A.5.1.2. Prompt

All candidate GFP designs in Experiment 1 are produced with a chain of thought beginning from a single prompt. The goal of the prompt is to capture essential residue identities and structural features needed for chromophore formation and fluorescence, leaving other degrees of freedom open for the model to generate diverse designs.

Template We prompt ESM3 with a minimal set of sequence and structure information from 16 residues near the chromophore formation site. We use a pre-cyclized intermediate crystal structure derived from Barondeau et al. (53), PDB ID 1QY3, as a template. Since this template contains the chromophore maturation slowing mutation R96A, for the prompt we use the unmutated Arg96, constructing coordinates for the unmutated residue by superimposing the backbone atoms of 1GFL (104) onto 1QY3 and replacing 1QY3's A96 with 1GFL's R96. We subsequently refer to the full sequence and structure of 1QY3 with mutation A96R as 1QY3 A96R or the template.

Sequence prompt The sequence portion of our prompt consists of 7 template residues: Met1, Thr62, Thr65, Tyr66, Gly67, Arg96, and Glu222. Residues 65-67 form the chromophore. Met1 ensures proper start codon placement. Residues 62, 96, and 222 are described in (53) and other works to have key catalytic roles in chromophore formation.

Structure prompt The structure portion of our prompt consists of structure tokens and backbone atomic coordinates taken from 16 template residues at positions 96, 222, and 58-71 (inclusive) which roughly captures the central alpha helix. The unique geometry of the central alpha helix is known to be crucial for chromophore formation (53).

All other positions and tracks in the prompt are masked. The overall prompt length is 229, matching that of the template. Residue indices are contiguous and begin from 1.

A.5.1.3. Joint Sequence Structure Optimization

We employ the following procedure to jointly optimize the sequence and structure of designs throughout our experiments: While annealing temperature linearly from 1 to 0, we perform multiple iterations of first predicting the structure of a designed sequence and subsequently Gibbs sampling each position in the sequence for that predicted structure. In algorithmic form:

Algorithm 15 `gibbs_seq_given_struct`

Input: ESM3 f, sequence $x \in: \{0..20\}^L$, structure y, temperature t

for $i = \text{shuffle}(\{1, ..., L\})$ **do**
  $x_i \sim \exp\left(\log f(x_i \mid x_{\setminus i}, y)/t\right)$
end for
return x

Algorithm 16 `joint_optimize`

Input: ESM3 f, initial sequence x_1, iterations I, initial temperature t_1, final temperature t_f

for $i = 1, \ldots, I$ **do**
  $t_i = (t_f - t_1) \cdot (i/(I-1)) + t_1$
  $y_i =$ `generate`$_{struct}(f, x_i, len(x_i), T = 0)$
  $x_{i+1} =$ `gibbs_seq_given_struct` (f, x_i, y_i, t_i)
end for
return x_{I+1}

Three variants of `gibbs_seq_given_struct` in `joint_optimize` were employed for Experiments 1 and 2. Joint optimization occasionally produces repetitive spans of amino acids when temperature is annealed to low values. Variant 1 and 2 are intended to address this, in differing ways. Variant 3 is an experiment in biasing the logits with a PSSM of known natural GFPs. Half of the candidates in Experiment 2 were produced using Variant 3. This half did not include esmGFP, which was generated by Variant 2 of joint optimization.

1. **Variant 1: Negative Local Sequence Guidance** We bias the logits of the model away from those produced just from a highly local span of the sequence. Specifically, we use classifier-free guidance (105):

$$\text{logits}' = \text{weight} * \left(\text{logits}_{\text{cond}} - \text{logits}_{\text{uncond}}\right) + \text{logits}_{\text{uncond}}$$

   but push away from the logits produced by inputting just 7 residues centered on the position being sampled, with weight 2 and nothing else. All other sequence positions and all other model inputs are left blank.

$$\text{logits}' = 2 * \left(\text{logits}_{\text{cond}} - \text{logits}_{\text{local\_seq}}\right) + \text{logits}_{\text{local\_seq}}$$

2. **Variant 2: Max Decoding Entropy Threshold** We optionally skip resampling of sequence during Gibbs sampling at positions whose entropy over sequence tokens exceeds a user specified threshold.

3. **Variant 3: PSSM Bias** In Experiment 2 only, we experiment with both including and excluding a PSSM-based bias during Gibbs sequence sampling. Specifically, we add a PSSM constructed from 71 natural GFPs (see Appendix A.5.1.4 for details) directly to the sequence output logits of the model, with a user-specific weight. esmGFP did not use this option; it was produced with weight 0.

A.5.1.4. METRICS

GFP designs are produced and scored by a number of ESM3derived and independent metrics. Unless otherwise noted, designed structures are predicted using ESM3 with only sequence as input, using iterative decoding of structure tokens with temperature 0 and subsequent decoding of backbone coordinates with an older version of the structure token decoder.

The following is an exhaustive list of metrics used. An exact break down of where and how specific metrics are used can be found in Appendix A.5.1.5, Appendix A.5.1.6 and Appendix A.5.1.7.

Template Chromophore Site RMSD is calculated via an optimal alignment (106) of N, C, CA, and inferred CB atoms at positions 62, 65, 66, 67, 96, and 222 in the predicted structure of a design and the template (crystal) structure.

Template Helix RMSD is calculated in the same way, but for N, C, CA atoms only, at design and template positions 58-71 (inclusive).

1EMA Helix RMSD is a metric proposed in (107). An RMSD is calculated between alpha helix residues in the predicted designed structure and a specific crystal structure of avGFP, PDB ID 1EMA. Our calculation differs slightly from (107). We calculate RMSD for N, C, CA and inferred O atoms, and consider only positions 60-64 and 68-74 (both ranges inclusive) to exclude chromophore positions 65-67.

Sequence Pseudo-perplexity is calculated as defined in (108). Given a protein sequence, positions are masked one at a time, negative log-likelihoods of input tokens at masked positions are averaged across all positions in the sequence, and the result is exponentiated.

Round-trip Perplexity is calculated for a designed sequence via predicting its structure with ESM3, and then evaluating the perplexity of the sequence given that predicted structure under a single forward pass of ESM3.

N-gram Score is calculated as the $\boldsymbol{E}_{\text{ngram}}$ term defined in (11). This score assesses the divergence between the N-gram frequencies of residues in the designed sequence and those found in a background distribution, derived from UniRef50 2018_03. Specifically, for a function $\text{ngram}_{\boldsymbol{i}}$ that takes in a sequence $\boldsymbol{x}$ and an N-gram order $\boldsymbol{i}$, and a precomputed distribuion of background N-gram frequencies $\text{ngram}_{\boldsymbol{i,bg}}$, the score is calculated as:

$$E_{\text{ngram}} = \sum_{i \in \{1,2,3\}} D_{KL}(\text{ngram}_i(x), \text{ngram}_{i,\text{bg}}) \quad (7)$$

PSSM A position-specific scoring matrix (PSSM) is constructed from a MSA of 71 natural GFPs (109). Specifically, at positions aligned to our template, frequencies for the 20 canonical amino acids (excluding gaps) are transformed to log odds via dividing by the uniform background ($\boldsymbol{p(aa) = 0.05}$), adding an epsilon of 1e-9, and applying log base 2. This produces a matrix of scores of size 229 x 20.

PSSM score We extract from the PSSM values at (position, amino acid) pairs occurring in an input sequence. These are averaged to produce a score.

N-terminus Coil Count is metric intended to measure structural disorder at the N-terminus of a design. We observed that predicted structures have various levels of disorder in this region. To quantify it for possible filtering, we apply mkdssp (82) to the ESM3-predicted structure of a design, and record how many of the first 12 positions are reported as having SS8 labels in {S,T,C}.

A.5.1.5. Selection Criteria

Among Experiment 1 and 2, designs are selected for testing by first applying a set of filters, and then selecting the top-N designs according to a score-based ranking. Scores are calculated by summing the values of several metrics, which are each normalized across designs to have zero mean and unit variance and which are negated when appropriate so that lower values are always better.

Common Filters The following filters are applied in both Experiments 1 and 2.

- Template Chromophore Site RMSD <1.5Å
- Template Helix RMSD <1.5Å
- N-gram Score <5

Common Score Terms The following score terms are used in both Experiments 1 and 2.

- Sequence Pseudo-perplexity
- Round-trip Perplexity
- ESM3 pTM

A.5.1.6. Generation and Selection of Designs for Experiment 1

In this experiment, we generate a set of GFP designs for experimental testing with a range of sequence identities to our template. Designs are generated by a chain of thought: from the prompt, ESM3 decodes all masked structure tokens, then all masked sequence tokens. Lastly, sequence and structure tokens are jointly optimized.

Initial Generation Starting from the prompt, we first generate 38k structures by decoding masked structure tokens one at a time using a fixed temperature sampled uniformly from the range (0, 1.25) for each generation. To focus compute on the most promising structures, we filter according to Template Chromophore Site RMSD <1Å, yielding 24k selected structures. We next generate ≈ **4** sequences for each structure with a temperature uniformly sampled from the range (0, 0.6), yielding 92k total sequences.

Selection We select a subset of promising initial generations for further optimization by applying Common Filters with N-gram score's threshold modified to <5.5, ranking designs according to {Common Score Terms, mean ESM3 pLDDT, mean ESMFold pLDDT, and ESMFold pTM}, and selecting the best 40 designs in each interval of 0.1 sequence identity to the template sequence in [0.2, 1.0], 320 in total.

Joint Sequence Structure Optimization We then jointly optimize the sequence and structure of designs. Using 30 iterations in each case, we run 5 seeds of optimization with max decoding entropy threshold = 1.5 and 2 seeds of optimization with negative local sequence guidance = 2.0, yielding 67k total designs. Designs from every iteration are included in this pool.

Selection To select a set of designs for laboratory testing, we apply {Common Filters, N-terminus Coil Count <6}, rank designs according to {Common Score Terms, ESMFold pTM, 15 * PSSM Score}, and select the best 88 designs across 8 buckets of sequence identity to our template among intervals of width 0.1 in range [0.2, 1].

A.5.1.7. Generation and Selection of Designs for Experiment 2

In this experiment, we perform further refinement of the dim, distant GFP found in Experiment 1, B10. To produce a diversity of designs, we sweep over a number of settings: two variations of refinement are performed, and 2 selection protocols are used.

Local Joint Optimization Starting from our dim GFP design, B10, we perform joint_optimize using a full grid sweep of the following sets of settings: Initial temperatures {0.001, 0.01, 0.05, 0.1, 0.5}, PSSM bias weights {0, 0.01, 0.05, 0.1, 0.5}, Max decoding entropy thresholds {0.8, 1, 1.25, 1.5, 2.0}. For each unique settings combination, we use 20 iterations of optimization with 3 seeds, continuing the final step of Gibbs sampling until convergence. After accounting for some distributed system machine failures, this yields 6.3k total candidate designs.

Selection We select two sets of 45 designs for laboratory testing via two filters and a shared set of ranking criteria.

1. **Set 1**: We filter according to {PSSM Bias ≠ 0, Common Filters, RMSD to starting structure <1Å, identity to starting sequence in (0.7, 1.0)}.

2. **Set 2**: We filter according to {PSSM Bias = 0 (no bias), Common Filters, RMSD to starting structure <1Å, Identity to starting sequence in (0.9, 1.0)}. esmGFP comes from this pool.

For each set, we rank according to {Common Score Terms, 8 * PSSM Score, 15 * 1EMA Helix RMSD} and select 45 designs each for testing.

A.5.2. Experimental Methods and Data Analysis

A.5.2.1. Strains and Plasmids

We designed a custom bacterial expression vector containing an Ampicillin-resistance gene, the BBa R0040 TetR promoter, the BBa B0015 terminator, and a Bsa-I golden gate site between the promoter and terminator. GFP designs were codon optimized for *E. coli* expression, compatible golden gate overhangs were added, and sequences were ordered from IDT (Integrated Device Technology Inc.). They were then cloned by golden gate assembly into the vector. We evaluated our GFP designs in the *E. coli* host Mach1 (Invitrogen).

A.5.2.2. Fluorescence Assays of GFP Designs

To evaluate the fluorescence of our GFP designs, we transformed our designs into Mach1 cells. For each of two replicates of a design, a colony was seeded into a 1 mL TB culture containing 50 µg/mL carbenicillin. Cultures were grown in 96 deep well blocks at 37 °C in an Infors HT Multitron Shaker with a shaking speed of 1000 RPM for 24 hours. After 24 hours, 1 µL of the cultures were diluted in 200 µl of 0.2 µm filtered DPBS.

Fluorescence intensity of the samples was then quantified at the single cell level using a NovoCyte Quanteon Flow Cytometer (Fig. S20).

The remaining cultures were spun down at 4000 g for 10 minutes, resuspended and lysed with 300 µL lysis buffer (1x bugbuster, 500 mM NaCl, 20 mM Tris-HCl pH 8, 10% glycerol, cOmplete™ , EDTA-free Protease Inhibitor Cocktail), incubated at room temperature on a Belly Dancer Orbital Shaker for 10 minutes, and lysate clarified by centrifugation at 4000 g for 20 minutes. 100-120 µl lysate was transferred to a 96 well black clear-bottom plate, and GFP fluorescence was measured using a Tecan Spark Reader. Fluorescence emission was captured at 515 nm with a 10 nm bandwidth and excited with 485 nm with a 10 nm bandwidth. Absorbance was captured at 280 nm with a 3.5 nm bandwidth to assess total protein content per well. For longer time points, plates containing lysate were sealed and incubated at 37°C for up to 7 days prior to measuring fluorescence. GFP fluorescence values were first ratio normalized within a well by their absorbance at 280 nm, and then further ratio normalized across wells using the measured values from a negative control *E. coli* containing vector without GFP. Data from two replicates was then averaged for (Fig. 4B bottom) and (Fig. 4C).

Overview photos of the plates (Fig. 4B top) were taken with an iPhone 12 mini under blue light illumination from an Invitrogen Safe Imager 2.0 Blue Light Transilluminator.

For excitation spectra, emission was captured at 570 nm with a 50 nm bandwidth, while the excitation wavelength was varied from 350 to 520 nm with a 10 nm bandwidth. For emission spectra, an excitation wavelength of 430 nm was used with a 50 nm bandwidth, while emission was captured at varying wavelengths from 480 to 650 nm with a 10 nm bandwidth. Excitation and emission spectra were normalized by their maximum values (Fig. 4C).

A.5.2.3. Additional Experiments

Plate overview photographs (Fig. 4B top) were taken over two weeks since the initial lysate was created and over one week after the final plate reader quantification was done, and so possibly show additional brightness from slow chromophore maturing designs. We observed some low level contamination of wells H11 (vector with no GFP or designs) and H12 (lysis buffer only) in the photograph of Experiment 1 (Fig. 4B top left). Some of this contamination is already visible in well H12 during the initial plate reader quantification (Fig. 4B bottom left). To address potential contamination concerns we performed an additional replication of B8 and observed a similar level of brightness to Experiment 1 (50x less bright than natural GFPs) (Fig. S21).

Chromophore knockout versions of 1QY3 A96R and esmGFP were created through additional T65G and Y66G mutations. These variants, along with 1QY3 and esmGFP, were synthesized and measured as part of an independent replicate performed by Genscript following the *E. Coli* based fluorescent plate reader assay described above. Normalization was performed with an OD600 measurement of the cells prior to lysis. Analysis otherwise proceeded as above. Two replicates were performed for each design and results were averaged. Chromophore knockout reduced fluorescence to background levels (Fig. S22).

A.5.3. Sequence searches and comparisons

A.5.3.1. Database Searches

BLAST nr search: esmGFP's sequence was searched with BLAST's online server using the non-redundant sequences database nr with all default settings. tagRFP's sequence was taken from the top hit. The exact top hit found was TagRFP [Cloning vector pLX-B2-TagRFP-T, Sequence ID ASG92118.1 and is shown in its entirety in Table S17.

Train set search: MMseqs2 (55), version 15.6f452, was used to search all datasets that ESM3 was trained on at the maximum available expansion level; for cluster resampling datasets all cluster members are searched, not just cluster centers. The goal is to search against every possible sequence that ESM3 may have seen during pretraining. Settings are selected for conducting a high sensitivity search: -s 6 -a --max-seqs 10000.

A.5.3.2. Sequence Identity Calculations

To calculate sequence identities involving the two highlighted GFP designs (B8, esmGFP) and select reference proteins, the following procedure is used. MAFFT (110) v7.525 is applied with all default settings to the sequences of B8, esmGFP, the top tagRFP sequence found by BLAST, eqFP578 (from FPBase (111)), the template (PDB ID 1QY3, with mutation A96R), and avGFP (from FPBase). Identities between two sequences are calculated as the number of matching non-gap residues at aligned positions divided by the minimum non-gapped length of the query and target protein. This is the same sequence identity formula used in Appendix A.5.4. Aligned sequences and identities and mutation counts to esmGFP are provided in Table S17.

A.5.3.3. Inner-Barrel Mutation Count

Positions in esmGFP are described as internal if they have SASA < 5 in their predicted structure. SASA is calculated as in Appendix A.2.1.6) from the all-atom structure of esmGFP, predicted with ESM3 7B.

A.5.4. Phylogenetic Analysis

Sequences and metadata of natural and designed fluorescent proteins were obtained from FPBase (111). An initial set of 1000 proteins was filtered according to the following criteria: presence of a specified parent organism, an amino acid sequence between 200 and 300 residues long, a specified emission maximum, and no cofactors. NCBI taxonomy database was used to obtain taxonomic information about each species. These sequences were further filtered according to keep those that had species found by NCBI and were Eukaryotic but not from Chlorophyta (to exclude Channel-rhodopsin like proteins). The 648 sequences that passed these criteria, along with the sequence for esmGFP, were aligned to a multiple sequence alignment using MAFFT and sequence identity was computed between each pair of sequences as described above. All pairs within and across taxa were considered for (Fig. 4F). All designed sequences were considered to belong to the species annotated as their parent organism.

All 648 used sequences belonged to the Leptocardii (e.g. laGFP), Hexanauplia (e.g. ppluGFP), Hydrozoa (e.g. avGFP), or Anthrozoa (e.g. efasGFP) classes. The sequence identity of esmGFP was computed to each protein in these classes (Fig. S23). esmGFP was found to be closest to Anthrozoan GFPs (average sequence identity 51.4%) but also shares some sequence identity to Hydrozoan GFPs (average sequence identity 33.4%).

To estimate the millions of years of evolutionary distance by time between esmGFP and known fluorescent proteins we built an estimator to go from sequence identity between pairs of GFPs to millions of years (MY) apart. We used the following six Anthozoan species: *Acropora millepora*, *Ricordea florida*, *Montastraea cavernosa*, *Porites porites*, *Discosoma sp.*, *Eusmilia fastigiata* along with the six GFPs amilGFP, rfloGFP, mcavGFP, pporGFP, dis3GFP, efasGFP respectively. These species and GFPs were chosen because they were annotated in both a recent time calibrated phylogenetic analysis of the Anthozoans (57) and a recent study of GFPs (48). Each of these species contains multiple GFP like sequences including red and cyan FPs. These particular GFPs were chosen as they were annotated to be the main GFP in each species. The millions of years between each species was estimated as twice the millions of years to the last common ancestor annotated in the time calibrated phylogenetic analysis. Using statsmodels (112), a line of best fit was fit between MY and sequence identity. The line was required to pass through a sequence identity of 1.0 and 0 MY. The MY to esmGFP was then estimated using this line and the sequence identity of esmGFP to the nearest known protein.

A.6. OPEN MODEL

We are releasing the ESM3 source code and model weights of an open model, ESM3-open. ESM3-open is a 1.4Bparameter model we trained without OAS antibody sequences and with precautionary risk mitigations for release to the academic research community.

As part of this release, we follow guidance from the Principles for the Responsible Development of AI for Biological Design (113). We adopted precautionary risk mitigations, described in

Appendix A.6.1, and performed risk evaluations, detailed in Appendix A.6.2. Additionally, we conducted a review of the risks and benefits of releasing ESM3-open with experts from the scientific community. We provided reviewers access to ESM3-open, along with a detailed technical report on our risk evaluations. We received unanimous feedback from our reviewers that the benefits of releasing the model greatly outweigh any potential risks.

We see this release as a first step and plan to work with the scientific community to continue to improve processes around responsible development. Open models enable the scientific community to better understand and reduce any potential risks of biological design tools. As our understanding develops alongside the capabilities of future models, we plan to continuously improve our evaluation frameworks, safeguards, and mitigation strategies.

A.6.1. ESM3-open Mitigations

As a precaution, we filtered the training data of ESM3-open to minimize model performance on sequences of potential concern while otherwise maintaining performance. We also removed the capability for the model to follow prompts related to viruses and toxins.

Filtering sequences of potential concern. Previous work has shown that the performance of protein language models is closely related to the number of similar sequences present in the training data (5). We therefore removed sequences aligned to potentially-concerning proteins from the training data in order to reduce the capability of ESM3-open on these sequences.

We identified and removed sequences unique to viruses, as well as viral and non-viral sequences from the Select Agents and Toxins List (114) maintained by the CDC and USDA. The U.S. Department of Health & Human Services recommends filtering based on the Select Agents list as part of their Screening Framework Guidance for Providers and Users of Synthetic Nucleic Acids (115).

To filter data, we create two denylists: the Viral Denylist and the Select Agent Denylist. We then remove all sequences from the training set that are detected to align to those in the denylists by MMseqs2 at or above a given sequence identity threshold.

To create the Viral Denylist, we identify ~4M sequences that are annotated as viral in UniProt and align almost exclusively to other viral sequences in UniProt. This gives us a procedure that removes viral proteins with both high sensitivity and specificity (as measured by UniProt taxonomic annotations). To create the Select Agents Denylist we identify all sequences in UniProt belonging to organisms on the Select Agents and Toxins List (114). This process gives us 147K non-viral sequences and 40K additional viral sequences.

For each denylist, MMseqs was used to query against the full set of training databases, (including PDB, UniRef, MGnify, and JGI) and all hits were removed from the training set. This filter removes a total of 10.6M sequences across all training sets.

Removal of keywords of concern. There are a number of keyword prompts associated with viruses and toxins that we aim to remove. We first identify a list of harmful keywords with the following steps:

1. We curate a list of filter terms associated with viruses and toxins. The full filter term list is available upon request.

2. We then identify all InterPro tags whose free-text term names contain at least one of the filter terms.

3. We identify keywords that are associated with flagged InterPro tags but that are not associated with nonflagged InterPro tags. We remove those keywords. Keywords which are associated with both flagged and non-flagged InterPro tags (e.g. “extracellular region”) are not removed.

4. We additionally remove all keywords that themselves directly contain one of the filter terms

Of the original 68,103 keywords that ESM3 is trained with, this filter removes a total of 9,462 (14%), creating a new vocabulary of 58,641 keywords.

The function vocabulary is defined via vectors representing Term Frequency Inverse Document Frequency (TF-IDF) which are then tokenized using Locality Sensitive Hashing (LSH), as previously described in Appendix A.1.8. To remove flagged keywords, they are first removed from the TF-IDF vocabulary by removing the entries corresponding to flagged keywords. This reduces the TF-IDF vector size to 58,641. The LSH tokenization is defined by 64 hyperplanes, each defined in the TF-IDF space, i.e. a Euclidean space with one dimension per keyword. We redefine the hyperplanes to be in the reduced space by removing the dimensions corresponding to the flagged keywords. This permanently removes the information required for tokenization of the flagged keywords. This mitigation is highly selective and does not change the tokenization for any non-flagged keywords.

A.6.2. ESM3-open Evaluations

In the section below, we outline our evaluations of ESM3open performance. When appropriate, we compare ESM3open to either existing open models, (e.g. ESM2 or ESMFold), or to a compute-matched version of ESM3-open, trained without any data mitigations.

Structure Prediction. In Fig. S24A, we show that ESM3open achieves competitive performance on structure prediction as measured by LDDT on CASP14, 15 and CAMEO, showing very slight degradation from our compute-matched 1.4B model without data filtering. The evaluation framework is described in Appendix A.3.4.

We also measure the ability of ESM3 to predict the structure of a subset of viral proteins. In Fig. S24A we evaluate structure prediction on a set of structures derived from viruses that were purged from the PDB training set. For the chains in PDB that were > 70% sequence identity hits to the Viral Denylist, we cluster at 40% sequence identity and then select the longest chain (with length ≤ 1024) from each cluster. ESM3-open has an average LDDT of 0.56 on the viral structures. Without the data mitigation, a compute-matched ESM3-open has an average LDDT of 0.60.

Representation Learning. ESM3-open achieves strong performance on representation learning, slightly outperforming ESM2 (3B) on contact prediction as measured by precision at L (P@L)

on structures derived from CASP14/15, and CAMEO, see Fig. S24B. The evaluation framework is described in Appendix A.3.3.

Function Keyword Prediction. ESM3-open is able to predict function keywords for proteins in a validation set derived from UniRef and annotated with InterProScan, see Fig. S24C. ESM3-open achieves a Mean Average Precision for all keywords of 0.81 (macro average), and a precision of 0.89 (micro average) for the top 1000 keywords, discarding common terms such as "the". The evaluation framework is the same as that described in Appendix A.1.8.2.2.

Zero-shot Viral Fitness Prediction. We measure the ability of ESM3 to identify viable sequences and understand the effects of mutations on viral proteins. The evaluation consists of the single mutant variants from 217 Deep Mutational Scanning (DMS) datasets collected in ProteinGym (116). This includes 28 DMS landscapes from viral proteins and 189 from other proteins. We evaluate the correlation (Spearman ρ) between the predicted variant effect and measured variant effect. The predicted variant effect is measured as the difference between the logit value for the variant allele and the logit value of the wildtype allele at a given masked position (18).

First, we compare the performance of ESM3-open to a compute-matched version of ESM3-open which did not undergo any data filtering. We also compare the performance of ESM3-open to existing open model baselines. Fig. S24D assesses performance relative to the EVMutation (117) baseline. EVMutation is a Markov Random Field model (not deep learning-based) trained on a multiple sequence alignment of the target protein. BLOSUM62 is a baseline based on amino acid substitution frequencies. Before mitigations, ESM3-small outperforms ESM2 on viral landscapes, but falls short of EVMutation. EVMutation uses lower sequence diversity thresholds when generating MSAs for viral proteins, while ESM2 and ESM3 use fixed 50% and 70% clustering for their training sets respectively, which may explain the difference in capability. On non-viral landscapes, ESM3 performs on par with ESM2. Applying data filtering as a mitigation reduces average Spearman ρ performance on viral fitness prediction from 0.28 (ESM3small) to 0.17 (ESM3-open), to the level of a BLOSUM prior, which incorporates no knowledge of the statistics of the protein family. This indicates that the model contains essentially no information about the evolutionary statistics of the evaluated viral protein families. Performance on nonviral proteins is not adversely affected, changing from 0.46 (ESM3-small) to 0.45 (ESM3-open).

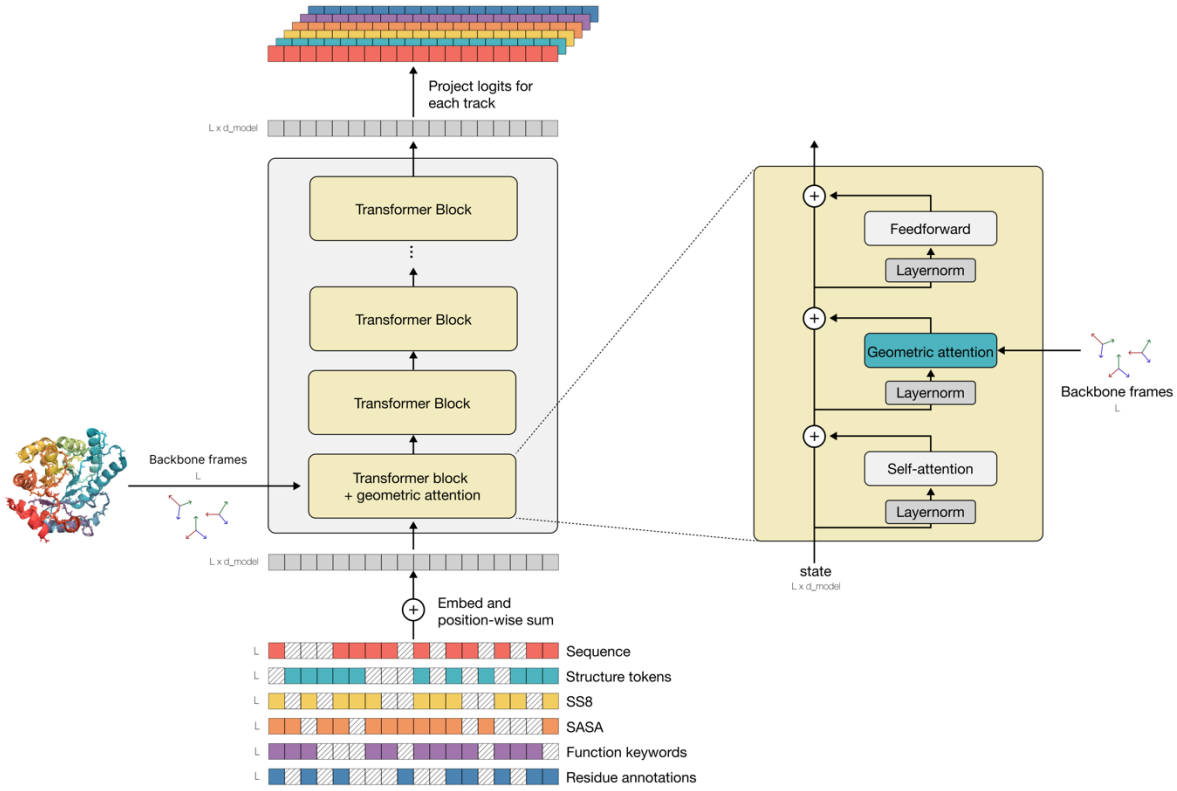

Fig. S1.

The ESM3 architecture. ESM3 is a masked language model that reasons over protein sequence, structure, and function, each of which are represented as token tracks at the input and output. Tokens are embedded and summed at the input to a transformer stack. The first block (expanded on the right) contains an additional geometric attention layer for processing atomic coordinate inputs. During training, random masks are sampled and applied to each track. Masked token positions are predicted at the output.

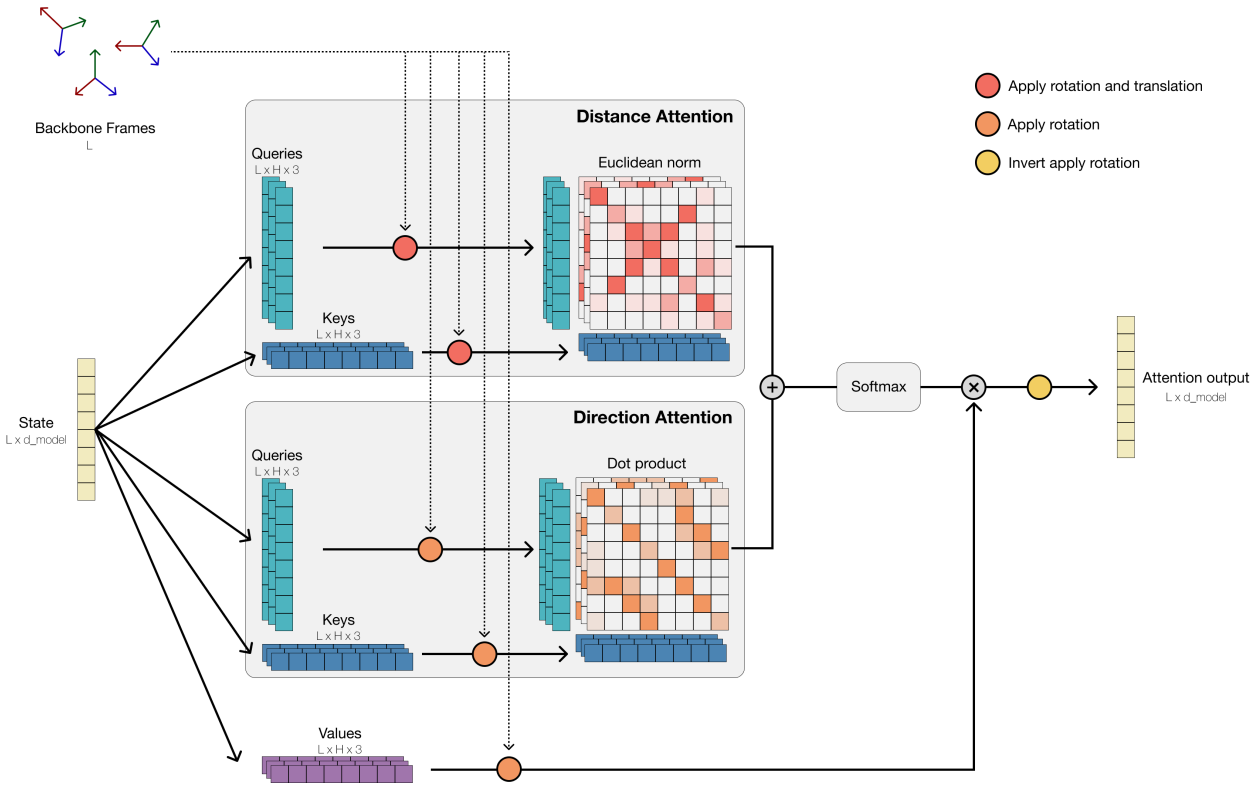

Fig. S2.

Geometric attention. Geometric attention is an SE(3) invariant all-to-all attention mechanism where the attention score matrix is a weighted sum of two terms: (1) the pairwise distances between queries and keys rotated and translated by their respective backbone frames, and (2) the pairwise dot products between queries and keys rotated by their respective backbone frames. This attention mechanism encodes structural information with throughput comparable to the standard attention operation in transformers.

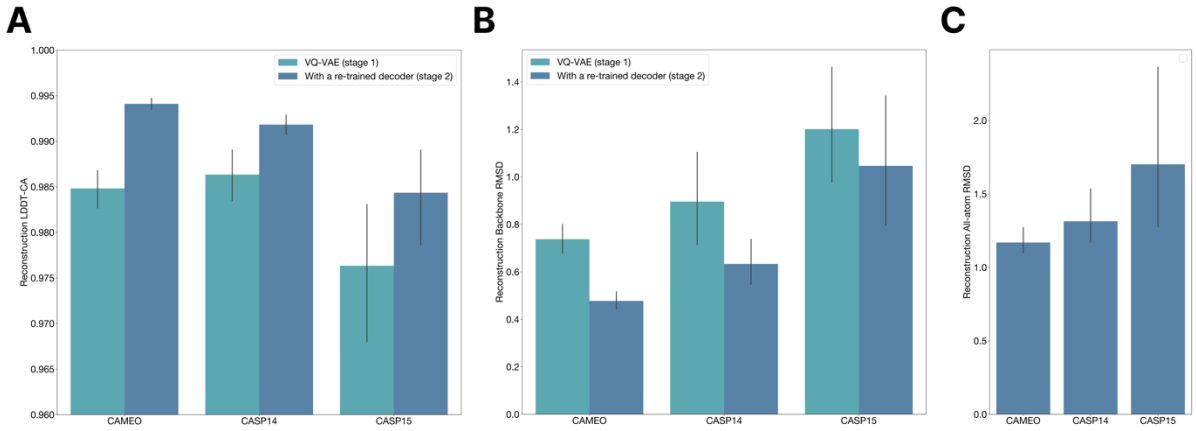

Fig. S3.

Structure tokenizer reconstruction quality. Reconstruction quality of the structure tokenizer after stage 1 and stage 2 of VQ-VAE decoder training evaluated on temporally held out CAMEO, CASP14, and CASP15. (A) Reconstruction LDDT-CA. (B) Reconstruction backbone RMSD. (C) All-atom reconstruction RMSD from the stage 2 decoder which additionally receives sequence input. Error bars represent 95% confidence intervals.

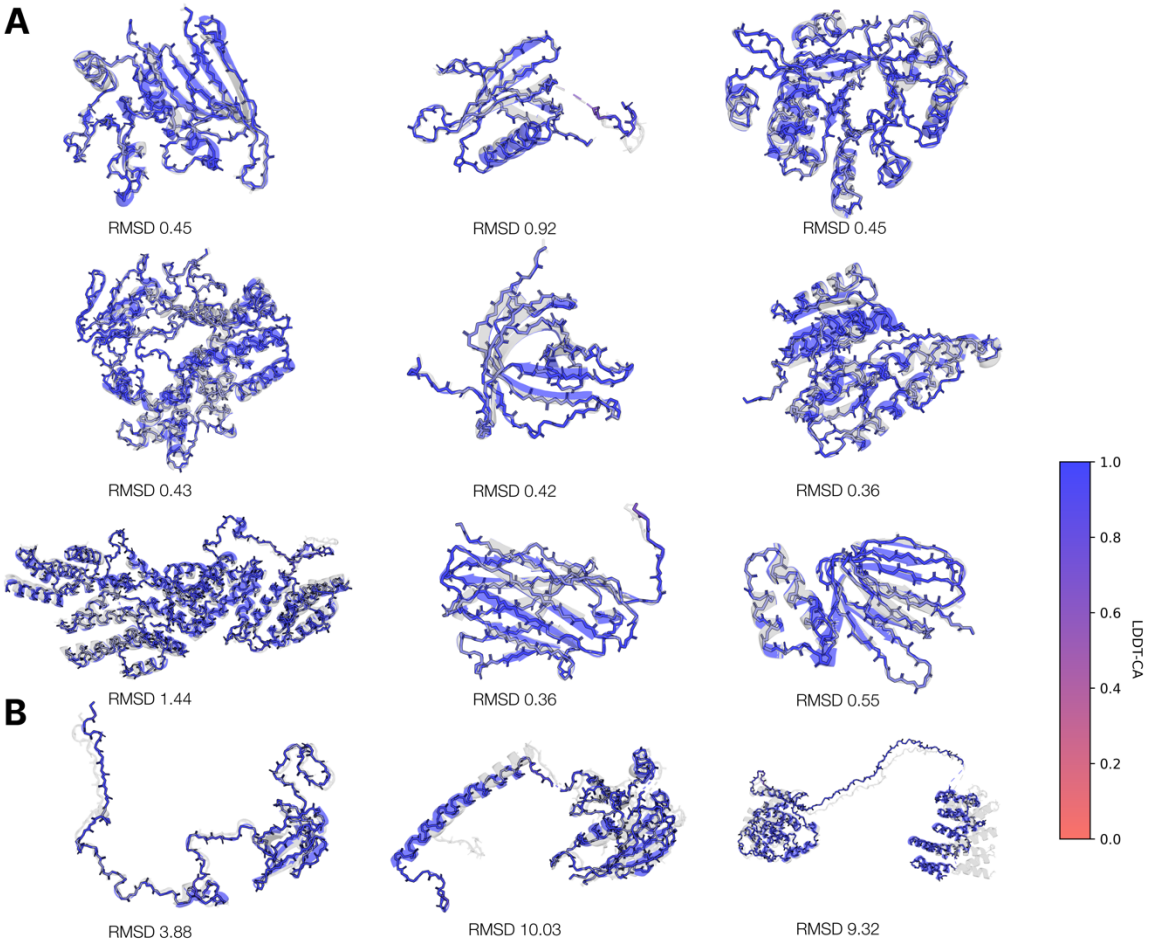

Fig. S4.

Visualization of structure tokenizer backbone reconstructions. (A) A random sample of reconstructions from the structure tokenizer on the test set. The vast majority of structures have near perfect backbone reconstruction. (B) A selection of the worst reconstructions in the test set. Long stretches of disordered regions, a lack of tertiary contacts, and unresolved coordinates can lead to inaccurate global orientation of structural elements, while local structure reconstruction remains largely error-free.

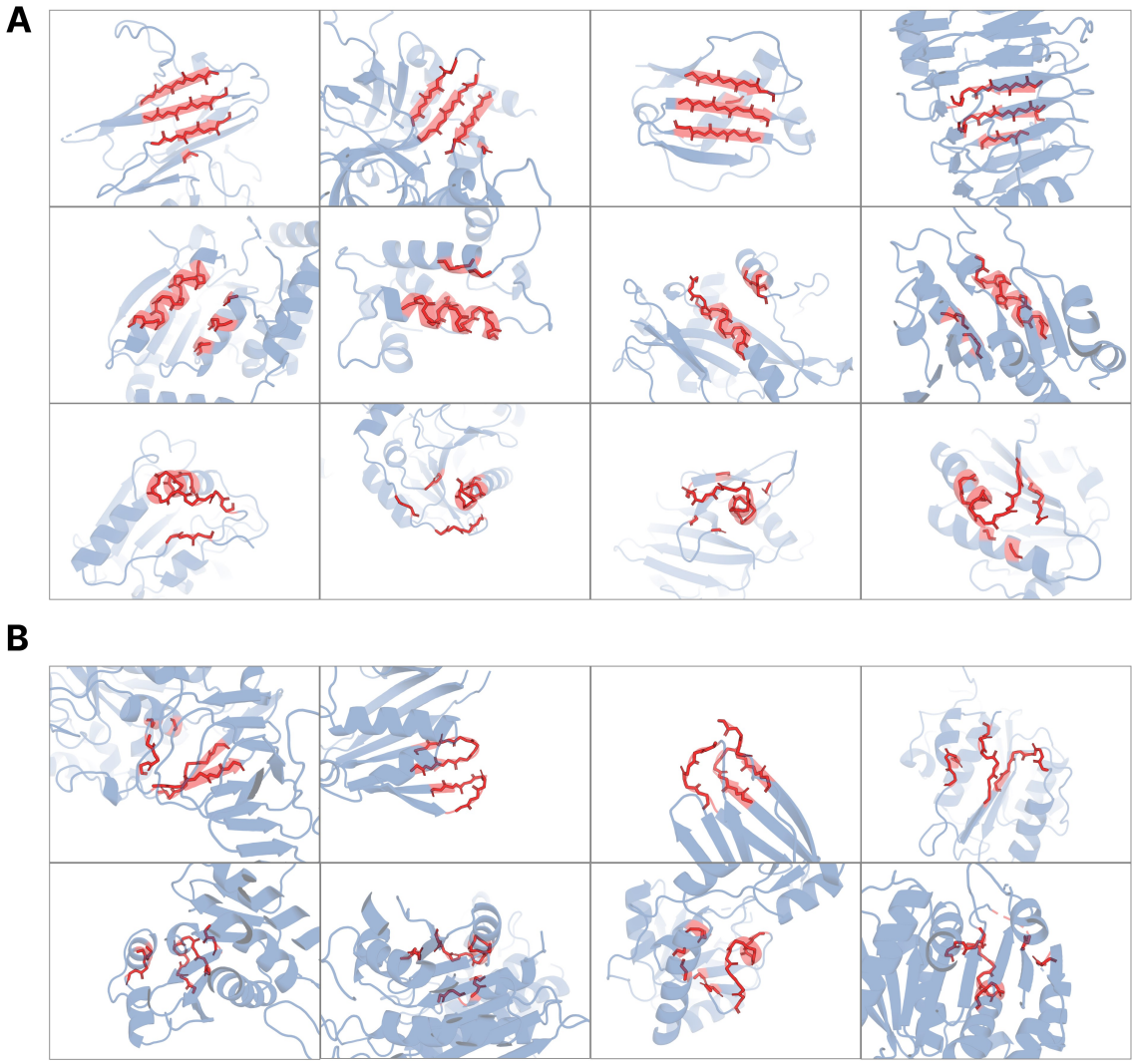

Fig. S5.

Visualization of local neighborhoods which map to the same learned structure token. The VQ-VAE encoder reasons over local structure neighborhoods (highlighted in red) which include the query residue and the 15 nearest neighbors in structure space. (A) Rows correspond to token indices 585, 59, and 3692 for top, middle, and bottom, respectively. Columns show different local structures mapping to the same token. (B) Some tokens represent multiple types of local neighborhoods. All local neighborhoods in B map to the same codebook index 3276. While the meaning of a single token may be ambiguous, the decoder is able to disambiguate given surrounding context in the full sequence of structure tokens.

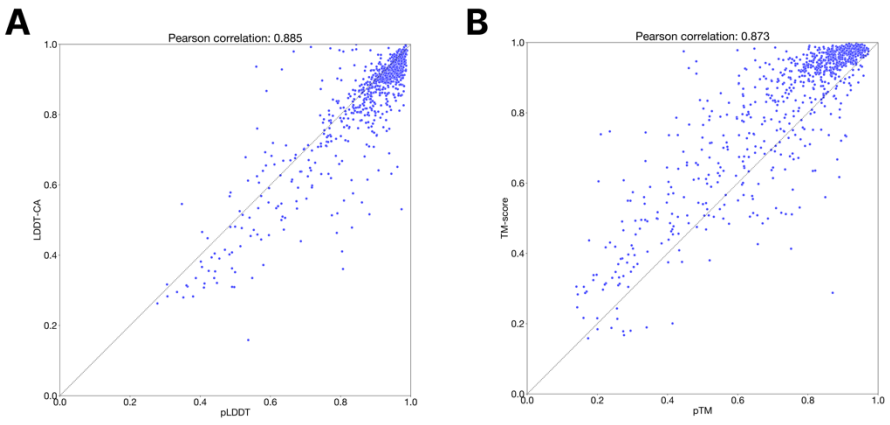

Fig. S6.

pTM and pLDDT calibration. Calibration of the structure token decoder pLDDT and pTM (using ESM3 7B as the structure token prediction model) on the CAMEO test set.

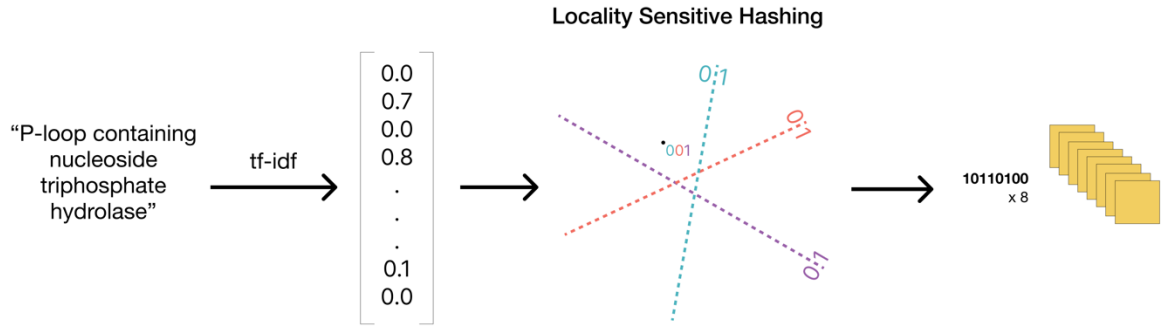

Fig. S7.

Schematic of function tokenization. The set of InterPro and GO descriptions of protein function are vectorized by a TF-IDF model and then hashed by a locality sensitive hash to produce 8 tokens each containing 8 bits.

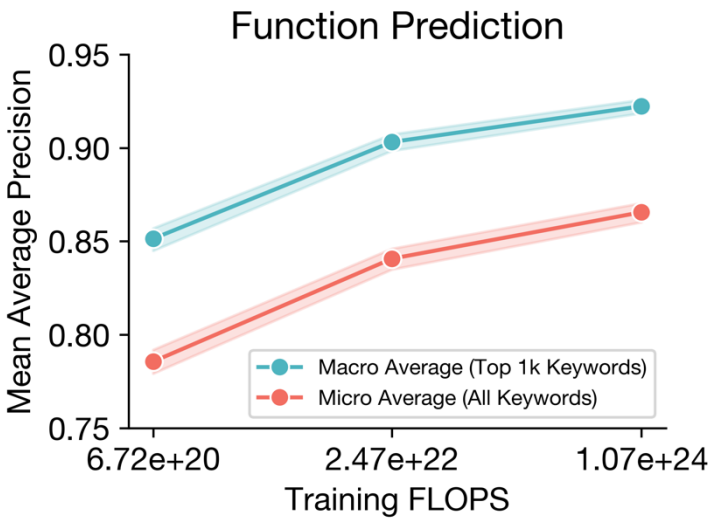

Fig. S8.

Function prediction benchmarking results. Mean Average Precision (mAP) for function keyword prediction. Predictions and labels are compared on a per-position basis to evaluate the model's ability to localize site-specific functional attributes by keywords such as "active site". We report mAP for the full keyword set (red) with a "micro" average because many keywords have few or no labels in the validation set. To report a "macro" average mAP we compute mAP for each of the top 1,000 most prevalent keywords in our evaluation set (discarding uninformative keywords such as "the") and report a uniform average (blue). 95% confidence intervals are shown by shading.

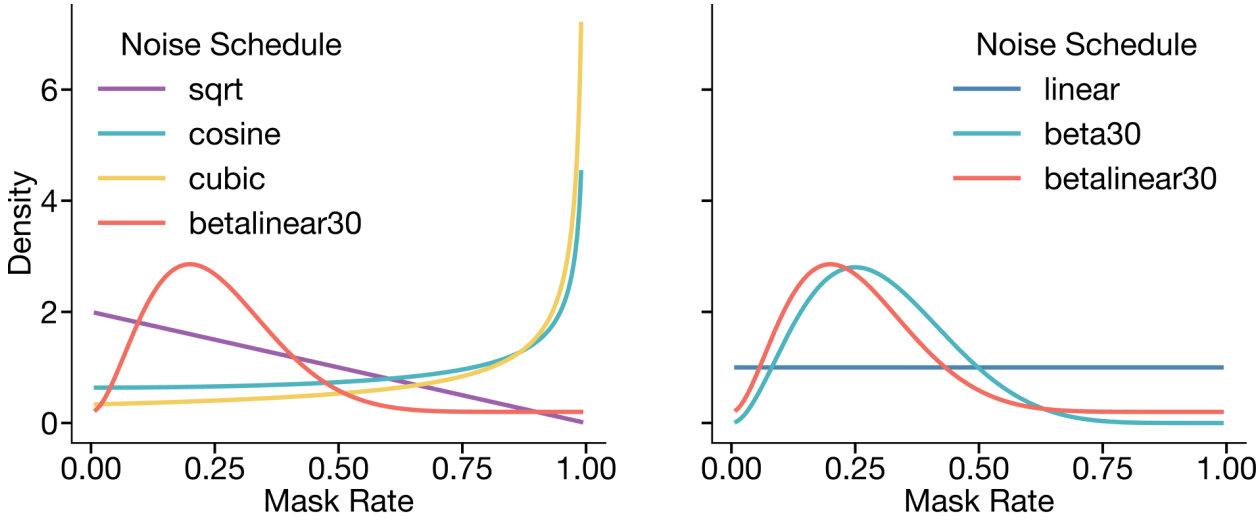

Fig. S9.

Visualization of noise schedules used. Left shows the probability density function of all noise schedules used. Right shows the betalinear30 distribution (which is drawn from β(3,9) with 80% probability and a linear distribution with 20% probability) against a beta30 distribution (defined by β(3,7)) and a linear distribution.

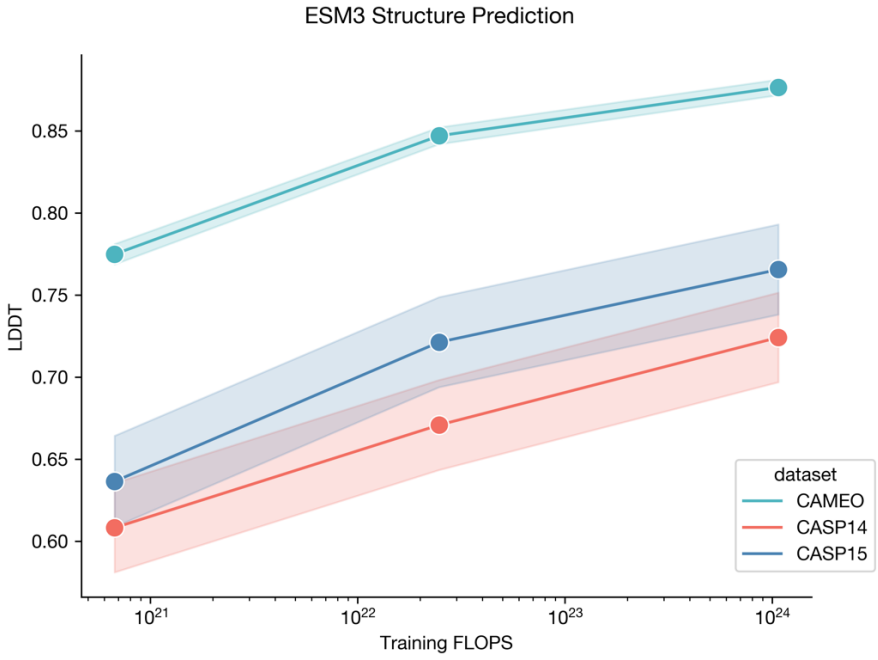

Fig. S10.
Scaling curves for structure prediction. Error bars are ±1 standard error.

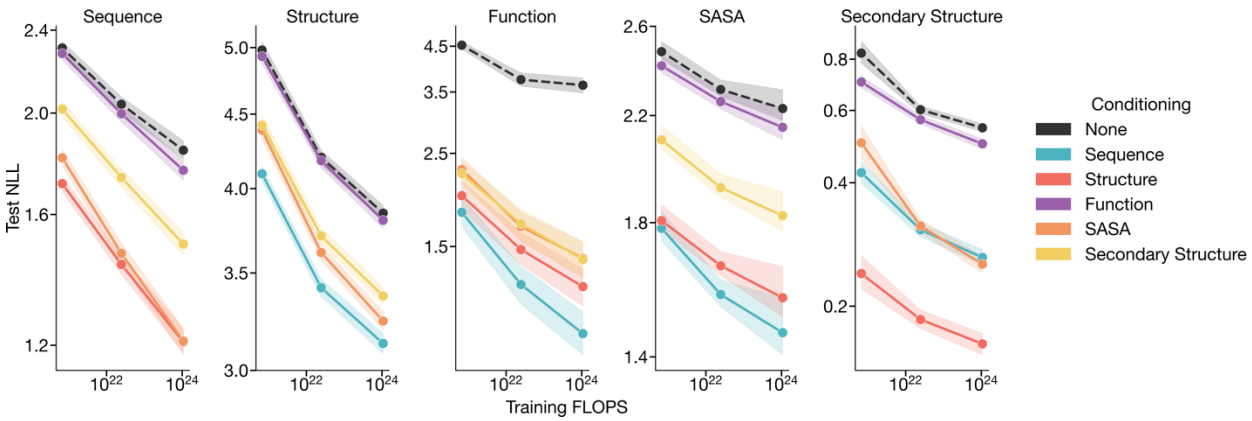

Fig. S11.

Conditional and unconditional scaling behavior for each track. Loss is shown on the test set (Appendix A.3.2). Shaded regions denote 95% confidence intervals.

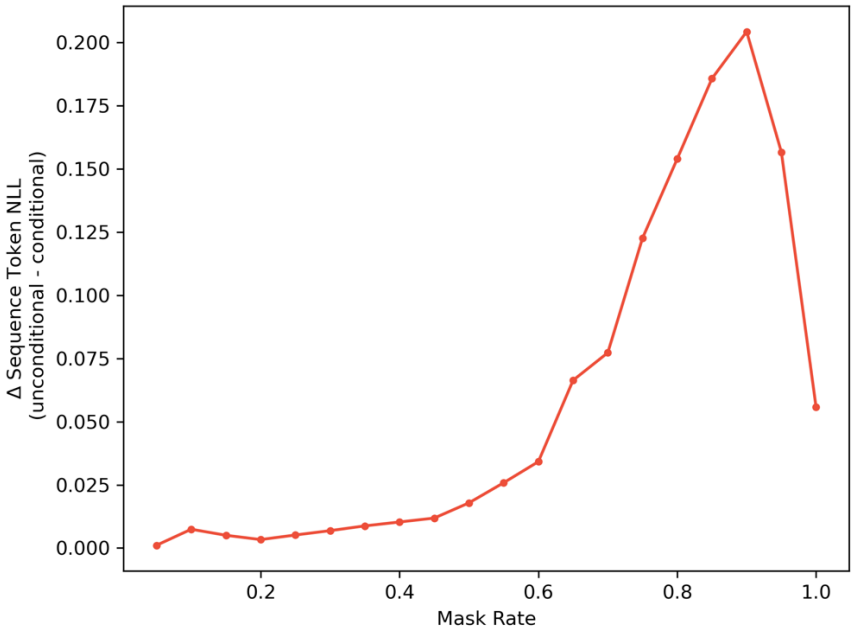

Fig. S12.

Function-conditioned sequence negative log-likelihood over mask rates. Loss is computed on the test set (Appendix A.3.2) using the ESM3 base 7B model. The sequence NLL loss is computed only in sequence positions constrained by function conditioning. The gap in sequence token NLL with and without function conditioning is plotted over a range of sequence token mask rates. Function conditioning drives down the loss at higher mask rates.

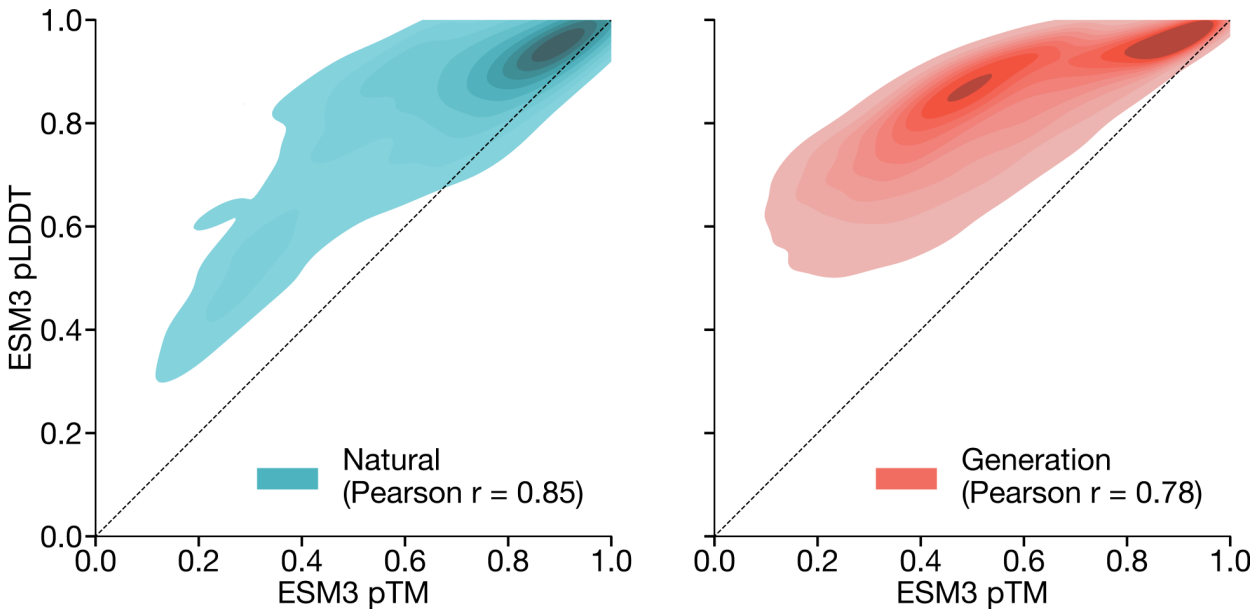

Fig. S13.

Distribution of pTM and pLDDT. Measured on natural (left) and generated (right) sequences under ESM3 7B structure prediction. Generated sequences show a clearly lower correlation (Pearson r 0.79 vs. 0.85) as well as a mode of sequences with high pLDDT but low pTM. Natural sequences are from the test set (Appendix A.3.2), generations are unconditional generations from ESM3 98B.

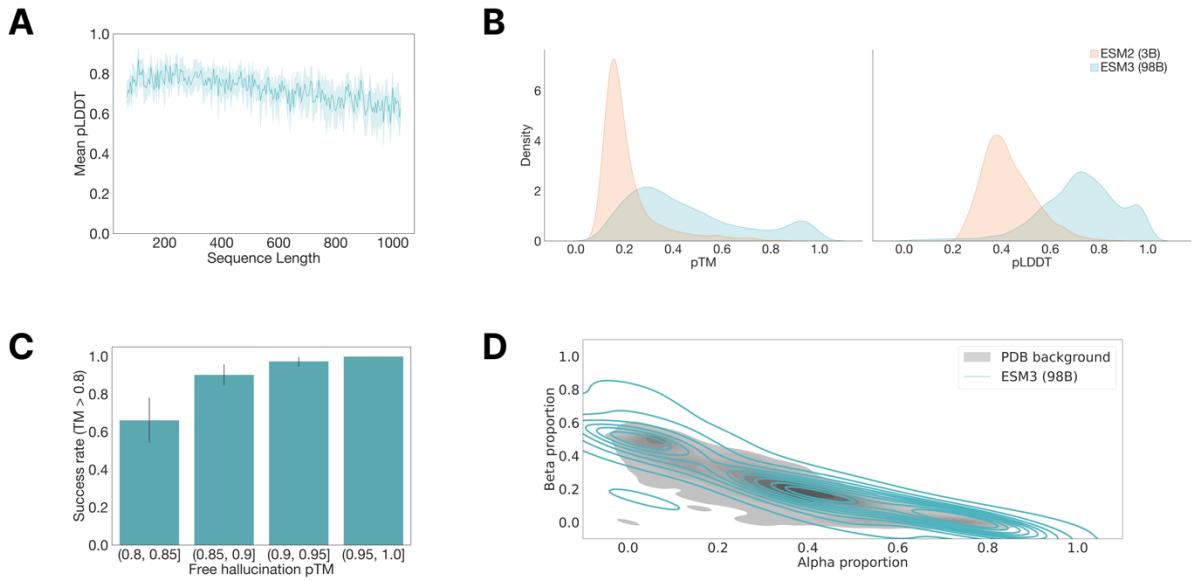

Fig. S14.

Unconditional generation of high-quality and diverse proteins using ESM3. (A) Distribution of ESM3 pLDDT (mean and 95% confidence intervals) for each sequence length in the unconditional generation dataset. (B) pLDDT and pTM of unconditional generations from ESM3 compared to sequences designed using the 3B-parameter ESM2 model. (C) Round-trip success rate of high-confidence generations using ESM3. Predicted structures were inverse folded to predict a new sequence and then re-folded to produce a new structure. Success was measured by a TM-score of greater than 0.8 between the original and refolded designs. (D) Secondary structure composition of unconditional generations relative to the distribution of proteins in the PDB, which is shown in gray.

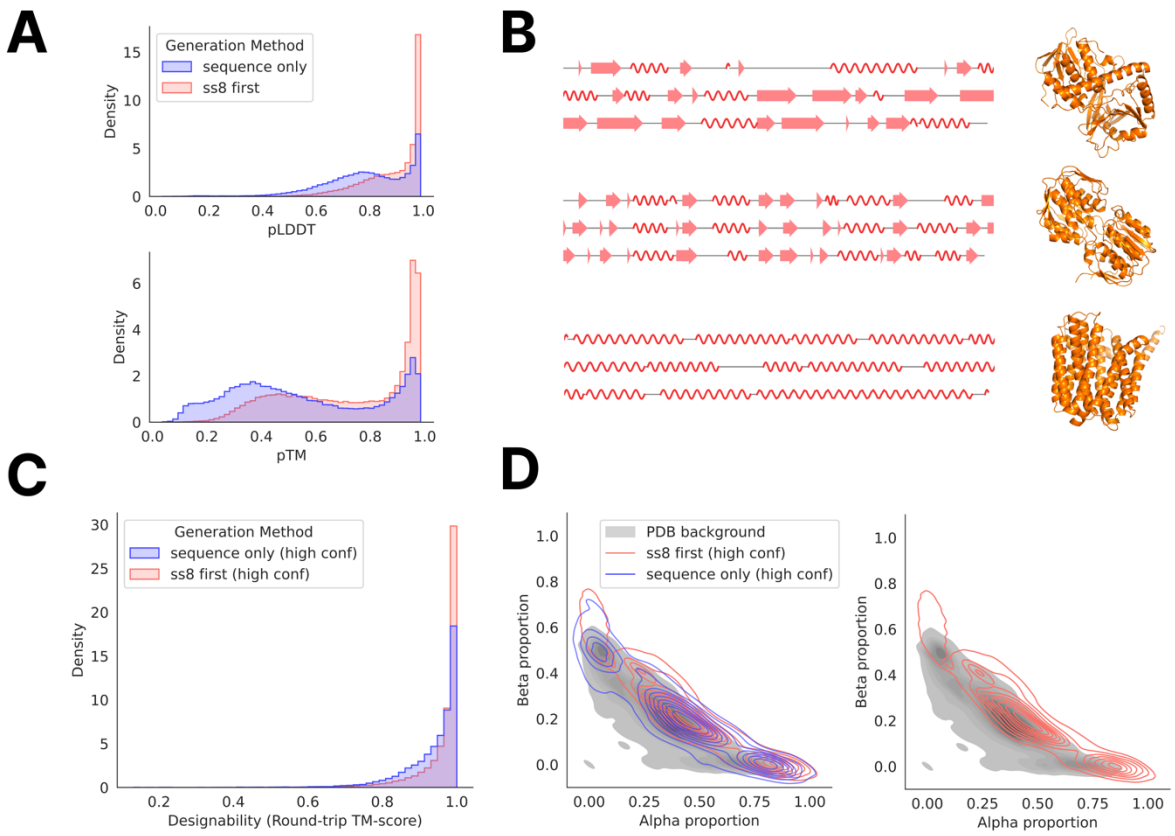

Fig. S15.

Generation of sequences using chain of thought. SS8 tokens are generated first, followed by structure tokens, then amino acid sequence with the ESM3 7B model. (A) Distribution of mean pLDDT and pTM of sequences generated by chain of thought ("ss8 first") compared to directly generating the sequence ("sequence only"). (B) Sample generations of SS8 tokens and the predicted structure of its corresponding CoT sequence. (C) TM-score between predicted structures of high-confidence (pTM > 0.8, mean pLDDT > 0.8) generated sequences and their corresponding inverse folded, then re-folded structures. (D) Comparison of the secondary structure composition of high-confidence generated sequences to the distribution of proteins in the PDB.

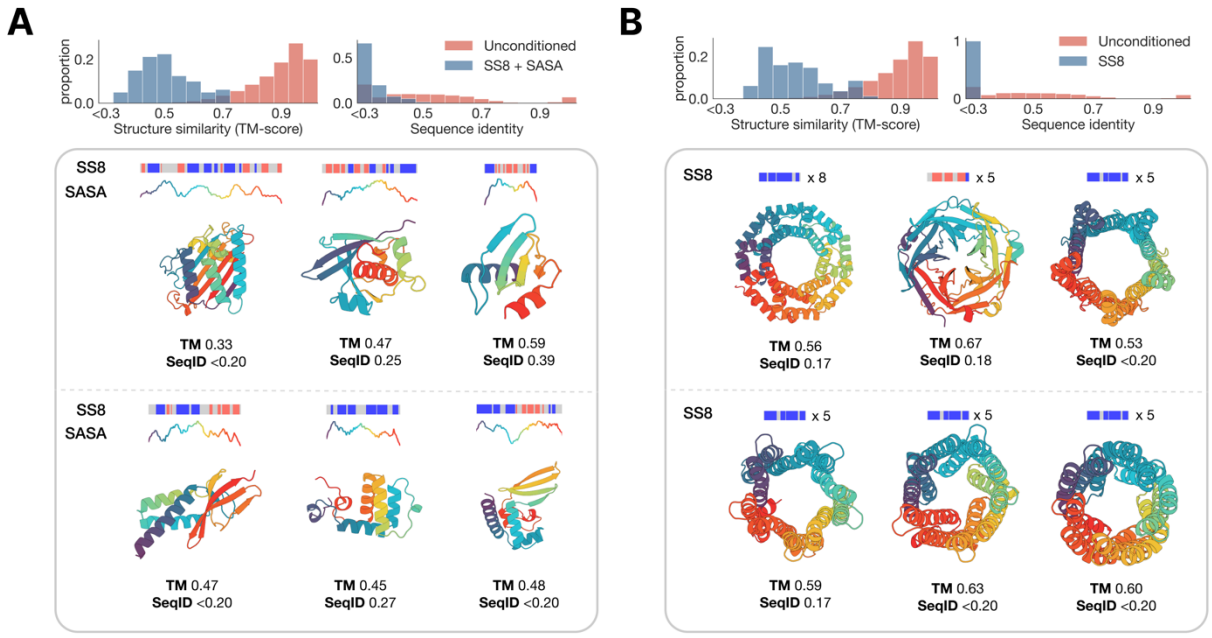

Fig. S16.

Prompting ESM3 to generalize beyond its training distribution. (A) Proteins designed using SS8 and SASA prompts derived from recent structures in the PDB with low structural similarity to the training set. Prompts along the protein length are visualized above each generation; secondary structure is shown using three-class (alpha = blue, beta = orange, coil = gray) and SASA is shown as a line plot colored by residue index to match the cartoon below. (B) Symmetric proteins designed using SS8 prompting. Histograms show the similarity to the nearest training set protein by structure (TM-score) and sequence (sequence identity) compared to unconditional generation.

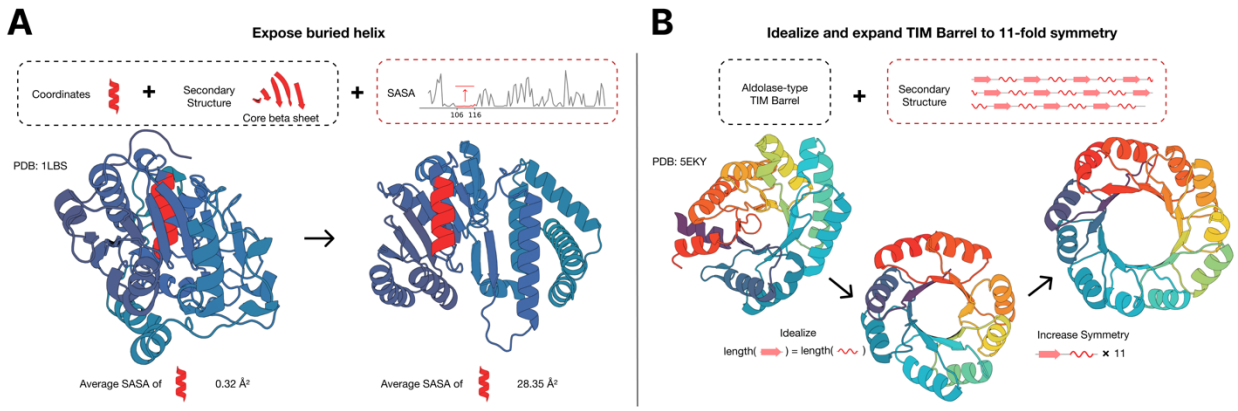

Fig. S17.

Multimodal protein editing with ESM3. (A) ESM3 exposes a buried helix in a protein while maintaining the alternating alpha-beta sandwich fold of the protein. (B) ESM3 is used in a two-step iterative edit, where first secondary structure prompting and function prompting are used to idealize a reference TIM barrel. Secondary structure prompting is then used to increase the number of subunits in the TIM barrel from 8 to 11.

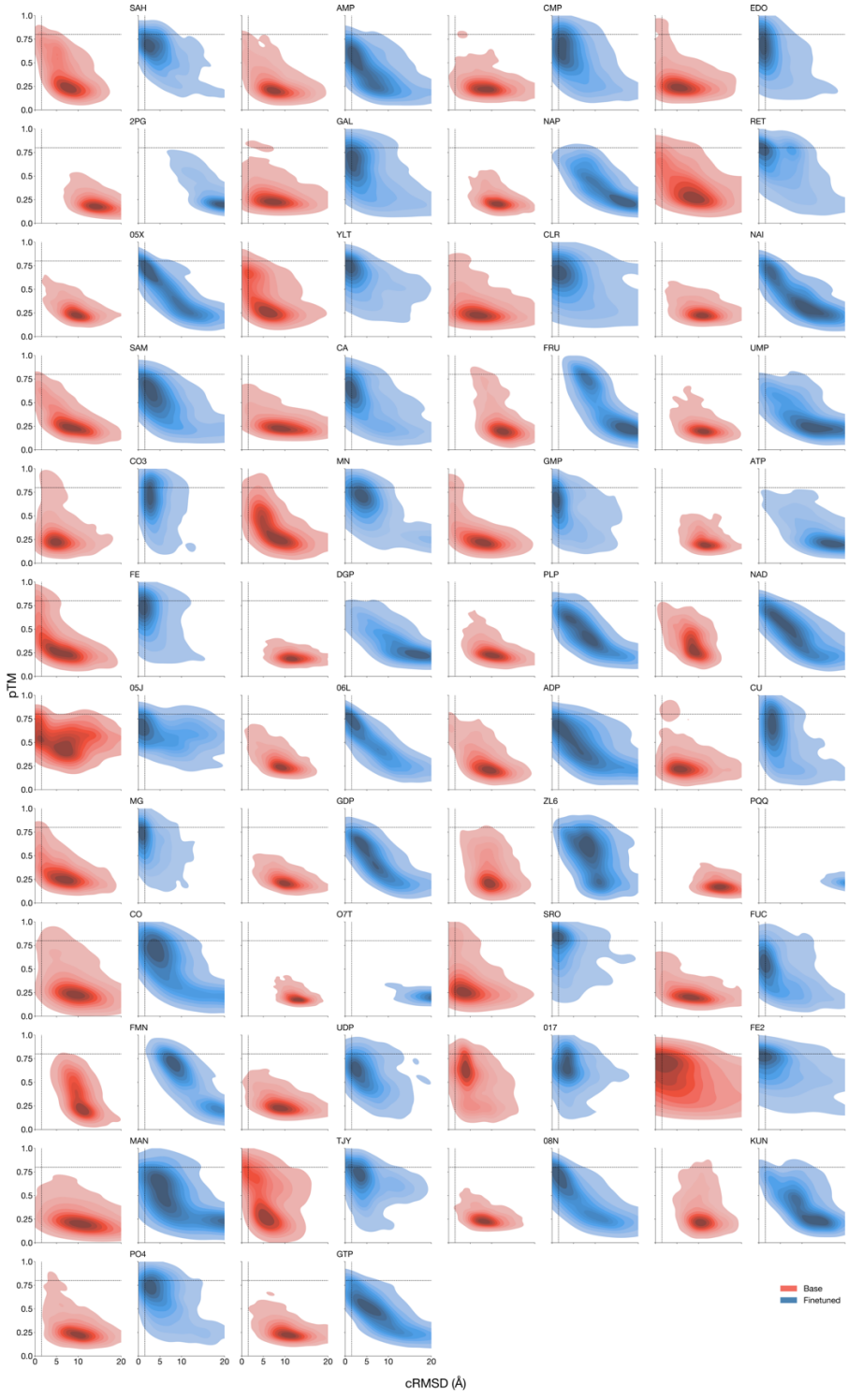

Fig. S18.

Alignment improves model generations. pTM, backbone cRMSD distributions of generations from the 98B base model and aligned model for all ligands in the tertiary coordination dataset. Each ligand/model pair has 1024 generations.

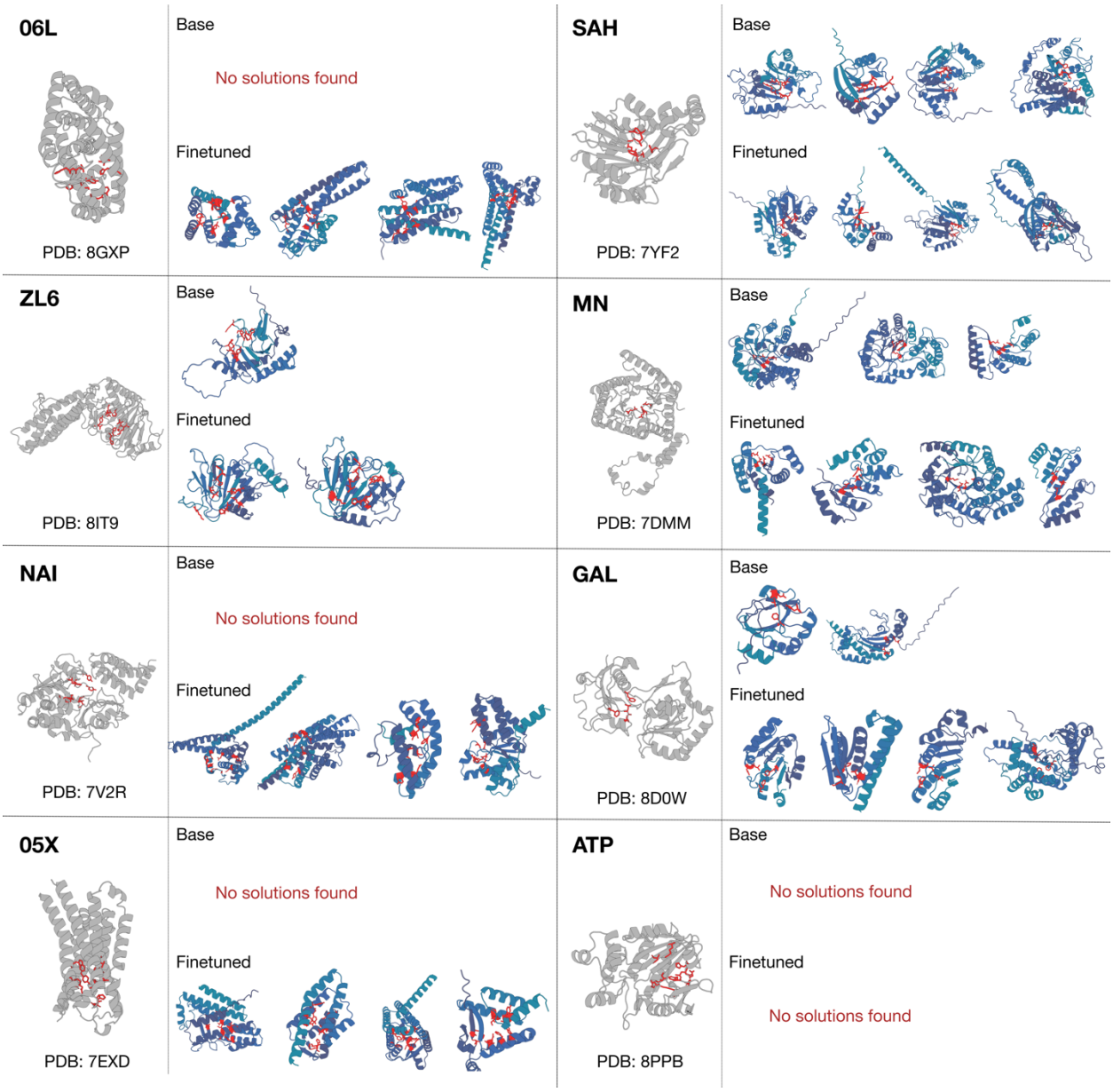

Fig. S19.

Randomly selected successful generations from the base and finetuned 98B parameter models. A random sample of ligands is selected and visualized with the ground truth PDB chain from which the ligand was taken. Solutions produced by ESM3 are diverse, and the finetuned model gives significantly more successes (out of 1024 total samples).

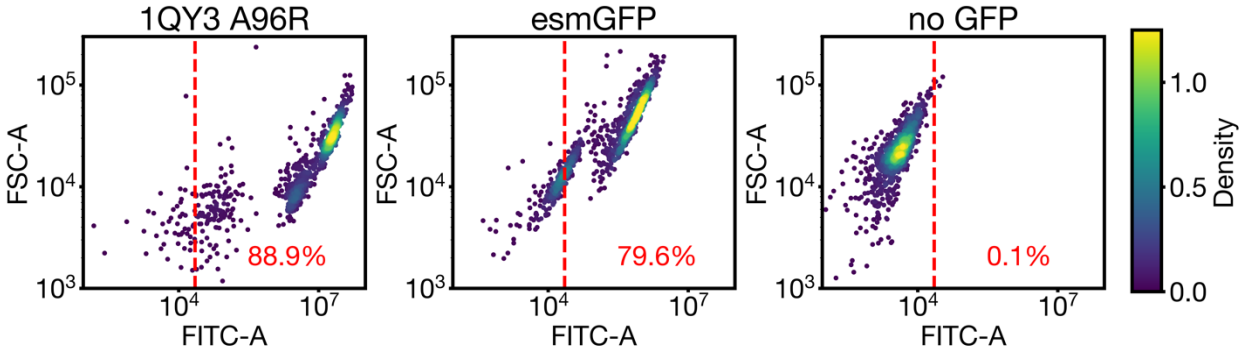

Fig. S20.

Flow cytometry data confirms cells expressing esmGFP can be detected at the single cell level. Forward Scatter-Area (FSC-A), a measure of cell size vs Fluorescein Isothiocyanate-Area (FITC-A), a measure of GFP-like fluorescent signal, for expressing 1QY3 A96R, esmGFP, and a negative control that does not express any GFP. A gate was set at the 99.9% quantile for the negative control data, and the fraction of cells passing the gate were quantified for each sample.

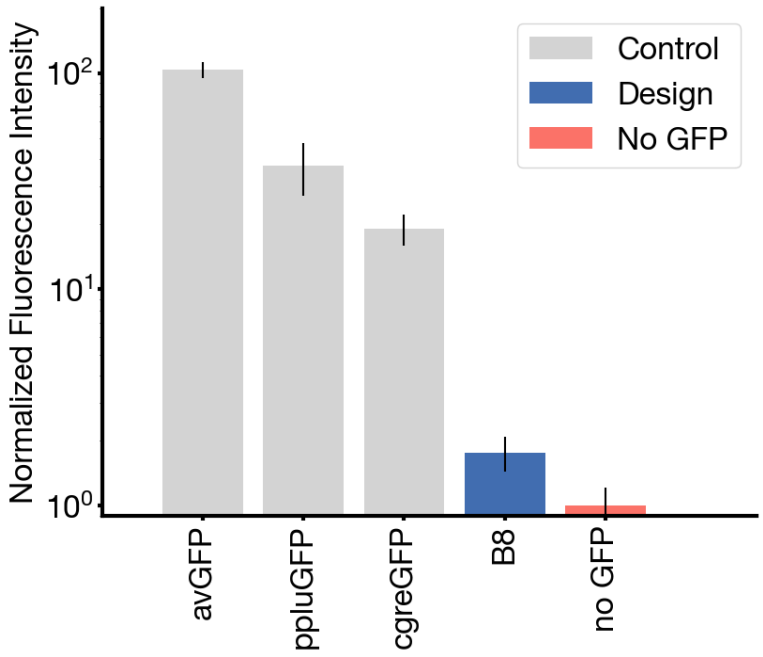

Fig. S21.
Replication of design B8 and select controls. Results are averages of eight wells across two plates.

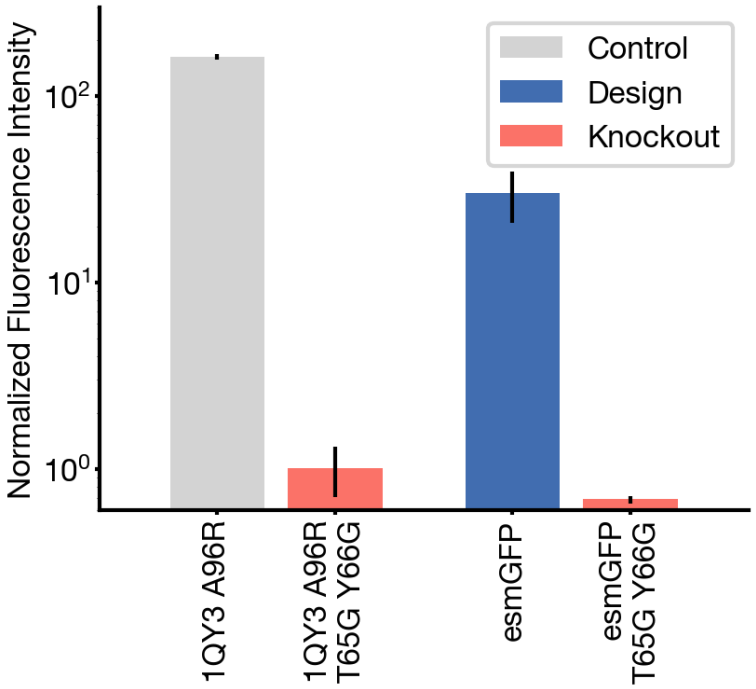

Fig. S22.

Chromophore knockout mutations. T65G and Y66G reduces fluorescence of both 1QY3 A96R and esmGFP to background levels.

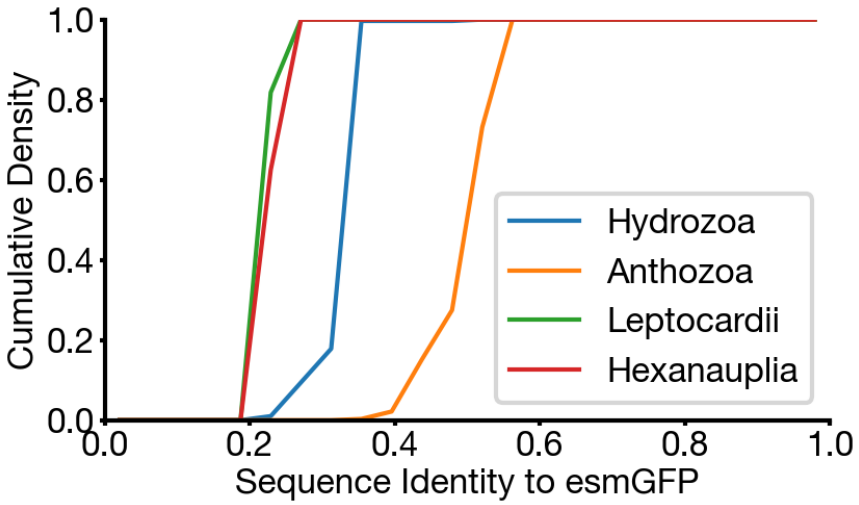

Fig. S23.

Sequence identity of esmGFP. Comparison with natural and designed GFPs from the four major classes found in nature.

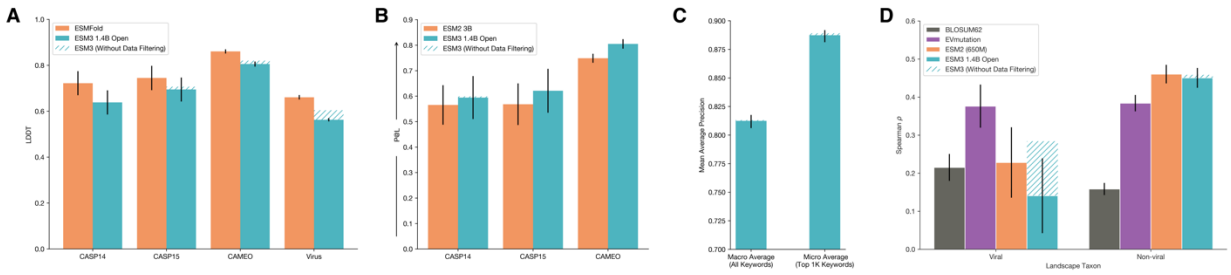

Fig. S24.

ESM3-open is a powerful predictor of structure and function trained for open release. A: Structure Prediction Single-pass structure prediction for ESM3-open (blue; quadratic time complexity) compared with ESMFold (orange; cubic time complexity). See Appendix A.3.4 for details on this evaluation. B: Representation Learning ESM3-open (blue) is competitive with ESM2-3B (orange) on representation learning as measured by contact prediction P@L for finetuned representations. See Appendix A.3.3 for details on this evaluation. C: Function Keyword Prediction. ESM3-open function prediction performance, as measured by Mean Average Precision across function keywords. ESM3-open achieves 0.81 precision across all keywords, and 0.89 for the top 1K most prevalent keywords in the test set (Appendix A.3.2). We use the same evaluation framework as in Appendix A.1.8.2.2. We report both the macro and micro averages as in Fig. S8. In each of the preceding evaluations, the data mitigation minimally impacted performance, as compared to a compute-matched model without data mitigations (hatched blue). D: Zero-shot Fitness Prediction. Fitness prediction performance as measured by correlation (Spearman ρ) across 217 Deep Mutational Scanning datasets collated in ProteinGym. Left and right subplots indicate viral (left) and non-viral (right) DMS datasets. The four columns per group indicate different models. ESM3-open performs substantially worse than EVMutation (purple) on viral fitness prediction, while being competitive with ESM2 (orange) on non-viral fitness prediction. Viral fitness prediction was substantially impacted by the data mitigation, while non-viral fitness prediction was not (hatched blue).

Params	*n* layers	*d* model	*d* head	**Context length**	**Learning rate**	**Warmup steps**	**Batch size in tokens**	**Num steps**	**Total tokens**	**FLOPs**
1.4B	48	1536	64	2048	4.0e-4	5K	1,572,864	50K	~80B	6.72×10^{20}
1.4B	48	1536	64	2048	4.0e-4	5K	1,572,864	200K	~320B	2.7×10^{21}
7.7B	96	2560	128	2048	2.5e-4	5K	3,932,160	140K	~550B	2.47×10^{22}
98.5B	216	6144	128	2048	1.0e-4	20K	4,194,304	430K	~1.8T	1.07×10^{24}

Table S1.

Parameter details for different model configurations.

Stage	Steps	All-atom geometric losses	pAE and pLDDT losses	Augmentation with ESM3 predicted tokens	Context length	Data mixture
2A	90k	✓	✗	✗	512	Roughly uniform sampling of predicted and experimental structures
2B	20k	✓	✓	✓	512	Roughly uniform sampling of predicted and experimental structures
2C	30k	✓	✓	✓	2048	Upsampling experimental structures

Table S2.

Training details for stage 2 training of an all-atom structure token decoder.

Dataset	Type	Clustering Level	Expansion Level	Tokens	Release
UniRef	Sequence	70% (83M)	90% (156M)	54.6B	2023 02
MGnify	Sequence	70% (372M)	90% (621M)	105.5B	2023 02
JGI	Sequence	70% (2029M)	-	256B	All non-restricted studies available on July 30th, 2023.
OAS	Sequence	95% (1192M)	-	132B	All sequences available on July 30th, 2023.
PDB	Structure	- (203K)	-	0.054B	All chains available on RCSB prior to May 1st, 2020
PDB Clustered	Structure	70% (46K)	100% (100K)	0.027B	
AlphaFoldDB	Structure	70% (36M)	90% (69M)	40.5B	v4
ESMAtlas	Structure	70% (87M)	90% (179M)	23.5B	v0, v2023 02

Table S3.

Pretraining dataset statistics. Includes number of tokens, release, and clustering level. Numbers are derived after dataset filtering.

Dataset Name	Unique Samples (M)	Unique Tokens (M)
UniRef	133	40,177
MGnify	406	65,780
JGI	2,039	265,070
OAS	203	22,363
PDB	0.2	55
AFDB	68	20,510
ESMAtlas	168	38,674
AFDB inverse folded	111	33,300
ESMAtlas inverse folded	251	57,730
Sequence	3,143	484,441
Structure	236	177,710
Annotation	539	105,957
Total unique training tokens		768,109

Table S4.
Pretraining unique token statistics. Broken down by token type and dataset type.

Dataset	Inverse Folding	Function Labels	SASA	Secondary Structure
UniRef	✓	✓	-	-
MGnify	✓	✓	-	-
JGI	✗	✗	-	-
OAS	✗	✗	-	-
PDB	✗	✗	✗	✗
AlphaFoldDB	✓	✓	✓	✓
ESMAtlas	✓	✓	✓	✓

Table S5.
Data augmentation and conditioning information applied to each dataset.

Track	Noise Schedule	Dropout Prob	Loss Weight
Sequence	betalinear30	0	1.0
Structure Tokens	cosine	0.25	0.5
Structure Coordinates	cubic	0.5	N/A
Secondary Structure (8-class)	square root	0.9	0.01
SASA	square root	0.9	0.01
Function Tokens	square root	0.9	0.01
Residue Annotations	square root	0.9	0.01

Table S6.
Noise schedules, dropout probabilities, and loss weights.

Reference	Task	Dataset	Holdout Method
Fig. 1D, Table S10, Fig. S11	Conditional and unconditional NLL	CAMEO	Temporal
Fig. 2A	Generative prompt following	CAMEO	Temporal
Fig. S3, Fig. S4	Structure tokenizer reconstruction	CAMEO, CASP14, CASP15	Temporal
Fig. S6	pTM, pLDDT calibration	CAMEO	Temporal
Table S8	Contact prediction	CAMEO, CASP14, CASP15	Temporal
Table S9, Fig. S10	Structure prediction	CAMEO, CASP14, CASP15	Temporal
Fig. S8	Function prediction	UniRef	70% sequence holdout

Table S7.
Datasets for model evaluations.

Model	CASP14	CASP15	CAMEO
ESM2 3B	0.57 (0.49 - 0.64)	0.57 (0.48 - 0.65)	0.75 (0.73 - 0.77)
ESM3 1.4B	0.56 (0.48 - 0.64)	0.59 (0.50 - 0.66)	0.76 (0.74 - 0.78)
ESM3 7B	0.62 (0.54 - 0.70)	0.64 (0.56 - 0.73)	0.82 (0.80 - 0.84)
ESM3 98B	0.66 (0.57 - 0.74)	0.66 (0.57 - 0.75)	0.85 (0.83 - 0.86)

Table S8.

Precision@L results. Measured on CASP14, CASP15 and CAMEO for the ESM3 model family. Intervals represent bootstrapped 95% confidence intervals.

	Iterative / $O(L^3)$			Single pass / $O(L^2)$		
Model	**CAMEO**	**CASP14**	**CASP15**	**CAMEO**	**CASP14**	**CASP15**
1.4B Open	0.801	0.624	0.691	0.806	0.639	0.695
1.4B Overtrained	0.816	0.623	0.704	0.821	0.637	0.707
1.4B	0.771	0.596	0.654	0.777	0.613	0.654
7B	0.845	0.661	0.735	0.848	0.670	0.738
98B	0.880	0.724	0.787	0.879	0.724	0.782
ESMFold	0.861	0.722	0.745			
AlphaFold2	0.903	0.857	0.833			

Table S9.

Protein structure prediction results. We benchmark ESMFold, ESM3 models, and Alphafold2. Left side: ESM3 iterative inference of structure tokens conditioned on sequence. Because iterative inference is O(L3) in length of a protein sequence, it is comparable to the runtime complexity of ESMFold and AF2. Right panel: Single pass argmax prediction of structure tokens given sequence. Iterative decoding does not significantly improve structure prediction performance, and O(L2) runtime complexity is sufficient for structure prediction. Both the Open and Overtrained models are trained up to 200k steps. The 1.4B model is used for scaling comparisons, and is trained to 50k steps.

		Conditioning				
	Model	Sequence	Structure	Function	SASA	Secondary Structure
Sequence	1.4B	*2.31*	1.71	2.28	1.81	2.02
	7B	*2.04*	1.43	2.00	1.47	1.74
	98B	*1.84*	1.21	1.76	1.21	1.50
Structure	1.4B	4.09	*4.98*	4.93	4.39	4.42
	7B	3.42	*4.20*	4.18	3.62	3.71
	98B	3.13	*3.85*	3.80	3.24	3.37
Function	1.4B	1.81	1.98	*4.52*	2.29	2.24
	7B	1.22	1.47	*3.75*	1.67	1.70
	98B	0.93	1.20	*3.63*	1.41	1.40
SASA	1.4B	1.78	1.81	2.42	*2.48*	2.10
	7B	1.57	1.66	2.26	*2.31*	1.92
	98B	1.46	1.56	2.15	*2.23*	1.82
Secondary Structure	1.4B	0.42	0.24	0.70	0.50	*0.83*
	7B	0.31	0.19	0.57	0.31	*0.60*
	98B	0.26	0.16	0.50	0.25	*0.54*

Table S10.

Negative log-likelihood of each track conditioned on other tracks. Each row is a model size, generating a particular modality. Each column is the conditioning. The diagonal, highlighted with italics, are the unconditional NLL of each track. We observe that indeed adding conditioning improves NLL in all cases.

Motif	PDB ID	Chain ID	PDB Residue Identifiers
ACE2 binding	6vw1	A	19-89, 319-366
Ferredoxin	6e6r	A	1-44
Barstar binding	7mrx	B	25-47
P53 binding	1ycr	B	19-28
PD-1 binding	5ius	A	63-83, 119-141
DNA-binding helix-turn-helix	1lcc	A	1-52
P-loop	5ze9	A	229-243
Double EF-hand	1a2x	A	103-115, 139-152
Lactate dehydrogenase	1ldb	A	186-206
Renal dipeptidase	1itu	A	124-147
Ubiquitin-activating enzyme E1C binding	1yov	B	213-223
DNA topoisomerase	1a41	A	248-280

Table S11.

Functional motif definitions for conserved regions.

Scaffold	Reference	InterPro tags	Total Prompt Length
Beta propeller	8siuA	IPR001680 (1-350) IPR036322 (1-350) IPR015943 (1-350)	353
TIM barrel	7rpnA	IPR000652 (0-248) IPR020861 (164-175) IPR035990 (0-249) IPR013785 (0-251) IPR000652 (2-249) IPR022896 (1-249)	252
MFS transporter	4ikvA	IPR011701 (1-380) IPR020846 (1-380) IPR036259 (1-380)	380
Immunoglobulin	7sbdH	IPR036179 (0-116; 124-199) IPR013783 (0-206) IPR003597 (124-202) IPR007110 (0-115; 121-207) IPR003599 (6-115) IPR013106 (11-114)	209
Histidine kinase	8dvqA	IPR003594 (47-156) IPR003594 (47-158) IPR004358 (118-137) IPR004358 (141-155) IPR004358 (101-112) IPR005467 (0-158) IPR036890 (4-159) IPR036890 (3-156)	166
Alpha/beta hydrolase	7yiiA	IPR029058 (0-274) IPR000073 (26-265)	276

Table S12.

InterPro tags extracted from CAMEO test set proteins for prompting with fold specification.

Site	Scaffold	Novelty (TM to original)	Designability (scTM)
017	beta	0.264	0.967
ACE2	alpha	0.606	0.871
CA	Immunoglobulin	0.441	0.781
Double-EF-hand	ab-hydrolase	0.293	0.969
MG	TIM-barrel	0.328	0.980
Renal-dipeptidase	alpha-beta-alpha	0.644	0.933
SRO	mfs-transporter	0.345	0.992
Topoisomerase	histidine-kinase	0.269	0.948
YLT	alpha-beta	0.229	0.899
ZN	alpha	0.567	0.996

Table S13.

Novelty and designability metrics. Metrics for motif scaffolds shown in Fig. 2C. Novelty is measured by computing the TM-score to the original scaffold from which the motif is derived. Designability is measured by self-consistency TM-score over eight samples by inverse folding with ESM-IF1 and refolding with ESMFold. All designs are distinct from their original scaffolds while retaining high designability.

Task	Prompt Length Variation	Prompt Motif Position Variation	Folding Method
Fig. 1E (Unconditional generation)	Yes; 100 protein lengths sampled from PDB	No constraints	ESM3 7B with T=0.0
Fig. 2A (Prompt following for structure coordinates, SS8, SASA, keywords)	No; same as template	No; same as template	ESMFold
Fig. 2B (Prompting for out-of-distribution folds)	No; same as template	No; same as template	ESM3 7B with T=0.0
Fig. 2B (Prompting for symmetric proteins)	No; same as template	No; same as template	ESM3 7B with T=0.0
Fig. 2C (Secondary structure + active site prompting)	Yes; sampled uniformly in [150, 400]	Yes; SS spans sampled randomly, active site positions sampled randomly	ESMFold
Fig. 2C (Keyword + active site prompting)	No; same as template	Yes; active site positions sampled randomly, keyword positions fixed	ESMFold
Fig. 2D (Compress structure)	Yes; reduced to 150 residues	Yes; active site positions sampled randomly, keyword positions fixed	ESMFold
Fig. 3 (Alignment for atomic coordination)	Yes; sampled uniformly between 150, 250, 350 residues	Yes; active site positions sampled randomly	ESMFold
Fig. 4A (GFP Design)	No; same as template	No; chromophore positions fixed	ESM3 7B with T=0.0
Fig. S14 (Unconditional generation)	Yes; sampled uniformly every 4 values between 64 and 1024	No constraints	ESM3 7B with T=0.0
Fig. S15 (Unconditional generation with a chain of thought)	Yes; 100 protein lengths sampled from PDB	No constraints	ESM3 7B with T=0.0
Fig. S16 (Prompting for out-of-distribution and symmetric proteins)	No; same as template	No; same as template	ESM3 7B with T=0.0
Fig. S17 (Expose buried helix)	No; same as template	No; helix and core sheet positions held fixed	ESMFold
Fig. S17 (Idealize + expand TIM barrel)	Yes; increased to achieve 11 fold symmetry	Yes; SS8 prompt repeated 11x to achieve 11-fold symmetry	ESMFold
Fig. S18, S19 (Alignment for atomic coordination)	Yes; sampled uniformly between 150, 250, 350 residues	Yes; active site positions sampled randomly	ESMFold

Table S14.

Prompt variation and folding methods for ESM3 design tasks.

PDB ID	Coordinating Residues	Ligand ID
7map	D25 G27 A28 D29 D30 G48 G49 V50	017
7n3u	I305 F310 V313 A326 K328 N376 C379 G382 D386 F433	05J
7exd	D103 I104 C107 T108 I174 H176 T182 W306 F309 E313 Y337	05X
8gxp	W317 C320 A321 H323 V376 F377 L396 I400 H479 Y502	06L
7n4z	M66 C67 R124 L130 C134 Y135 D152 F155	08N
7vrd	A40 S41 H161 Q169 E170 E213 D248 D324 K349 H377 R378 S379 K400	2PG
7zyk	V53 V66 V116 H160 N161 I174 D175	ADP
6yj7	K23 V24 A25 Y45 T46 A47 F115 I128	AMP
8ppb	H185 F198 K209 Q249 D250 L251 D262 K336 I415 D416	ATP
7knv	E33 F94 E95 D125	CA
7xer	Y466 L505 T525	CLR
7tj6	F366 G367 T378 R418	CMP
6xm7	H167 H218 H284 H476	CO
7bfr	Q62 X126 H248	CO3
6xlr	X272 Y495 H496 H581	CU
6tnh	N40 A41 S127 T128 Q187 L191 C201 T202 V236	DGP
7ndr	F73 S101 F102 D103 R106	EDO
8axy	H68 H109 E144	FE
7o6c	E62 E107 Q141	FE2
8aul	P31 M32 T33 Q106 H185 R237 S319 G320 G321 G342 R343 F369 Y370	FMN
7vcp	N37 D38 Q54 F97 S98 R159 D160 E214 Y276 W297	FRU
7b7f	G167 T168 G189 W195	FUC
8d0w	F73 L136 E137 F329	GAL
7yua	T13 T14 I15 D40 H85 S86 D87 D110 N290	GDP
7w1a	L44 Y88 L91 I212	GMP
7ljn	G71 S72 D91 K236 S253 V254 D309 R310	GTP
6s4f	Y84 N87 K88 V131 Q132 L133 D155 F157 I276 P309 G310 G313 P314 V317	KUN
7mg7	Y12 G98 L99 Y100 A207 D208 G227 R228	MAN
7qow	D12 T118 E268	MG
7dmm	E181 E217 D245 D287	MN
7qoz	G11 G12 I13 Y34 D35 V36 A86 G87 V126 T127 N128 H185 M235	NAD
7v2r	G89 F93 K98 F101 E121 Y204 E209 F229	NAI
7a7b	F51 Y128 K165 N166 S167 Y186 R187 I248 G249 A299	NAP
7pae	M20 L22 L38 V49 I53 C56 K57 R61 Q78 V80 W90 I109 M117 I129 L147 Y149	O7T
8egy	H82 K83 S186 G230 S231 N232 E345 S368 G369	PLP
7qow	S65 R129 D273 H465	PO4
7wmk	E77 L124 R129 S174 T189 Q191 W241 D304 E306 K349 D410 W411 Y486	PQQ
7pl9	D607 A608 Y637 M638 Y705 G706 M735 K736	RET
7yf2	G153 E174 L175 L209 N210 L211 Y295	SAH
7v6j	G207 D230 L231 D250 M251 K264	SAM
7ys6	D106 C110 N288	SRO
6w8m	A22 A23 G70 S110 T111 G112 V113 Y114	TJY
8g27	S258 D294 K435 R717	UDP
7xyk	R24 C170 R190 S191 D193 N201 H231 Y233	UMP
8g3s	H224 F228 V249 M250 V253 R263 T266 L267 F270	YLT
8it9	T92 P93 R96 Y108 L109 K216 V228 S229 H231 H232	ZL6

Table S15.

Tertiary coordination dataset. Selected PDBs and coordinating residues (along with binding ligand) for each protein sample in the tertiary coordination dataset.

	Model	Alignment	Pass@128	Fraction ≥ *X* Solutions		
				8	16	32
Without CoT	1.4B	Base	0.151 ± 0.010	0.087	0.044	0.000
		Finetuned	0.188 ± 0.010	0.174	0.109	0.044
	7B	Base	0.190 ± 0.010	0.130	0.087	0.022
		Finetuned	0.374 ± 0.011	0.304	0.283	0.152
	98B	Base	0.268 ± 0.011	0.196	0.152	0.044
		Finetuned	0.655 ± 0.010	0.609	0.500	0.348
With CoT	1.4B	Base	0.140 ± 0.010	0.087	0.065	0.022
		Finetuned	0.207 ± 0.010	0.130	0.087	0.044
	7B	Base	0.187 ± 0.011	0.152	0.044	0.022
		Finetuned	0.421 ± 0.011	0.391	0.283	0.196
	98B	Base	0.251 ± 0.013	0.152	0.065	0.022
		Finetuned	0.666 ± 0.011	0.630	0.522	0.304
	RFDiffusion	N/A	0.476 ± 0.012	0.413	0.370	0.261

Table S16.

Tertiary coordination benchmark. Fraction of tertiary coordination tasks solved with 128 generations (Pass@128; 2 s.d.), and fraction of tasks with at least 8, 16, and 32 distinct solutions (clustered at TM > 0.8) given a budget of 1024 generations. ESM3 models can generate sequences either directly from a prompt ("Without CoT") or via a chain of thought by first generating structure tokens, then sequence ("With CoT"). Chain of thought improves success rates in finetuned models. The ESM3 98B finetuned model with chain of thought achieves the strongest performance.

Protein	Sequence Identity to esmGFP	Mutations to esmGFP	Aligned Sequence
B8	0.93	15	-MSKVEELIKPEMKMKLEMEGEVNGHKFSIEAEGEGKPYEGKQTIKAWSTT-GKLPFAW DILSTSLTYGFRMFTKYPEGLEEHDYFKQSFPEGYSWERTITYEDGATVKVTSDISLED GVLINKIKFKGTNFPSDGPVM-QKKTTGWEPSTELITPDPATGGLKGEVKMRLKLEGGG HLLADFKTTYRSKKKEK-LPLPGVHYVDHTIRNEKAPHPEGKEYVVQYETAVARLA--- --------
esmGFP	1.0	0	-MSKVEELIKPDMKMKLEMEGEVNGHKFSIEAEGEGKPYEGKQTIKAWSTT-GKLPFAW DILSTSLTYGNRAFTKYPEGLEQHDFFKQSFPEGYSWERTITYEDGATVKVTADISLED GVLINKVKFKGENFPSDGPVM-QKKTTGWEASTELITPDPATGGLKGEVKMRLKLEGGG HLLADFKTTYRSKKKEK-LPLPGVHYVDHRIVNEKATHPEGKEYMIQYEHAVARLA--- --------
tagRFP	0.58	96	MVSKGEELIKENMHMKLYMEGTVNNHHFKCTSEGEGKPYEGTQTMRIKVVEGGPLPFAF DILATSFMYGSRTFINHTQGIP--DFFKQSFPEGFTWERVTTYEDGGVLTATQDTSLQD GCLIYNVKIRGVNFPSNGPVM-QKKTLGWEANTEMLY--PADGGLEGRTDMALKLVGGG HLICNFKTTYRSKKPAKNLKMPGVYYVDHRL--ERIKEADKETYVEQHEVAVARYCDLP SKLGHKLN
eqFP578	0.53	107	----MSELIKENMHMKLYMEGTVNNHHFKCTSEGERKPYEGTQTMKIKVVEGGPLPFAF DILATSFMYGSKTFINHTQGIP--DLFKQSFPEGFTWERITTYEDGGVLTATQDTSLQN GCIIYNVKINGVNFPSNGSVM-QKKTLGWEANTEMLY--PADGGLRGHSQMALKLVGGG YLHCSFKTTYRSKKPAKNLKMPGFHFVDHRL--ERIKEADKETYVEQHEMAVAKYCDLP SKLGHR--
template	0.38	143	-MSKGEELFTGVVPILVELDGDVNGHKFSVSGEGEGDATYGKLTLKFICTT-GKLPVPW PTLVTTLTYGVQCFSRYPDHMKQHDFFKSAMPEGYVQERTISFKDDGNYKTRAEVKFEG DTLVNRIELKGIDFKEDGNILGHKLEYNYNSHNVYITADKQKNGIKANFKIRHNIEDGS VQLADHYQQNTPIGDGP-VLLPDNHYLSTQSALSKDPN-EKRDHMVLLEFVTAAGI--- --------
avGFP	0.36	146	-MSKGEELFTGVVPILVELDGDVNGHKFSVSGEGEGDATYGKLTLKFICTT-GKLPVPW PTLVTTFSYGVQCFSRYPDHMKQHDFFKSAMPEGYVQERTIFFKDDGNYKTRAEVKFEG DTLVNRIELKGIDFKEDGNILGHKLEYNYNSHNVYIMADKQKNGIKVNFKIRHNIEDGS VQLADHYQQNTPIGDGP-VLLPDNHYLSTQSALSKDPN-EKRDHMVLLEFVTAAGITHG MDELYK--

Table S17.

Multiple sequence alignment of select GFP designs (B8, esmGFP) and reference proteins. Template is the full sequence of our template structure (PDB ID 1QY3), with chromophore slowing mutation A96R removed. tagRFP is the full sequence of the top hit returned by BLAST search of the nonredundant database nr, avGFP and eqFP578 are from FPBase. Sequence identities for GFP designs are in general calculated as the number of non-gap matches at aligned positions, divided by the minimum length of the query and target ungapped sequences. Here, only sequence identities to esmGFP are shown. Similarly, the number of mutations to esmGFP are calculated as the number of mismatches at aligned positions where esmGFP does not have a gap.

References and Notes

1. UniProt Consortium, UniProt: A hub for protein information. *Nucleic Acids Res.* **43**, D204–D212 (2015). doi:10.1093/nar/gku989 Medline

2. I. V. Grigoriev, H. Nordberg, I. Shabalov, A. Aerts, M. Cantor, D. Goodstein, A. Kuo, S. Minovitsky, R. Nikitin, R. A. Ohm, R. Otillar, A. Poliakov, I. Ratnere, R. Riley, T. Smirnova, D. Rokhsar, I. Dubchak, The genome portal of the department of energy joint genome institute. *Nucleic Acids Res.* **40**, D26–D32 (2012). doi:10.1093/nar/gkr947 Medline

3. A. L. Mitchell, A. Almeida, M. Beracochea, M. Boland, J. Burgin, G. Cochrane, M. R. Crusoe, V. Kale, S. C. Potter, L. J. Richardson, E. Sakharova, M. Scheremetjew, A. Korobeynikov, A. Shlemov, O. Kunyavskaya, A. Lapidus, R. D. Finn, MGnify: The microbiome analysis resource in 2020. *Nucleic Acids Res.* **48**, D570–D578 (2020). Medline

4. M. Varadi, D. Bertoni, P. Magana, U. Paramval, I. Pidruchna, M. Radhakrishnan, M. Tsenkov, S. Nair, M. Mirdita, J. Yeo, O. Kovalevskiy, K. Tunyasuvunakool, A. Laydon, A. Žídek, H. Tomlinson, D. Hariharan, J. Abrahamson, T. Green, J. Jumper, E. Birney, M. Steinegger, D. Hassabis, S. Velankar, AlphaFold Protein Structure Database in 2024: Providing structure coverage for over 214 million protein sequences. *Nucleic Acids Res.* **52**, D368–D375 (2024). doi:10.1093/nar/gkad1011 Medline

5. Z. Lin, H. Akin, R. Rao, B. Hie, Z. Zhu, W. Lu, N. Smetanin, R. Verkuil, O. Kabeli, Y. Shmueli, A. Dos Santos Costa, M. Fazel-Zarandi, T. Sercu, S. Candido, A. Rives, Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science* **379**, 1123–1130 (2023). doi:10.1126/science.ade2574 Medline

6. E. C. Alley, G. Khimulya, S. Biswas, M. AlQuraishi, G. M. Church, Unified rational protein engineering with sequence-based deep representation learning. *Nat. Methods* **16**, 1315–1322 (2019). doi:10.1038/s41592-019-0598-1 Medline

7. M. Heinzinger, A. Elnaggar, Y. Wang, C. Dallago, D. Nechaev, F. Matthes, B. Rost, Modeling aspects of the language of life through transfer-learning protein sequences. *BMC Bioinformatics* **20**, 723 (2019). doi:10.1186/s12859-019-3220-8 Medline

8. A. Rives, J. Meier, T. Sercu, S. Goyal, Z. Lin, J. Liu, D. Guo, M. Ott, C. L. Zitnick, J. Ma, R. Fergus, Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proc. Natl. Acad. Sci. U.S.A.* **118**, e2016239118 (2021). doi:10.1073/pnas.2016239118 Medline

9. A. Madani, B. Krause, E. R. Greene, S. Subramanian, B. P. Mohr, J. M. Holton, J. L. Olmos Jr., C. Xiong, Z. Z. Sun, R. Socher, J. S. Fraser, N. Naik, Large language models generate functional protein sequences across diverse families. *Nat. Biotechnol.* **41**, 1099–1106 (2023). doi:10.1038/s41587-022-01618-2 Medline

10. N. Ferruz, S. Schmidt, B. Höcker, ProtGPT2 is a deep unsupervised language model for protein design. *Nat. Commun.* **13**, 4348 (2022). doi:10.1038/s41467-022-32007-7 Medline

11. R. Verkuil, O. Kabeli, Y. Du, B. I. Wicky, L. F. Milles, J. Dauparas, D. Baker, S. Ovchinnikov, T. Sercu, A. Rives, Language models generalize beyond natural proteins. bioRxiv 521521 [Preprint] (2022); https://doi.org/10.1101/2022.12.21.521521.

12. A. Elnaggar, M. Heinzinger, C. Dallago, G. Rihawi, Y. Wang, L. Jones, T. Gibbs, T. Feher, C. Angerer, D. Bhowmik, B. Rost, ProtTrans: Understanding the language of life through self-supervised learning. *IEEE Trans. Pattern Anal. Mach. Intell.* **14**, 7112–7127 (2022). doi:10.1109/TPAMI.2021.3095381 Medline

13. D. Hesslow, N. Zanichelli, P. Notin, I. Poli, D. Marks, RITA: A study on scaling up generative protein sequence models. arXiv:2205.05789 [q-bio.QM] (2022).

14. E. Nijkamp, J. A. Ruffolo, E. N. Weinstein, N. Naik, A. Madani, ProGen2: Exploring the boundaries of protein language models. *Cell Syst.* **14**, 968–978.e3 (2023). doi:10.1016/j.cels.2023.10.002 Medline

15. S. Alamdari, N. Thakkar, R. van den Berg, A. X. Lu, N. Fusi, A. P. Amini, K. K. Yang, Protein generation with evolutionary diffusion: sequence is all you need. bioRxiv 556673 [Preprint] (2023); https://doi.org/10.1101/2023.09.11.556673.

16. M. Heinzinger, K. Weissenow, J. G. Sanchez, A. Henkel, M. Mirdita, M. Steinegger, B. Rost, Bilingual language model for protein sequence and structure. bioRxiv 550085 [Preprint] (2024); https://doi.org/10.1101/2023.07.23.550085.

17. J. Su, C. Han, Y. Zhou, J. Shan, X. Zhou, F. Yuan, SaProt: Protein language modeling with structureaware vocabulary. bioRxiv 560349 [Preprint] (2023); https://doi.org/10.1101/2023.10.01.560349.

18. J. Meier, R. Rao, R. Verkuil, J. Liu, T. Sercu, A. Rives, Language models enable zero-shot prediction of the effects of mutations on protein function. bioRxiv 450648 [Preprint] (2021); https://doi.org/10.1101/2021.07.09.450648.

19. J. Vig, A. Madani, L. R. Varshney, C. Xiong, R. Socher, N. F. Rajani, BERTology meets biology: Interpreting attention in protein language models. arXiv:2006.15222 [cs.CL] (2020).

20. R. Rao, J. Meier, T. Sercu, S. Ovchinnikov, A. Rives, Transformer protein language models are unsupervised structure learners. bioRxiv 422761 [Preprint] (2021); https://doi.org/10.1101/2020.12.15.422761.

21. B. Chen, X. Cheng, P. Li, Y. Geng, J. Gong, S. Li, Z. Bei, X. Tan, B. Wang, X. Zeng, C. Liu, A. Zeng, Y. Dong, J. Tang, L. Song, xTrimoPGLM: Unified 100B-scale pre-trained transformer for deciphering the language of protein. bioRxiv 547496 [Preprint] (2023); https://doi.org/10.1101/2023.07.05.547496.

22. J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, D. Amodei, Scaling laws for neural language models. arXiv:2001.08361 [cs.LG] (2020).

23. T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language models are few-shot learners. arXiv:2005.14165 [cs.CL] (2020).

24. J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. d. Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, O. Vinyals, L. Sifre, Training compute-optimal large language models. arXiv:2203.15556 [cs.CL] (2022).

25. J. Abramson, J. Adler, J. Dunger, R. Evans, T. Green, A. Pritzel, O. Ronneberger, L. Willmore, A. J. Ballard, J. Bambrick, S. W. Bodenstein, D. A. Evans, C.-C. Hung, M. O'Neill, D. Reiman, K. Tunyasuvunakool, Z. Wu, A. Žemgulytė, E. Arvaniti, C. Beattie, O. Bertolli, A. Bridgland, A. Cherepanov, M. Congreve, A. I. Cowen-Rivers, A. Cowie, M. Figurnov, F. B. Fuchs, H. Gladman, R. Jain, Y. A. Khan, C. M. R. Low, K. Perlin, A. Potapenko, P. Savy, S. Singh, A. Stecula, A. Thillaisundaram, C. Tong, S. Yakneen, E. D. Zhong, M. Zielinski, A. Žídek, V. Bapst, P. Kohli, M. Jaderberg, D. Hassabis, J. M. Jumper, Accurate structure prediction of biomolecular interactions with AlphaFold 3. *Nature* **630**, 493–500 (2024). doi:10.1038/s41586-024-07487-w Medline

26. J. L. Watson, D. Juergens, N. R. Bennett, B. L. Trippe, J. Yim, H. E. Eisenach, W. Ahern, A. J. Borst, R. J. Ragotte, L. F. Milles, B. I. M. Wicky, N. Hanikel, S. J. Pellock, A. Courbet, W. Sheffler, J. Wang, P. Venkatesh, I. Sappington, S. V. Torres, A. Lauko, V. De Bortoli, E. Mathieu, S. Ovchinnikov, R. Barzilay, T. S. Jaakkola, F. DiMaio, M. Baek, D. Baker, De novo design of protein structure and function with RFdiffusion. *Nature* **620**, 1089–1100 (2023). doi:10.1038/s41586-023-06415-8 Medline

27. J. B. Ingraham, M. Baranov, Z. Costello, K. W. Barber, W. Wang, A. Ismail, V. Frappier, D. M. Lord, C. Ng-Thow-Hing, E. R. Van Vlack, S. Tie, V. Xue, S. C. Cowles, A. Leung, J. V. Rodrigues, C. L. Morales-Perez, A. M. Ayoub, R. Green, K. Puentes, F. Oplinger, N. V. Panwar, F. Obermeyer, A. R. Root, A. L. Beam, F. J. Poelwijk, G. Grigoryan, Illuminating protein space with a programmable generative model. *Nature* **623**, 1070–1078 (2023). doi:10.1038/s41586-023-06728-8 Medline

28. Y. Lin, M. Lee, Z. Zhang, M. AlQuraishi, Out of many, one: Designing and scaffolding proteins at the scale of the structural universe with Genie 2, May 2024. arXiv:2405.15489 [q-bio.BM] (2024).

29. O. Shimomura, F. H. Johnson, Y. Saiga, Extraction, purification and properties of aequorin, a bioluminescent protein from the luminous hydromedusan, Aequorea. *J. Cell. Comp. Physiol.* **59**, 223–239 (1962). doi:10.1002/jcp.1030590302 Medline

30. R. Y. Tsien, The green fluorescent protein. *Annu. Rev. Biochem.* **67**, 509–544 (1998). doi:10.1146/annurev.biochem.67.1.509 Medline

31. J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805 [cs.CL] (2018).

32. H. Chang, H. Zhang, L. Jiang, C. Liu, W. T. Freeman, Maskgit: Masked generative image transformer. arXiv:2202.04200 [cs.CV] (2022).

33. B. Uria, I. Murray, H. Larochelle, A deep and tractable density estimator. arXiv:1310.1757 [stat.ML] (2014).

34. J. Austin, D. D. Johnson, J. Ho, D. Tarlow, R. van den Berg. Structured denoising diffusion models in discrete state-spaces. arXiv:2107.03006 [cs.LG] (2023).

35. A. van den Oord, O. Vinyals, K. Kavukcuoglu, Neural discrete representation learning. arXiv:1711.00937 [cs.LG] (2017).

36. B. E. Suzek, Y. Wang, H. Huang, P. B. McGarvey, C. H. Wu; UniProt Consortium, UniRef clusters: A comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics* **31**, 926–932 (2015). doi:10.1093/bioinformatics/btu739 Medline

37. L. Richardson, B. Allen, G. Baldi, M. Beracochea, M. L. Bileschi, T. Burdett, J. Burgin, J. Caballero-Pérez, G. Cochrane, L. J. Colwell, T. Curtis, A. Escobar-Zepeda, T. A. Gurbich, V. Kale, A. Korobeynikov, S. Raj, A. B. Rogers, E. Sakharova, S. Sanchez, D. J. Wilkinson, R. D. Finn, MGnify: The microbiome sequence data analysis resource in 2023. *Nucleic Acids Res.* **51**, D753–D759 (2023). doi:10.1093/nar/gkac1080 Medline

38. T. H. Olsen, F. Boyles, C. M. Deane, Observed Antibody Space: A diverse database of cleaned, annotated, and translated unpaired and paired antibody sequences. *Protein Sci.* **31**, 141–146 (2022). doi:10.1002/pro.4205 Medline

39. S. K. Burley, H. M. Berman, C. Bhikadiya, C. Bi, L. Chen, L. Di Costanzo, C. Christie, K. Dalenberg, J. M. Duarte, S. Dutta, Z. Feng, S. Ghosh, D. S. Goodsell, R. K. Green, V. Guranovic, D. Guzenko, B. P. Hudson, T. Kalro, Y. Liang, R. Lowe, H. Namkoong, E. Peisach, I. Periskova, A. Prlic, C. Randle, A. Rose, P. Rose, R. Sala, M. Sekharan, C. Shao, L. Tan, Y.-P. Tao, Y. Valasatava, M. Voigt, J. Westbrook, J. Woo, H. Yang, J. Young, M. Zhuravleva, C. Zardecki, RCSB Protein Data Bank: Biological macromolecular structures enabling research and education in fundamental biology, biomedicine, biotechnology and energy. *Nucleic Acids Res.* **47**, D464–D474 (2019). doi:10.1093/nar/gky1004 Medline

40. T. Paysan-Lafosse, M. Blum, S. Chuguransky, T. Grego, B. L. Pinto, G. A. Salazar, M. L. Bileschi, P. Bork, A. Bridge, L. Colwell, J. Gough, D. H. Haft, I. Letunić, A. Marchler-Bauer, H. Mi, D. A. Natale, C. A. Orengo, A. P. Pandurangan, C. Rivoire, C. J. A. Sigrist, I. Sillitoe, N. Thanki, P. D. Thomas, S. C. E. Tosatto, C. H. Wu, A. Bateman, InterPro in 2022. *Nucleic Acids Res.* **51**, D418–D427 (2023). doi:10.1093/nar/gkac993 Medline

41. M. van Kempen, S. Kim, C. Tumescheit, M. Mirdita, J. Soding, M. Steinegger, Foldseek: fast and accurate protein structure search. bioRxiv 479398 [Preprint] (2022); https://doi.org/10.1101/2022.02.07.479398.

42. C. Hsu, R. Verkuil, J. Liu, Z. Lin, B. Hie, T. Sercu, A. Lerer, A. Rives, Learning inverse folding from millions of predicted structures. bioRxiv 487779 [Preprint] (2022); https://doi.org/10.1101/2022.04.10.487779.

43. D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano, G. Irving, Fine-tuning language models from human preferences. arXiv:1909.08593 [cs.CL] (2019).

44. L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, R. Lowe, Training language models to follow instructions with human feedback. arXiv:2203.02155 [cs.CL] (2022).

45. R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, C. Finn, Direct preference optimization: Your language model is secretly a reward model. arXiv:2305.18290 [cs.LG] (2023).

46. R. Y. Pang, W. Yuan, K. Cho, H. He, S. Sukhbaatar, J. Weston. Iterative reasoning preference optimization. arXiv:2404.19733 [cs.CL] (2024).

47. Y. A. Labas, N. G. Gurskaya, Y. G. Yanushevich, A. F. Fradkov, K. A. Lukyanov, S. A. Lukyanov, M. V. Matz, Diversity and evolution of the green fluorescent protein family. *Proc. Natl. Acad. Sci. U.S.A.* **99**, 4256–4261 (2002). doi:10.1073/pnas.062552299 Medline

48. L. Gonzalez Somermeyer, A. Fleiss, A. S. Mishin, N. G. Bozhanova, A. A. Igolkina, J. Meiler, M.-E. Alaball Pujol, E. V. Putintseva, K. S. Sarkisyan, F. A. Kondrashov, Heterogeneity of the GFP fitness landscape and data-driven protein design. *eLife* **11**, e75842 (2022). doi:10.7554/eLife.75842 Medline

49. S. Biswas, G. Kuznetsov, P. J. Ogden, N. J. Conway, R. P. Adams, G. M. Church, Toward machine-guided design of proteins. bioRxiv 337154 [Preprint](2018); https://doi.org/10.1101/337154.

50. S. Biswas, G. Khimulya, E. C. Alley, K. M. Esvelt, G. M. Church, Low-N protein engineering with data-efficient deep learning. *Nat. Methods* **18**, 389–396 (2021). doi:10.1038/s41592-021-01100-y Medline

51. M. Ormö, A. B. Cubitt, K. Kallio, L. A. Gross, R. Y. Tsien, S. J. Remington, Crystal structure of the *Aequorea victoria* green fluorescent protein. *Science* **273**, 1392–1395 (1996). doi:10.1126/science.273.5280.1392 Medline

52. K. S. Sarkisyan, D. A. Bolotin, M. V. Meer, D. R. Usmanova, A. S. Mishin, G. V. Sharonov, D. N. Ivankov, N. G. Bozhanova, M. S. Baranov, O. Soylemez, N. S. Bogatyreva, P. K. Vlasov, E. S. Egorov, M. D. Logacheva, A. S. Kondrashov, D. M. Chudakov, E. V. Putintseva, I. Z. Mamedov, D. S. Tawfik, K. A. Lukyanov, F. A. Kondrashov, Local fitness landscape of the green fluorescent protein. *Nature* **533**, 397–401 (2016). doi:10.1038/nature17995 Medline

53. D. P. Barondeau, C. D. Putnam, C. J. Kassmann, J. A. Tainer, E. D. Getzoff, Mechanism and energetics of green fluorescent protein chromophore synthesis revealed by trapped intermediate structures. *Proc. Natl. Acad. Sci. U.S.A.* **100**, 12111–12116 (2003). doi:10.1073/pnas.2133463100 Medline

54. C. Camacho, G. Coulouris, V. Avagyan, N. Ma, J. Papadopoulos, K. Bealer, T. L. Madden, BLAST+: Architecture and applications. *BMC Bioinformatics* **10**, 421 (2009). doi:10.1186/1471-2105-10-421 Medline

55. M. Steinegger, J. Söding, MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nat. Biotechnol.* **35**, 1026–1028 (2017). doi:10.1038/nbt.3988 Medline

56. J. Y. Weinstein, C. Martí-Gómez, R. Lipsh-Sokolik, S. Y. Hoch, D. Liebermann, R. Nevo, H. Weissman, E. Petrovich-Kopitman, D. Margulies, D. Ivankov, D. M. McCandlish, S. J. Fleishman, Designed active-site library reveals thousands of functional GFP variants. *Nat. Commun.* **14**, 2890 (2023). doi:10.1038/s41467-023-38099-z Medline

57. A. M. Quattrini, E. Rodríguez, B. C. Faircloth, P. F. Cowman, M. R. Brugler, G. A. Farfan, M. E. Hellberg, M. V. Kitahara, C. L. Morrison, D. A. Paz-García, J. D. Reimer, C. S. McFadden, Palaeoclimate ocean conditions shaped the evolution of corals and their skeletons through deep time. *Nat. Ecol. Evol.* **4**, 1531–1538 (2020). doi:10.1038/s41559-020-01291-1 Medline

58. J. M. Smith, Natural selection and the concept of a protein space. *Nature* **225**, 563–564 (1970). doi:10.1038/225563a0 Medline

59. U. Kamath, J. Liu, J. Whitaker, "Distributed representations" in *The Philosophy of Artificial Intelligence*, G. E. Hinton, J. L. McClelland, D. E. Rumelhart, Eds. (Springer, 1986); pp. 203–261.

60. N. Tishby, F. C. Pereira, W. Bialek, The information bottleneck method. arXiv:physics/0004057 [physics.data-an] (1999).

61. EvolutionaryScale, "Esm3 source code," Zenodo (2024); https://doi.org/10.5281/zenodo.

62. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need. arXiv:1706.03762 [cs.CL] (2017).

63. R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, T.-Y. Liu, On layer normalization in the transformer architecture. arXiv:2002.04745 [cs.LG] (2020).

64. J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, D. Hassabis, Highly accurate protein structure prediction with AlphaFold. *Nature* **596**, 583–589 (2021). doi:10.1038/s41586-021-03819-2 Medline

65. W. Kabsch, C. Sander, Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers* **22**, 2577–2637 (1983). doi:10.1002/bip.360221211 Medline

66. J. Su, Y. Lu, S. Pan, B. Wen, Y. Liu, RoFormer: Enhanced transformer with rotary position embedding. arXiv:2104.09864 [cs.CL] (2021).

67. N. Shazeer, GLU variants improve Transformer. arXiv:2002.05202 [cs.LG] (2020).

68. A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, N. Fiedel. PaLM: Scaling language modeling with Pathways. arXiv:2204.02311 [cs.CL] (2022).

69. T. Henighan, J. Kaplan, M. Katz, M. Chen, C. Hesse, J. Jackson, H. Jun, T. B. Brown, P. Dhariwal, S. Gray, B. Hallacy, B. Mann, A. Radford, A. Ramesh, N. Ryder, D. M. Ziegler, J. Schulman, D. Amodei, S. McCandlish, Scaling laws for autoregressive generative modeling. arXiv:2010.14701 [cs.LG] (2020).

70. N. Wies, Y. Levine, D. Jannai, A. Shashua. Which transformer architecture fits my data? A vocabulary bottleneck in self-attention. arXiv:2105.03928 [cs.LG] (2021).

71. J. Ingraham, V. Garg, R. Barzilay, T. Jaakkola, "Generative models for graph-based protein design." In: *Advances in Neural Information Processing Systems 32 (NeurlPS 2019)*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett, Eds. (MIT, 2019); https://proceedings.neurips.cc/paper/2019/file/f3a4ff4839c56a5f460c88cce3666a2b-Paper.pdf.

72. T. Dao, D. Y. Fu, S. Ermon, A. Rudra, C. Re, FlashAttention: Fast and memory-efficient exact attention with IO-awareness. arXiv:2205.14135 [cs.LG] (2022).

73. J. Su, C. Han, Y. Zhou, J. Shan, X. Zhou, F. Yuan, SaProt: Protein language modeling with structure-aware vocabulary. bioRxiv 560349 [Preprint] (2024); https://doi.org/10.1101/2023.10.01.560349.

74. A. van den Oord, O. Vinyals, K. Kavukcuoglu, Neural discrete representation learning. arXiv:1711.00937 [cs.LG] (2018).

75. A. Razavi, A. van den Oord, O. Vinyals, Generating diverse high-fidelity images with VQ-VAE2. arXiv:1906.00446 [cs.LG] (2019).

76. A. Roy, A. Vaswani, A. Neelakantan, N. Parmar, Theory and experiments on vector quantized autoencoders. arXiv:1805.11063 [cs.LG] (2018).

77. J. Yu, Y. Xu, J. Y. Koh, T. Luong, G. Baid, Z. Wang, V. Vasudevan, A. Ku, Y. Yang, B. K. Ayan, B. Hutchinson, W. Han, Z. Parekh, X. Li, H. Zhang, J. Baldridge, and Y. Wu. Scaling autoregressive models for content-rich text-to-image generation. arXiv:2206.10789 [cs.CV] (2022).

78. A. Bateman, M.-J. Martin, S. Orchard, M. Magrane, S. Ahmad, E. Alpi, E. H. Bowler-Barnett, R. Britto, H. Bye-A-Jee, A. Cukura, P. Denny, T. Dogan, T. G. Ebenezer, J. Fan, P. Garmiri, L. J. da Costa Gonzales, E. Hatton-Ellis, A. Hussein, A. Ignatchenko, G. Insana, R. Ishtiaq, V. Joshi, D. Jyothi, S. Kandasaamy, A. Lock, A. Luciani, M. Lugaric, J. Luo, Y. Lussi, A. MacDougall, F. Madeira, M. Mahmoudy, A. Mishra, K. Moulang, A. Nightingale, S. Pundir, G. Qi, S. Raj, P. Raposo, D. L. Rice, R. Saidi, R. Santos, E. Speretta, J. Stephenson, P. Totoo, E. Turner, N. Tyagi, P. Vasudev, K. Warner, X. Watkins, R. Zaru, H. Zellner, A. J. Bridge, L. Aimo, G. Argoud-Puy, A. H. Auchincloss, K. B. Axelsen, P. Bansal, D. Baratin, T. M. Batista Neto, M.-C. Blatter, J. T. Bolleman, E. Boutet, L. Breuza, B. C. Gil, C. Casals-Casas, K. C. Echioukh, E. Coudert, B. Cuche, E. de Castro, A. Estreicher, M. L. Famiglietti, M. Feuermann, E. Gasteiger, P. Gaudet, S. Gehant, V. Gerritsen, A. Gos, N. Gruaz, C. Hulo, N. Hyka-Nouspikel, F. Jungo, A. Kerhornou, P. Le Mercier, D. Lieberherr, P. Masson, A. Morgat, V. Muthukrishnan, S. Paesano, I. Pedruzzi, S. Pilbout, L. Pourcel, S. Poux, M. Pozzato, M. Pruess, N. Redaschi, C. Rivoire, C. J. A. Sigrist, K. Sonesson, S. Sundaram, C. H. Wu, C. N. Arighi, L. Arminski, C. Chen, Y. Chen, H. Huang, K. Laiho, P. McGarvey, D. A. Natale, K. Ross, C. R. Vinayaka, Q. Wang, Y. Wang, J. Zhang; UniProt Consortium, UniProt: The

Universal Protein Knowledgebase in 2023. *Nucleic Acids Res.* **51**, D523–D531 (2023). doi:10.1093/nar/gkac1052 Medline

79. I. A. Chen, K. Chu, K. Palaniappan, A. Ratner, J. Huang, M. Huntemann, P. Hajek, S. J. Ritter, C. Webb, D. Wu, N. J. Varghese, T. B. K. Reddy, S. Mukherjee, G. Ovchinnikova, M. Nolan, R. Seshadri, S. Roux, A. Visel, T. Woyke, E. A. Eloe-Fadrosh, N. C. Kyrpides, N. N. Ivanova, The IMG/M data management and analysis system v.7: Content updates and new features. *Nucleic Acids Res.* **51**, D723–D732 (2023). doi:10.1093/nar/gkac976 Medline

80. P. Jones, D. Binns, H.-Y. Chang, M. Fraser, W. Li, C. McAnulla, H. McWilliam, J. Maslen, A. Mitchell, G. Nuka, S. Pesseat, A. F. Quinn, A. Sangrador-Vegas, M. Scheremetjew, S.-Y. Yong, R. Lopez, S. Hunter, InterProScan 5: Genome-scale protein function classification. *Bioinformatics* **30**, 1236–1240 (2014). doi:10.1093/bioinformatics/btu031 Medline

81. P. Kunzmann, K. Hamacher, Biotite: A unifying open source computational biology framework in Python. *BMC Bioinformatics* **19**, 346 (2018). doi:10.1186/s12859-018-2367-z Medline

82. W. G. Touw, C. Baakman, J. Black, T. A. te Beek, E. Krieger, R. P. Joosten, G. Vriend, A series of PDB-related databanks for everyday needs. *Nucleic Acids Res.* **43**, D364–D368 (2015). doi:10.1093/nar/gku1028 Medline

83. I. Loshchilov, F. Hutter, Decoupled weight decay regularization. arXiv:1711.05101 [cs.LG] (2017).

84. Y. Zhao, A. Gu, R. Varma, L. Luo, C.-C. Huang, M. Xu, L. Wright, H. Shojanazeri, M. Ott, S. Shleifer, A. Desmaison, C. Balioglu, P. Damania, B. Nguyen, G. Chauhan, Y. Hao, A. Mathews, S. Li. Pytorch fsdp: Experiences on scaling fully sharded data parallel. arXiv:2304.11277 [cs.DC] (2023).

85. NVIDIA, "Transformer engine" (Github, 2024); https://github.com/NVIDIA/TransformerEngine.

86. B. Lefaudeux, F. Massa, D. Liskovich, W. Xiong, V. Caggiano, S. Naren, M. Xu, J. Hu, M. Tintore, S. Zhang, P. Labatut, D. Haziza, L. Wehrstedt, J. Reizenstein, G. Sizov, "xformers: A modular and hackable transformer modelling library" (Github, 2022); https://github.com/facebookresearch/xformers.

87. Y. Dong, J.-B. Cordonnier, A. Loukas, Attention is not all you need: Pure attention loses rank doubly exponentially with depth. arXiv:2103.03404 [cs.LG] (2021).

88. M. Dehghani, J. Djolonga, B. Mustafa, P. Padlewski, J. Heek, J. Gilmer, A. P. Steiner, M. Caron, R. Geirhos, I. Alabdulmohsin, R. Jenatton, L. Beyer, M. Tschannen, A. Arnab, X. Wang, C. Riquelme Ruiz, M. Minderer, J. Puigcerver, U. Evci, M. Kumar, S. V. Steenkiste, G. F. Elsayed, A. Mahendran, F. Yu, A. Oliver, F. Huot, J. Bastings, M. Collier, A. A. Gritsenko, V. Birodkar, C. N. Vasconcelos, Y. Tay, T. Mensink, A. Kolesnikov, F. Pavetic, D. Tran, T. Kipf, M. Lucic, X. Zhai, D. Keysers, J. J. Harmsen, N. Houlsby, Scaling vision transformers to 22 billion parameters. arXiv:2302.05442 [cs.CV] (2023).

89. M. Wortsman, P. J. Liu, L. Xiao, K. E. Everett, A. A. Alemi, B. Adlam, J. D. Co-Reyes, I. Gur, A. Kumar, R. Novak, J. Pennington, J. Sohl-Dickstein, K. Xu, J. Lee, J. Gilmer, S.

Kornblith, Small-scale proxies for large-scale transformer training instabilities. arXiv:2309.14322 [cs.LG] (2024).

90. G. Yang, E. Hu, I. Babuschkin, S. Sidor, X. Liu, D. Farhi, N. Ryder, J. Pachocki, W. Chen, J. Gao, Tensor programs V: Tuning large neural networks via zeroshot hyperparameter transfer. arXiv:2203.03466 [cs.LG] (2021).

91. G. Yang, D. Yu, C. Zhu, S. Hayou, Tensor programs VI: Feature learning in infinite depth neural networks. arXiv:2310.02244 [cs.NE] (2023).

92. J. Haas, A. Barbato, D. Behringer, G. Studer, S. Roth, M. Bertoni, K. Mostaguir, R. Gumienny, T. Schwede, Continuous Automated Model EvaluatiOn (CAMEO) complementing the critical assessment of structure prediction in CASP12. *Proteins* **86**, 387–398 (2018). doi:10.1002/prot.25431 Medline

93. A. Kryshtafovych, T. Schwede, M. Topf, K. Fidelis, J. Moult, Critical assessment of methods of protein structure prediction (CASP)-Round XIV. *Proteins* **89**, 1607–1617 (2021). doi:10.1002/prot.26237 Medline

94. A. Kryshtafovych, M. Antczak, M. Szachniuk, T. Zok, R. C. Kretsch, R. Rangan, P. Pham, R. Das, X. Robin, G. Studer, J. Durairaj, J. Eberhardt, A. Sweeney, M. Topf, T. Schwede, K. Fidelis, J. Moult, New prediction categories in CASP15. *Proteins* **91**, 1550–1557 (2023). doi:10.1002/prot.26515 Medline

95. E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685 [cs.CL] (2021).

96. L. McInnes, J. Healy, J. Melville, UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. arXiv:1802.03426 [stat.ML] (2020).

97. B. Hie, S. Candido, Z. Lin, O. Kabeli, R. Rao, N. Smetanin, T. Sercu, A. Rives, A high-level programming language for generative protein design. bioRxiv 521526 [Preprint] (2022); https://doi.org/10.1101/2022.12.21.521526.

98. N. Hulo, A. Bairoch, V. Bulliard, L. Cerutti, E. De Castro, P. S. Langendijk-Genevaux, M. Pagni, C. J. A. Sigrist, The PROSITE database. *Nucleic Acids Res.* **34**, D227–D230 (2006). doi:10.1093/nar/gkj063 Medline

99. C. Zhang, X. Zhang, P. L. Freddolino, Y. Zhang, BioLiP2: An updated structure database for biologically relevant ligand-protein interactions. *Nucleic Acids Res.* **52**, D404–D412 (2024). doi:10.1093/nar/gkad630 Medline

100. M. G. Azar, M. Rowland, B. Piot, D. Guo, D. Calandriello, M. Valko, R. Munos, A general theoretical paradigm to understand learning from human preferences. arXiv:2310.12036 [cs.AI] (2023).

101. K. Ethayarajh, W. Xu, N. Muennighoff, D. Jurafsky, D. Kiela, KTO: Model alignment as prospect theoretic optimization. arXiv:2402.01306 [cs.LG] (2024).

102. L. Gao, J. Schulman, J. Hilton, Scaling laws for reward model overoptimization. arXiv:2210.10760 [cs.LG] (2023).

103. M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M.

Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, W. Zaremba. Evaluating large language models trained on code. arXiv:2107.03374 [cs.LG] (2021).

104. F. Yang, L. G. Moss, G. N. Phillips Jr., The molecular structure of green fluorescent protein. *Nat. Biotechnol.* **14**, 1246–1251 (1996). doi:10.1038/nbt1096-1246 Medline

105. J. Ho, T. Salimans, Classifier-free diffusion guidance. arXiv:2207.12598 [cs.LG] (2022).

106. W. Kabsch, A solution for the best rotation to relate two sets of vectors. *Acta Crystallogr. A* **32**, 922–923 (1976). doi:10.1107/S0567739476001873

107. S. M. Hartley, K. A. Tiernan, G. Ahmetaj, A. Cretu, Y. Zhuang, M. Zimmer, AlphaFold2 and RoseTTAFold predict posttranslational modifications. Chromophore formation in GFP-like proteins. *PLOS ONE* **17**, e0267560 (2022). doi:10.1371/journal.pone.0267560 Medline

108. J. Salazar, D. Liang, T. Q. Nguyen, K. Kirchhoff, Masked language model scoring. arXiv:1910.14659 [cs.CL] (2019).

109. L. Somermeyer, "Orthologous gfp fitness peaks" (SoftwareHeritage, 2022); https://archive.softwareheritage.org/swh:1:cnt:a4c63cdf2f4524c8d5c813a1972a5ac649266e2b.

110. K. Katoh, D. M. Standley, MAFFT multiple sequence alignment software version 7: Improvements in performance and usability. *Mol. Biol. Evol.* **30**, 772–780 (2013). doi:10.1093/molbev/mst010 Medline

111. T. J. Lambert, FPbase: A community-editable fluorescent protein database. *Nat. Methods* **16**, 277–278 (2019). doi:10.1038/s41592-019-0352-8 Medline

112. S. Seabold, J. Perktold, "Statsmodels: Econometric and statistical modeling with Python," in *Proceedings of the 9th Python in Science Conference (SciPy 2010), Austin, Texas June 28 – July 3* (SciPy, 2010); https://proceedings.scipy.org/articles/Majora-92bf1922-011.

113. Responsible AI × Biodesign, "Community values, guiding principles, and commitments for the responsible development of AI for protein design" (2024); https://responsiblebiodesign.ai/.

114. US Centers for Disease Control and Prevention, "Select agents and toxins list" (CDC, 2024); https://www.selectagents.gov/sat/list.htm.

115. US Department of Human Health Services, "Screening framework guidance for providers and users of synthetic nucleic acids" (USDHHS, technical report, 2023); https://aspr.hhs.gov/legal/synna/Documents/SynNA-Guidance-2023.pdf.

116. P. Notin, A. W. Kollasch, D. Ritter, L. van Niekerk, S. Paul, H. Spinner, N. Rollins, A. Shaw, R. Weitzman, J. Frazer, M. Dias, D. Franceschi, R. Orenbuch, Y. Gal, D. S. Marks, ProteinGym: Large-scale benchmarks for protein design and fitness prediction. bioRxiv 570727 [Preprint] (2023); https://doi.org/10.1101/2023.12.07.570727.

117. T. A. Hopf, J. B. Ingraham, F. J. Poelwijk, C. P. Schärfe, M. Springer, C. Sander, D. S. Marks, Mutation effects predicted from sequence co-variation. *Nat. Biotechnol.* **35**, 128–135 (2017). doi:10.1038/nbt.3769 Medline