Supplementary Information for

Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences

Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, and Rob Fergus

[§]To whom correspondence should be addressed. E-mail: arives@cs.nyu.edu.

This PDF file includes:

Supporting Information Text

A. Approach & Data

1. Pre-training datasets.

UniParc pre-training dataset. A series of development models are trained on UniParc (1) Release 2019_01 which contains approximately 250M sequences. 1M sequences are held-out randomly for validation. These models were used in the preprint of this paper, and representations from the models are used in Figures 1, 2, and 3.

UniRef pre-training datasets. Datasets are based on UniRef (2) dated March 28, 2018 to permit a temporal hold-out with CASP13. 10% of UniRef50 clusters are randomly selected as a held-out evaluation set, yielding 3.02 million representative sequences for evaluation. Three training datasets are used, removing all sequences belonging to clusters selected for the evaluation set: (i) UR100, 124.9M UniRef100 representative sequences; (ii) UR50/S, 27.1M UniRef50 representative sequences; (iii) UR50/D, 124.9M UniRef50 cluster members sampled evenly by cluster. To ensure a deterministic validation set, we removed sequences longer than 1024 amino acids from the validation set.

2. Downstream tasks.

Remote Homology. A dataset of remote homolog pairs is derived from SCOPe (3) containing 256,806 pairs of remote homologs at the fold level and 92,944 at the superfamily level, consisting of 217 unique folds and 366 unique superfamilies. Creation of the dataset is detailed below in the section on remote homology.

Linear projections. Five-fold cross validation datasets implementing structural hold-outs at the family, superfamily, and fold level are constructed using SCOPe (3). Independently for each level of structural hold-out, the domains are split into 5 equal sets, i.e. five sets of folds, superfamilies, or families. This ensures that for each of the five partitions, structures having the same classification do not appear in both the train and test sets. For a given classification level each structure appears in a test set once, so that in the cross validation experiment each of the structures will be evaluated exactly once. Scores reported are the mean and standard deviation over each of the five test sets. Further details on construction of the dataset are given below in the section on linear projections.

Secondary structure prediction. All downstream models are trained using the Netsurf (4) training dataset containing 10,837 examples with labels and HMM profiles. Netsurf features are replicated for CASP13 domains using MMseqs2 (5) on the Uniclust90 (6) dataset released April 2017. For test sets we use (i) the standard CB513 (7) test set of 513 sequences with sequence identity hold-out at 25% identity; and (ii) the 34 publicly available CASP domains, using DSSP (8) to label secondary structure, with temporal hold-out for both pre-training and downstream data.

Contact prediction. All downstream models are trained using the training and test sets of Wang et al. (9). Comparisons with RaptorX features use features from Wang et al. (9) and Xu (10). The following test sets are used: (i) RaptorX Test, 500 domains (25% sequence identity hold-out); (ii) CASP11, 105 domains (25% sequence identity hold-out); (iii) CASP12, 55 domains (temporal hold-out from training data but not pre-training data); (iv) CASP13, 34 publicly released domains (temporal hold-out from training data and pre-training data). The training set consists of 6,767 sequences with contact map targets, a subset of PDB created in February 2015 (9). The use of an earlier version of the PDB ensures a temporal hold-out w.r.t. both CASP12 and CASP13. Additionally, Wang et al. (9) implemented a sequence identity hold-out for Test and CASP11 by removing proteins from the training set which share >25% sequence identity or have BLAST E-value <0.1 with the proteins in these test sets.

Mutational effect prediction. The model is fine-tuned on deep mutational scanning datasets compiled by Gray et al. (11) and Riesselman et al. (12).

3. The Transformer. We use a deep Transformer encoder model (13, 14), processing input as character sequences of amino acids. In contrast to recurrent and convolutional neural networks, the Transformer makes no assumptions on the ordering of the input and instead uses position embeddings. Particularly relevant to protein sequences is the Transformer's ability to model long range dependencies, which are not effectively captured by RNNs or LSTMs (15). A key factor affecting the performance of LSTMs on these tasks is the path lengths that must be traversed by forward activation and backward gradient signals in the network (16).

It is well known that structural properties of protein sequences are reflected in long-range dependencies. Direct coupling analysis (17–19) which aims to detect pairwise dependencies in multiple sequence alignments uses a Markov Random Field (Potts Model) which models the complete sequence with pairwise coupling parameters. Similarly, the Transformer builds up a representation of a sequence by alternating self-attention with non-linear projections. Self-attention structures computation so that each position is represented by a weighted sum of the other positions in the sequence. The attention weights are computed dynamically and allow each position to choose what information from the rest of the sequence to integrate at every computation step.

Developed to model large contexts and long range dependencies in language data, self-attention architectures currently give state-of-the-art performance on various natural language tasks, mostly due to the Transformer's scalability in parameters and the amount of context it can integrate (14). The tasks include token-level tasks like part-of-speech tagging, sentence-level tasks such as textual entailment, and paragraph-level tasks like question-answering.

Scaled dot-product attention. Self-attention takes a sequence of vectors $(h_1, \ldots, h_n)$ and produces a sequence of vectors $(h'_1, \ldots, h'_n)$ by computing interactions between all elements in the sequence. The Transformer model uses scaled dot-product attention (13):

$$A(h) = \mathrm{softmax}(1/\sqrt{d}\, Q(h)K(h)^T)V(h) \quad [1]$$

Here the query Q, key K, and value V, are projections of the input sequence to $n \times d$ matrices where n is the length of the sequence and d is the inner dimension of the matrix outer product between Q and K. This outer product parameterizes an $n \times n$ map of attention logits, which are rescaled, and passed through the softmax function row-wise, thereby representing each position of the sequence in the output as a convex combination of the sequence of values V. One step of self-attention directly models possible pairwise interactions between all positions in the sequence simultaneously. Note the contrast to recurrent and convolutional models which can only represent long-range context through many steps, and the parallel in inductive bias with the explicit pairwise parameterization of Markov Random Fields in widespread use for modeling protein MSAs.

Multi-headed self-attention concatenates the output of t independent attention heads:

$$A_{\mathrm{MH}}(x) = A_1(x) \ldots A_t(x) \quad [2]$$

Use of multiple heads enables representation of different inter-position interaction patterns.

Architecture The Transformer models (13) in this work take a sequence of tokens $(x_1, \ldots, x_n)$ and output a sequence of log probabilities $(y_1, \ldots, y_n)$ which are optimized using the masked language modeling objective. The computation proceeds through a series of residual blocks producing hidden states, each a sequence of vectors $(h_1, \ldots, h_n)$ with embedding dimension d.

The Transformer model architecture consists of a series of encoder blocks interleaving two functions: a multi-headed self-attention computing position-position interactions across the sequence, and a feed-forward network applied independently at each position.

The attention unit:

$$U_{\mathrm{AT}}(h) = P(A_{\mathrm{MH}}(n(h))) \quad [3]$$

Applies one step of multi-headed scaled dot-product attention to the normalized input, denoted by $n(x)$, projecting the result into the residual path.

The feed-forward network (with the output state of P_1 defining the "MLP dimension"):

$$U_{\mathrm{FF}}(h) = P_2(g(P_1(n(h)))) \quad [4]$$

Passes the normalized input through a position-independent multi-layered perceptron (MLP) with activation function $g(x)$.

The full Transformer block:

$$\begin{aligned} &B(h): \\ &\quad h \leftarrow h + U_{\mathrm{AT}}(h) \\ &\quad h \leftarrow h + U_{\mathrm{FF}}(h) \end{aligned}$$

Successively applies the self-attention unit, and the feed-forward network on a residual path.

The Transformer model:

$$\begin{aligned} &\mathrm{Transformer}(x): \\ &\quad h \leftarrow E(x) + H(x) \\ &\quad h \leftarrow B_k(h) \ \text{ for } k \in 1 \ldots K \\ &\quad y \leftarrow W(h) \end{aligned}$$

Consists of an embedding step with token $E(x)$ and positional $H(x)$ embeddings, followed by K layers of Transformer blocks, before a projection W to log probabilities. The raw input sequence is represented as a sequence of 1-hot vectors of dimension 25, which is passed through $E(x)$ the learned embedding layer before being presented to the first Transformer layer.

The models trained in the paper use pre-activation blocks (20), where the layer normalization (21) is applied prior to the activation as in Radford et al. (22), enabling stable training of deep Transformer networks. No dropout is used. All projections include biases, except for the token and positional embeddings. We use learned token embeddings, and harmonic positional embeddings as in (13). The feed-forward network uses the Gaussian error linear unit (23) activation function. We initialize all layers from a zero centered normal distribution with standard deviation 0.02, and re-scale the initialization of the projections into the residual path by $1/\sqrt{L}$ where L is the number of residual layers. All biases are initialized to zero. The query, key, and value projections are to d dimensions, and the hidden dimension of the feed-forward network is $4d$.

4. Pre-trained Transformer Models.

# Layers	# Heads	Embedding Dim	MLP Dim	# Params	Steps
12	12	768	3072	85.1M	1.6M
24	12	768	3072	170.2M	300k
36	20	1280	5120	708.6M	300k

Table A.1. Hyperparameters for development Transformer models trained on UniParc. Embedding dim is the dimension of the hidden states at the output of each transformer block. MLP Dim refers to the width of hidden layer P_1 in the Transformer's MLPs.

# Layers	# Heads	Embedding Dim	MLP Dim	# Params	Data	Steps
6	12	768	3072	42.6M	UR50/S	840K
12	12	768	3072	85.1M	UR50/S	840K
34	20	1280	5120	669.2M	UR100	275K
34	20	1280	5120	669.2M	UR50/S	840K
34	20	1280	5120	669.2M	UR50/D	906K

Table A.2. Hyperparameters for UniRef Transformer models. Note: UR100 model stopped making significant progress on valid loss and was stopped at 275K updates.

UniParc development models We experimented with Transformer models of various depths, including a 36-layer Transformer with 708.6 million parameters, and a 12-layer model with 85.1M parameters trained. Development models were trained on UniParc. Details are in Table A.1.

UniRef models We train 34-layer models with 669.2M parameters across different datasets and fractions of training data. Additionally we train 6 and 12-layer models. These models are detailed in Table A.2.

ESM-1b The ESM-1b hyperparameter sweep and model is described in detail in Appendix B. In brief, ESM-1b is the result of an extensive hyperparameter sweep that was performed on smaller 12-layer models. ESM-1b is the result of scaling up that model to 33 layers. Compared to the Uniref models, the main changes in ESM-1b are: higher learning rate; dropout after word embedding; learned positional embeddings; final layer norm before the output; and tied input/output word embedding.

Pre-training task The masked language modeling pre-training task follows Devlin et al. (14). Specifically, we select as supervision 15% of tokens randomly sampled from the sequence. For those 15% of tokens, we change the input token to a special "masking" token with 80% probability, a randomly-chosen alternate amino acid token with 10% probability, and the original input token (i.e. no change) with 10% probability. We take the loss to be the whole batch average cross entropy loss between the model's predictions and the true token for these 15% of amino acid tokens. In contrast to Devlin et al. (14), we do not use any additional auxiliary prediction losses. The ESM-1b models, as well as the UniParc development models used in visualizations and in the supplemental results are trained with the masking procedure above. The UniRef models used across the experiments of the main text are trained similarly, except that for the 15% of tokens selected as prediction targets, all are replaced by the mask token.

Pre-training details Our model was pre-trained using a context size of 1024 tokens. As most Uniparc sequences (96.7%) contain fewer than 1024 amino acids, the Transformer is able to model the entire context in a single model pass. For those sequences that are longer than 1024 tokens, we sampled a random crop of 1024 tokens during each training epoch. The model was optimized using Adam ($\beta_1 = 0.9, \beta_2 = 0.999$) with learning rate 10^{-4}. We trained with 131,072 tokens per batch (128 gpus x 1024 tokens). The models follow a warm-up period of 16000 updates, during which the learning rate increases linearly. Afterwards, the learning rate follows an inverse square root decay schedule. All models were trained using the fairseq toolkit (24) on 128 NVIDIA V100 GPUs.

5. Evaluating the models for downstream tasks. After pre-training the model with unsupervised learning, we can adapt the parameters to supervised tasks. By passing the input sequence $(x_1, \ldots, x_n)$ through our pre-trained model, we obtain a final vector representation of the input sequence $(h_1, \ldots, h_n)$. During pre-training, this representation is projected to log probabilities $(y_1, \ldots, y_n)$. Recall that a softmax over y_i represents the model's posterior for the amino acid at position i. These final representations $(h_1, \ldots, h_n)$ are used directly, or fine-tuned in a task-dependent way by adding additional layers to the model and allowing the gradients to backpropagate through the weights of the pre-trained model to adapt them to the new task. Hidden representations from intermediate layers rather than the final layer can also be used.

6. Language Modeling Baselines. In addition to comparing to past work, we also implemented deep learning baselines for our experiments.

Frequency (n-gram) models To establish a meaningful performance baseline on the sequence modeling task (see Materials and Methods), we construct n-gram frequency-based models for context sizes $1 \leq n \leq 10^4$, applying optimal Laplace smoothing for each context size. The Laplace smoothing hyperparameter in each case was tuned on the validation set. ECE is reported for the best left-conditioning n-gram model.

Bidirectional LSTM language models We trained state-of-the-art LSTM (25) language models on the UR50 dataset. We use the ELMo model of Peters et al. (26) which concatenates two independent autoregressive language models with left-to-right and right-to-left factorization. Unlike standard LSTM language models, the ELMo model receives context in both directions and is therefore comparable to the Transformers we train that also use the whole context of the sequence. We train two models: (i) the small model has approximately 28.4M parameters across 3 layers, with an embedding dimension of 512 and a hidden dimension of 1024; (ii) the large model has approximately 113.4M parameters across 3 layers, with an embedding dimension of 512 and a hidden dimension of 4096. The models are trained with a nominal batch size of 32,768, with truncated backpropagation to 100 tokens, dropout of 0.1, learning rate of 8e-4, using the Adam optimizer with betas of (0.9, 0.999), clip norm 0.1 and warmup of 1500 updates using an inverse square root learning rate schedule. We searched across a range of learning rates and found 8e-4 to be optimal.

7. Metric structure experiments.

Dataset An orthologous group dataset was constructed from eggNOG 5.0 (27) by selecting 25 COG orthologous groups toward maximizing the size of the intersected set of species within each orthologous group. Through a greedy algorithm, we selected 25 COG groups with an intersecting set of 2,609 species. We shrank the dataset above by selecting only one species from each of 24 phyla in order to ensure species-level diversity.

8. Remote Homology.

Dataset We used the database of SCOP 2.07e filtered to 40% sequence similarity, provided by the ASTRAL compendium (3). Following standard practices (28), we exclude folds that are known to be related, specifically Rossman-like folds (c.2-c.5, c.27 and 28, c.30 and 31) and four- to eight-bladed β-propellers (b.66-b.70). This yields 256,806 pairs of remote homologs at the fold level and 92,944 at the superfamily level, consisting of 217 unique folds and 366 unique superfamilies. We then perform an 80-20 split, and tune our hyperparameters on the "training set" and report results on the held out 20% of the data.

Metrics Given a protein sequence x, with final hidden representation $(h_1, \ldots, h_n)$, we define the embedding of the sequence to be a vector e which is the average of the hidden representations across the positions in the sequence:

$$e = \frac{1}{n} \sum_{i=1}^{n} h_i$$

We can compare the similarity of two protein sequences, x and x' having embeddings e and e' using a metric in the embedding space.

We evaluate the L2 distance $\|e - e'\|_2$ and the cosine distance $e \cdot e' / \|e\| \|e'\|$. Additionally we evaluated the L2 distance after projecting the e vectors to the unit sphere.

Evaluation To evaluate HHblits (29), first we construct HMM profiles for each sequence using default parameters for 'hhblits', except we use 3 iterations. Then, we do an all-to-all alignment using 'hhalign' with default parameters, and use the resulting E-value as a measure of similarity. Given a query sequence, a sequence is more similar with a smaller E-value.

The two metrics reported are Hit-10 as introduced in Ma et al. (30) and AUC. For both metrics, for each sequence, we treat it as a query and we rank each other sequence according to the distance metric used. Following Ma et al. (30), for evaluation at the fold level, any domain with the same fold is a positive; any domain with a different fold is a negative; and domains belonging to the same superfamily are excluded. For evaluation at the superfamily level, any domain with the same superfamily is a positive; any domain with a different superfamily is a negative; and domains belonging to the same family are excluded. This ensures we specifically measure how well our models do on finding *remote* homologs.

For Hit-10, we consider the query a success if any of the top 10 sequences is a remote homolog. We report the proportion of successes averaged across all queries. For AUC, we first compute the AUC under the ROC curve when classifying sequences by vector similarity to the query. Then, we average the AUC across all query sequences.

We found that cosine similarity results in the best Hit-10 scores, while the L2 with unnormalized vectors result in the best AUC scores, so we report this in Table 2.

Implementation We used the FAISS similarity search engine (31).

9. Representational similarity-based alignment of sequences within MSA families.

Family selection We use the Pfam database (32). We first filtered out any families whose longest sequence is less than 32 residues or greater than 1024 residues in length. We then ranked the families by the number of sequences contained in each family and selected the 128 largest families and associated MSAs. Finally, we reduced the size of each family to 128 sequences by uniform random sampling.

Aligned pair distribution For each family, we construct an empirical distribution of aligned residue pairs by enumerating all pairs of positions and indices that are aligned within the MSA and uniformly sampling 50,000 pairs.

Unaligned pair distribution We also construct for each family a background empirical distribution of unaligned residue pairs. This background distribution needs to control for within-sequence position, since the residues of two sequences that have been aligned in an MSA are likely to occupy similar positions within their respective unaligned source sequences. Without controlling for this bias, a difference in the distributions of aligned and unaligned pairs could arise from representations encoding positional information rather than actual context. We control for this effect by sampling from the unaligned-pair distribution in proportion to the observed positional differences from the aligned-pair distribution. Specifically, the following process is repeated for each pair in the empirical aligned distribution:

1. Calculate the absolute value of the difference of each residue's within-sequence positions in the aligned pair.
2. Select a pair of sequences at random.
3. For that pair of sequences, select a pair of residues at random whose absolute value of positional difference equals the one calculated above.
4. Verify that the residues are unaligned in the MSA; if so, add the pair to the empirical background distribution.
5. Otherwise, return to step 2.

This procedure suffices to compute a empirical background distribution of 50,000 unaligned residue pairs.

Similarity distributions Finally, for each family and each distribution, we apply the cosine similarity operator to each pair of residues to obtain the per-family aligned and unaligned distribution of representational cosine similarities.

10. Linear projections.

Dataset We construct five-fold cross validation datasets with structural hold-outs at the family, superfamily, and fold level using SCOPe 2.07 (3). We use the full version of SCOPe 2.07, clustered at 90% sequence identity, generated on January 23, 2020, and extract the domain annotations with labels. There are 19,695 domains. Then, independently for each hold-out level, we split the domains at the hold-out level into 5 equal sets, i.e. five sets of folds, superfamilies, or families. This ensures that for each partition, structures having the same classification do not appear in both the train and test sets. For a given classification level each structure appears in a test set once, so that in the cross validation experiment each of the structures will be evaluated once. Scores reported are the mean and standard deviation over each of the five test sets.

For each domain, we first obtain the sequence x whose residues align with the domain specification. To construct the secondary structure labels, we take each CIF file (pulled from PDB), and run DSSP (8, 33). If x has any residues where DSSP has not provided a secondary structure label, we mark them as missing data and do not supervise for those positions.

To construct the contact map, we obtain C_β coordinates from the structure portion of the CIF file (C_α in the case of glycine), defaulting to NaN where information is missing, and finally calculating pairwise distances and thresholding at 8Å. Similar to secondary structure, we do not supervise over NaNs.

We discard any domains where (1) DSSP fails, (2) we are unable to align the sequence to the structure, or (3) the domain is longer than 1023 residues. This leaves 15,297 domains.

For the MSA baselines, we query each sequence against the Uniclust30, 2017 database (6) with HHblits (29) using the default settings with additional parameters (n=3, 1e-3). For secondary structure prediction, we construct HMM profiles using HHmake (default settings). For contact prediction, we apply CCMpred (34) implementation of pseudolikelihood maximization (35, 36) using default settings with (n=500) to each MSA, from which we extract both an output matrix ($\mathbb{R}^{L\times L}$), as well as a sequence profile ($\mathbb{R}^{L\times K}$) where L is the length of the sequence and K is the size of the amino acid vocab, i.e. 25.

Representations To obtain sequence representations, we provide the sequence of the domain as input to a forward pass of the Transformer model. We retrieve the activations into the final multi-head attention (after the layer normalization), using this as the matrix of sequence representations ($\mathbb{R}^{L\times d}$) where d is the hidden dimension of the model.

Secondary structure projections For secondary structure, we fit a multi-class logistic regression taking as input an individual representation h_i and as output the secondary structure label from DSSP. We observe that the logistic regression model's performance does not change with penalty settings (L1, L2, no penalty); therefore we report the result where the L2 penalty is applied during training.

Contact projections For contacts, we fit two linear projections. Given two representations h_i, h_j at positions i and j, we regress to whether residues at positions i and j are in contact:

$$P(\text{i,j contact}) = \text{sigmoid}[(Ph_i + p) \cdot (Qh_j + q) + b]$$

With learned projections P, Q, vector biases p, q, and scalar bias b. We use the AdamW optimizer (37) to fit the projections and bias term.

For each partition we set aside approximately 12.5% of the training set as validation. We sweep over a range of projection dimensions (32 to 512), learning rates (1e-6 to 1e-2) and weight decay values (0 to 0.9). Based on the best validation Top-L, long-range precision score, we set the projection dimension to 128, the learning rate to 1e-4, and the optimizer weight decay to 0.2. We observe that the precision score does not improve with an increased projection dimension over 128.

The above setup for contact projections only applies to the Transformer models and the sequence profile baseline. No supervision is applied to the CCMpred output.

11. Single-family data and analysis. For each of the three domains used, we extracted all domain sequences from the Pfam dataset (32) and located the subset of PDB files containing the domain, using the latter to derive ground truth secondary structure labels (8).

Pre-training is with the masked language modeling objective, using the same hyperparameters as used to train the UniParc development models. The domain sequences were randomly partitioned into training, validation, and testing datasets. For each family, the training dataset comprises 65,536 sequences, the validation dataset comprises either 16,384 sequences (PF00005 and PF00069) or 8,192 sequences (PF00072), and the test dataset comprises the remainder.

Each Pfam family also forms an evaluation dataset for linear projection; from the sequences with corresponding crystal structures, the training dataset comprises 80 sequences and the test dataset comprises the remainder.

12. Secondary structure prediction. Deep neural networks and feature combination. We use features from the final hidden representations of the models. We removed the final embedding layer, added layer norm, and applied a top-level architecture following (4). In particular, this top-level architecture consists of two parallel convolution layers and an identity layer, whose outputs are concatenated in the feature dimension and fed to a two layer bidirectional LSTM containing 1024 hidden units and dropout $p = 0.5$. The output is then projected to an 8-dimensional feature vector at each position and the model is trained with a categorical cross-entropy loss with the Q8 labels. The training data was obtained from the (4). Secondary structure labels for the CASP13 test set were constructed using DSSP.

In feature combination experiments, we used the features provided by (4) which were generated using MMseqs2 on the Uniclust90 dataset released April 2017. For CASP13 experiments, we generated these features using code provided by (4) on CASP13 domains.

As a baseline, we reimplemented (4) by replacing the Transformer features with the MMseqs2 features and keeping the top-level architecture. For feature combination experiments, we projected (a) the features from this baseline and (b) the features from the Transformer to the same dimension (512 units), concatenated along the feature dimension, and fed the resulting tensor to a two layer bidirectional LSTM with 512 hidden units and dropout $p = 0.3$.

To check our dataset construction, we used the pretrained weights provided by (4) and evaluated their model directly in our evaluation pipeline. We were able to reproduce the values reported in (4).

13. Contact prediction. Deep neural networks and feature combination.

Data We use the datasets and features distributed with Wang et al. (9) and Xu (10). The base features are those used by RaptorX (10) a state-of-the-art method in CASP13, including sequence features, PSSM, 3-state secondary structure prediction, predicted accessibility, one-hot embedding of sequence, and pairwise features APC-corrected Potts model couplings, mutual information, pairwise contact potential.

We use the training, standard test set, and CASP11 test set from Wang et al. (9). We use the CASP12 test set from Xu (10). For the CASP13 test set we use the 34 publicly released domains.

Wang et al. (9) established training and test sets as follows. The train (6,367 proteins), valid (400 proteins) and test (500 proteins) datasets were selected as subsets of PDB25 (each protein having <25% sequence similarity). Proteins having sequence similarity >25% or BLAST E-value <0.1 with any test or CASP11 protein were excluded from training data.

All our MSAs (used for the avg and cov combination methods) are constructed by running HHblits (29) with 3 iterations and E-value 0.001 against Uniprot20 released on 2016-02; except for CASP12 and CASP13 where we used the four different MSAs released with and described in Xu (10). Note that for the Transformer pre-training UniRef50 from 2018-03 was used; hence no data which was not already available prior to the start of CASP13 was present during either pre-training or contact prediction training.

Model architecture On top of the sequence and pairwise features we use a depth-32 residual network (ResNet) model to predict binary contacts. The ResNet model architecture is similar to Wang et al. (9) and Xu (10).

The first component of the ResNet is a learned sequence pipeline $y = f_\theta^S(x)$ which maps sequence features $x \in \mathbb{R}^{L \times d_1}$ to $y \in \mathbb{R}^{L \times d_2}$ with L the length of the protein. Though f_θ^S could be a 1D convolutional network or residual network as in Wang et al. (9), we chose our sequence net to be a simple linear projection from input dimension d_1 to $d_2 = 128$ dimensions. The input dimension d_1 is either 46 (RaptorX only), 1280 (Transformer hidden state), or 1326 (feature combination). We varied d_2 and empirically determined 128 to be optimal.

The 128-D output y of the sequence net gets converted to pairwise matrix features z_1 with 256 feature maps, by an outer concatenation operation; i.e. at position i, j we concatenate y_i and y_j along the feature dimension, giving rise to $2 \times d_2$ feature maps. This $z_1 \in \mathbb{R}^{2d_2 \times L \times L}$ is then concatenated in the first (feature map or channel) dimension, with the pairwise features $z_2 \in \mathbb{R}^{6 \times L \times L}$ i.e. the pairwise RaptorX features described in previous subsection and/or the msa embedding covariance features ($z_3 \in \mathbb{R}^{256 \times L \times L}$) described in the next subsection. As such the concatenated $z \in \mathbb{R}^{262 \times L \times L}$ or $z \in \mathbb{R}^{518 \times L \times L}$.

The final component is the actual 2D residual network operating in z, which computes the binary contact probability $\hat{p} = g_\theta^R(z)$ with $\hat{p} \in \mathbb{R}^{L \times L}$ and $\hat{p}_{ij}$ the continuous predicted probability of position i and j of the protein being in contact. The

ResNet has an initial 1×1 convolutional layer going to $d_3 = 128$ feature maps, followed by MaxOut over the feature maps with stride 2, reducing to 64 feature maps. After this, there are 32 residual blocks. Each residual block has on its weight path consecutively BatchNorm - ReLU - Conv 3×3 (64 feature maps) - Dropout (0.3) - ReLU - Conv 3×3 (64 feature maps). The residual blocks have consecutive dilation rates of 1,2,4. This follows Adhikari (38). The final output is computed with a Conv 3×3 (1 output feature map) and sigmoid to produce probability of contact $\hat{p}_{ij} \in [0, 1]$. As such there are 66 convolutional layers in the main 2D ResNet.

Note that a number of shortcomings exist from our pipeline to CASP13 winners (10, 39); most importantly we use an earlier training dataset of PDB structures compiled from PDB dated Feb 2015 by Wang et al. (9), additionally we do not incorporate more recent developments like distance distribution prediction, sliding window on small crops allowing deeper ResNets, auxiliary losses like torsion angles, or data augmentation.

For reference, the officially released AlphaFold (39) predictions achieve a top-L/5,LR and top-L,LR precision on the same subset of CASP-13 targets of 75.2% and 52.2% respectively. The discrepancies in the pipeline explain why our best precisions using RaptorX features are about 7-9% lower (compare CASP13-AVG (a): 68.0% / 43.4%)

MSA Embedding feature combination. We construct features based on the embedding of the MSA of a protein sequence in our training data. We denote the original protein in our labeled dataset, i.e. query sequence x of length L, to have corresponding embedding $h = \text{Transformer}(x) \in \mathbb{R}^{L \times d}$, and the embedding of the i-th position to be $h_i \in \mathbb{R}^d$. Typically h is the last hidden state from the pre-trained Transformer model. The mth sequence in the MSA is x^m, with corresponding embedding h^m. $m \in [0, M[$ with M the MSA depth. The embeddings are computed by embedding the original sequence x^m without inserting gaps (there is no gap character in our vocabulary), then realigning the embedding according to the alignment between x^m and query sequence x by inserting 0-vectors at position i if the x_i^m is the gap character; ie $h_{i,k}^m = 0$. We also use indicator variable $\alpha_i^m = 1$ if x_i^m is non-gap (match state), or $\alpha_i^m = 0$ if x_i^m is gap. We further compute sequence weights w^m as the commonly used debiasing heuristic to reduce the influence of the oversampling of many similar sequences. The weights are defined in the usual way with 70% sequence similarity threshold: sequence weight $w^m = 1/|\{x^{m'} | seqid(x^m, x^{m'}) > 70\%\}|$ which is the inverse of the number of sequences $x^{m'}$ that are more than 70% similar to the sequence x^m i.e. hamming distance less than 0.3L.

Now we introduce the average embedding over an MSA:

$$h_{ik}^{\text{avg}} = \frac{1}{M_{\text{eff}}(i)} \sum_{m=0}^{M-1} w^m \alpha_i^m h_{i,k}^m$$

with per-position denominator $M_{\text{eff}}(i) = \sum_{m=0}^{M-1} w^m \alpha_i^m$ This is effectively a weighted average over the sequence embeddings in the MSA. Note that if the embeddings were one-hot encodings of AA identities, we would recover the position probability matrix (except the absence of a pseudocount).

Similarly; we introduce the (uncentered) covariance of the embeddings, with PCA-projected $\bar{h}$:

$$C_{ijkl} = \frac{1}{M_{\text{eff}}(i,j)} \sum_{m=0}^{M-1} w^m \alpha_i^m \bar{h}_{i,k}^m \alpha_j^m \bar{h}_{j,l}^m$$

With pairwise position denominator $M_{\text{eff}}(i,j) = \sum_{m=0}^{M-1} w^m \alpha_i^m \alpha_j^m$.

Note that to make above covariance embedding feasible, we first reduce the dimensionality of the embeddings by projecting onto the first 16 PCA directions: $\bar{h}_i = P h_i$ with $P \in \mathbb{R}^{16 \times d}$, giving rise to a covariance per pair of positions i, j and pair of interacting PCA components k, l of $L \times L \times 16 \times 16$. The 256 different k, l pairs of $C_{ijkl} \in \mathbb{R}^{L \times L \times 16 \times 16}$ will now become the feature maps of $z \in \mathbb{R}^{256 \times L \times L}$, such that $z_{16k+l,i,j} = C_{ijkl}$. We tried training (rather than fixed PCA) the projection of the features $h \to \bar{h}$ before covariance (learned linear projection P or training a 3-layer MLP). We also varied the formulation to center the embeddings over the MSA (normal covariance) and to rescale the feature maps with a pre-computed mean and standard deviation for each feature map corresponding to a pair of k, l. We found no gains from these variations over the current formulation. Note that centering with the average h^{avg} as in normal empirical covariance calculation, introduces a shift that is independent per protein (because specific to the MSA), and independent per position. Therefore it is not unexpected that the uncentered covariance gives better (more consistent) features.

14. Mutational Effect.

Datasets We used two datasets of variant effect measurements compiled by Gray et al. (11) and Riesselman et al. (12). The first dataset is a collection of 21,026 measurements from nine experimental deep mutational scans. The second dataset contains 712,218 mutations across 42 deep mutational scans.

Fine-tuning procedure To fine-tune the model to predict the effect of changing a single amino acid or combination of amino acids we regress the scaled mutational effect with:

$$y = \sum_i \log p_i(\text{mt}(i)) - \log p_i(\text{wt}(i))$$

Where $\text{mt}(i)$ is the mutated amino acid at position i, and $\text{wt}(i)$ is the wildtype amino acid. The sum runs over the indices of the mutated positions. As an evaluation metric, we report the Spearman ρ between the model's predictions and experimentally measured values.

15. Area under the ROC curve. For a binary classification task, the ROC curve plots the true positive rate against the false positive rate at various classification thresholds. The area under the ROC curve gives a measure that quantifies the model's ability to distinguish between classes. Intuitively a perfect classifier has an AUC of 1, while a uniform random classifier has an AUC of 0.5.

B. ESM-1b Hyperparameter Optimization

Experimental setup We perform a systematic analysis on Transformer models with 100M parameters. We train models on the UniRef50 dataset following the same methodology described in the rest of this work.

After identifying the best performing settings on 100M parameter models, we explore scale by training 650M parameter models. All models are trained with a target batch size of 1M tokens. To accommodate the large batch size, we use gradient accumulation and distributed data parallel. Under this setup, each epoch of a 100M parameter model completes in 1.8 hours on 32 GPUs. Each epoch of a 650M parameter model completes in 8.5 hours on 64 GPUs.

When studying architectural variants, we assess the quality of representations from each model after 10 or 12 epochs of pre-training. We observed that relative performance ranking of the models does not change after this point. Notably, this is still early in training; our best performing model, ESM-1b, is trained for 56 epochs.

Hyperparameter: Masking and data setup Protein language models are trained with a masked language modeling objective, wherein each input sequence is corrupted by replacing a fraction of the tokens with a special mask token. We train 100M parameter models for 10 epochs and compare their performance on the CB513 test set. We investigate four masking strategies:

- *All masks*: following supplemental section 4, 15% of the input tokens are replaced with a mask token and predicted.
- *All random (uniform)*: 15% of the input tokens are replaced with an amino acid selected uniform randomly and predicted.
- *All random (frequency)*: 15% of the input tokens are replaced with an amino acid selected according to their frequency in the dataset and predicted.
- *BERT*: 15% of the input tokens are selected and predicted. Of these, 80% are replaced with mask token; 10% with a uniform random amino acid; 10% not changed.

In all cases, we follow Liu et al. (40) and dynamically mask the sequences, such that a new mask is randomly selected at each epoch. Since the input data changes across runs, language modeling perplexities cannot be fairly compared. Therefore, we evaluate the downstream performance of the models on a secondary structure benchmark. Table B.1 finds that the *BERT* masking pattern performs better than the other masking patterns and is therefore used for all model variations below.

Masking pattern	CB513
All masks	60.4
All random (uniform)	59.3
All random (frequency)	59.0
BERT	60.8

Table B.1. Comparison of masking patterns, 8-class secondary structure prediction accuracy. Models are pre-trained for 10 epochs on Uniref50. The model and all other hyperparameters remain fixed between experiments. The BERT masking pattern performs best and is used for future experiments.

Hyperparameter: Dynamic batching Models in section 4 were trained with dynamic batching, which results in a single sequence in each sample in the batch. This design choice contrasts with existing language modeling works. For example, in NLP, Liu et al. (40) and Devlin et al. (41), use a static batching approach, wherein multiple proteins are concatenated in the same batch along the sequence dimension. This approach is common in NLP, as sentences that are nearby in a corpus generally relate to the same topic. As this situation is not the case in protein language modeling, we analyze the impact of static batching schemes in protein language models, finding that they reduce model performance (Table B.2). 100M parameter models are evaluated on the secondary structure downstream task after 10 epochs.

Batching mode	CB513
Static	56.2
Static [cropping long sequences]	57.0
Dynamic	60.8

Table B.2. Comparison of batching modes, 8-class secondary structure accuracy. Batching modes are investigated on 100M parameter models and trained for 10 epochs. All models are trained with a context size of 1024 tokens. In the static batching mode, sequences longer than 1024 can span multiple batches. We crop long sequences for the other batching modes considered. The dynamic batching scheme we used performs significantly better than static batching modes.

Hyperparameters: further sweeps After fixing the data distribution to use the BERT masking scheme with dynamic batching, we next perform a hyperparameter sweep to identify the best learning rates, initializations and layer norm placement. We also propose a token dropout scheme which further improves performance on downstream tasks.

Initializations We compare our initializations to the initializations presented in Liu et al. (40), finding that they perform similarly (Table B.3).

Initializations	CB513
Radford et al. (42)	60.9
Liu et al. (40)	60.8

Table B.3. Comparison of initializations, 8-class secondary structure accuracy. 100M parameter models are trained for 10 epochs and evaluated on CB513.

Layer norm placement Recent works Child et al. (43) and Shoeybi et al. (44) have suggested that pre-activation layer norm results in more stable training for larger models. We investigate the impact of this choice in a smaller controlled setting using 100M parameter transformer models, finding that pre-activation layer norm improves performance on downstream tasks (Table B.4). To account for the lack of a final layer norm, we additionally add a final layer norm before the linear output projection, improving performance (Table B.5).

Model	CB513
Post-activation	60.8
Pre-activation	61.1

Table B.4. Comparison of layer norm placement, 8 class secondary structure accuracy. 100M parameter models are trained for 10 epochs and evaluated on CB513.

Final layer norm	CB513
No	61.3
Yes	61.9

Table B.5. Including a final layer norm before the language modeling head performs best, 8-class secondary structure accuracy. 100M parameter models are trained for 12 epochs.

Token dropout Usually, masked language models are pre-trained with corrupted inputs and fine-tuned with complete sequences. We hypothesize that this shift in data distribution reduces performance on downstream tasks. Therefore, we propose a token dropout scheme which replaces the mask token embedding with a fixed tensor of zeros. As $10 - 15\%$ of positions are masked, the zero tensors cause a change in the mean statistics of the word embeddings. We therefore adjust the distribution during fine-tuning by multiplying the word embeddings by a fixed constant. Formally, if a mask token is introduced during pre-training with probability $p = 0.15 \cdot 0.8$, then during fine-tuning, we multiply the embeddings by $1/p$. We find that this significantly improves performance (Table B.6).

Masking	CB513
Mask Tokens	61.9
Token Dropout	62.6

Table B.6. Token dropout performs better than mask tokens, 8-class secondary structure accuracy. 100M parameter models are trained for 12 epochs.

Learning rate Next, we investigate learning rate, finding that a peak learning rate of $4 \cdot 10^{-4}$ performs best (Table B.7). Higher learning rates result in instability, while lower learning rates result in lower performance. In all cases, learning rate is warmed up linearly for 16000 steps and then decayed following an inverse square-root schedule (45).

Learning rate	CB513
$1 \cdot 10^{-4}$	61.9
$2 \cdot 10^{-4}$	63.7
$4 \cdot 10^{-4}$	65.2

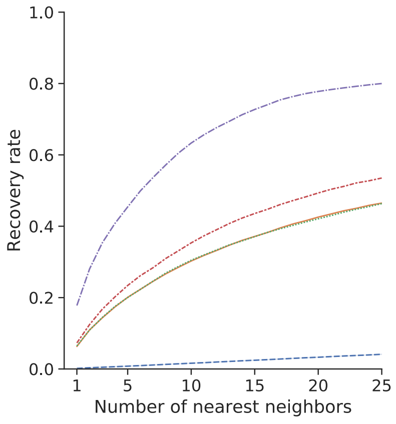
Table B.7. A peak learning rate of $4 \cdot 10^{-4}$ performs best, 8 class secondary structure accuracy. 100M parameter models are trained for 10 epochs.

Positional embeddings	Weights	CB513
Harmonic	Untied	63.7
Harmonic	Tied	64.8
Learned	Untied	65.7
Learned	Tied	66.0

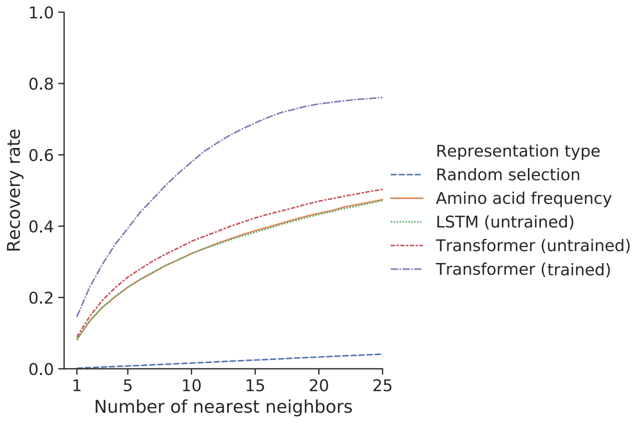
Table B.8. Comparison of tied embeddings and l.earned or harmonic positional embeddings, 8 class secondary structure accuracy. 100M parameter models are trained for 17 epochs.

Tying embeddings Although all experiments above are performed with tied input and output embeddings, we investigate whether learning these separately could improve performance. Our results (Table B.8) indicate that sharing the weights negatively impacts performance. Therefore, we maintain our initial setup.

Positional embeddings We also investigate the impact of learning positional embeddings compared to fixed harmonic embeddings (46), finding that learned positional embeddings perform better (Table B.8).



(a) Recovery rates under group translation



(b) Recovery rate under species translation

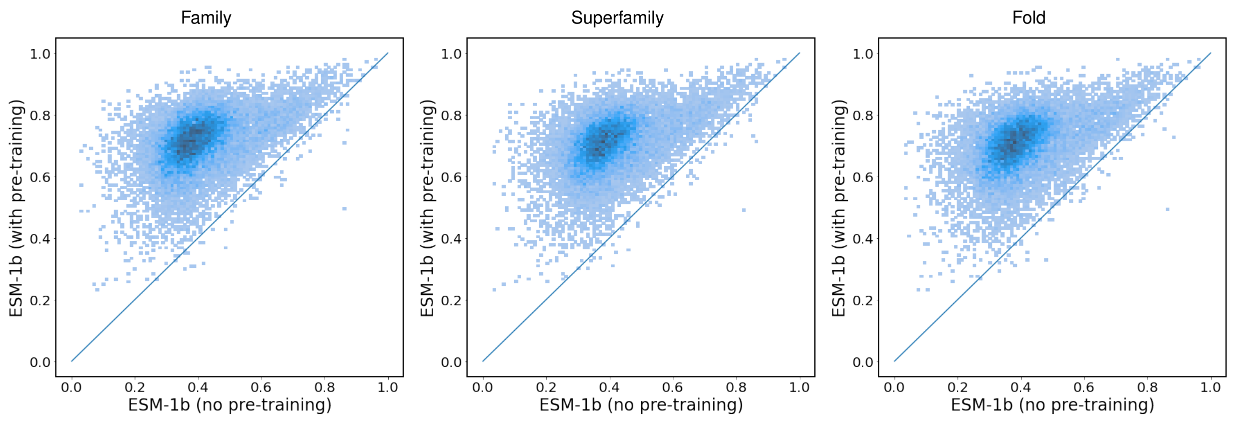
Fig. S1. Learned sequence representations can be translated between orthologous groups and species. Depicted is the recovery rate of nearest-neighbor search under (a) orthologous group translation, and (b) species translation. In both settings, the trained Transformer representation space has a higher recovery rate. Results shown for 36-layer dev Transformer pre-trained on UniParc. To define a linear translation between protein a and protein b of the same species, we define the source and target sets as the average of protein a or protein b across all 24 diverse species. If representation space linearly encodes orthology, then adding the difference in these averages to protein a of some species will recover protein b in the same species. We use an analogous approach to translate a protein of a source species s to its ortholog in the target species t. Here, we consider the average representation of the proteins in s and in t. If representation space is organized linearly by species, then adding the difference in average representations to a protein in species s will recover the corresponding protein in species t.

Representation	**Overall**	**Identical A.A. pairs**	**Distinct A.A. pairs**
Transformer (trained)	**0.841**	**0.870**	**0.792**
Transformer (untrained)	0.656	0.588	0.468

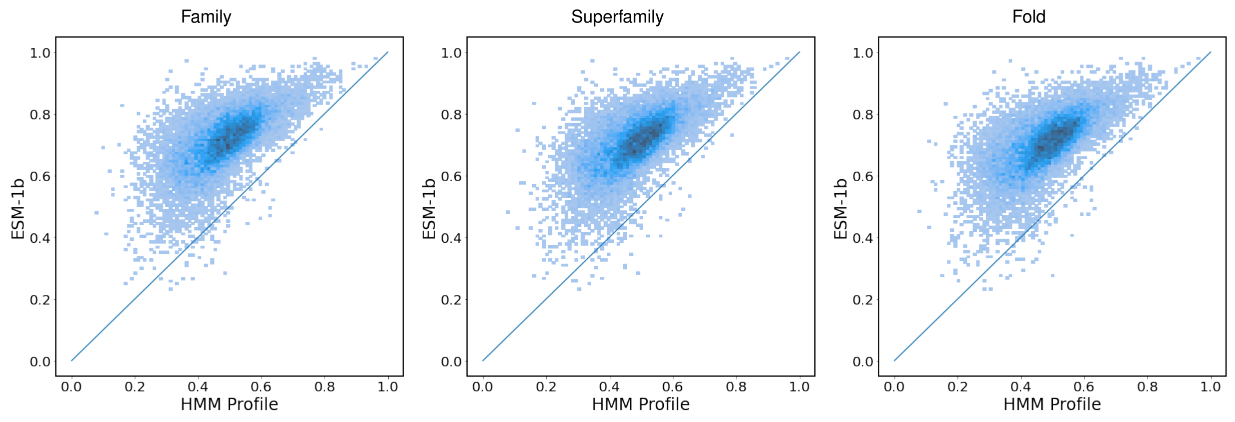
Table S1. Area under the ROC curve (AUC) of per-residue representational cosine similarities in distinguishing between aligned and unaligned pairs of residues within a Pfam family. Results displayed are averaged across 128 families.

Representation	PF00005	PF00069	PF00072
Amino acid identity	0.516	0.506	0.536
12-layer (untrained)	0.818	0.719	0.835
12-layer (PF00005)	<u>0.864</u>	0.725	0.842
12-layer (PF00069)	0.816	<u>0.842</u>	0.850
12-layer (PF00072)	0.789	0.688	<u>0.888</u>
12-layer (UniParc)	0.900	0.872	**0.906**
36-layer (UniParc)	**0.902**	**0.884**	0.902

Table S2. Three-class secondary structure prediction accuracy by linear projection. Learning across many protein families produces better representations than learning from single protein families. Transformer models are trained on three PFAM families: ATP-binding domain of the ABC transporters (PF00005), Protein kinase domain (PF00069), and Response regulator receiver domain (PF00072). The single-family models are contrasted with models trained on the full UniParc data. Comparisons are relative to the family (columnwise), since each of the families differ in difficulty. Underline indicates models trained and evaluated on the same family. Representations learned from single families perform well within the family, but do not generalize as well to sequences outside the family. Representations trained on UniParc outperform the single-family representations in all cases.

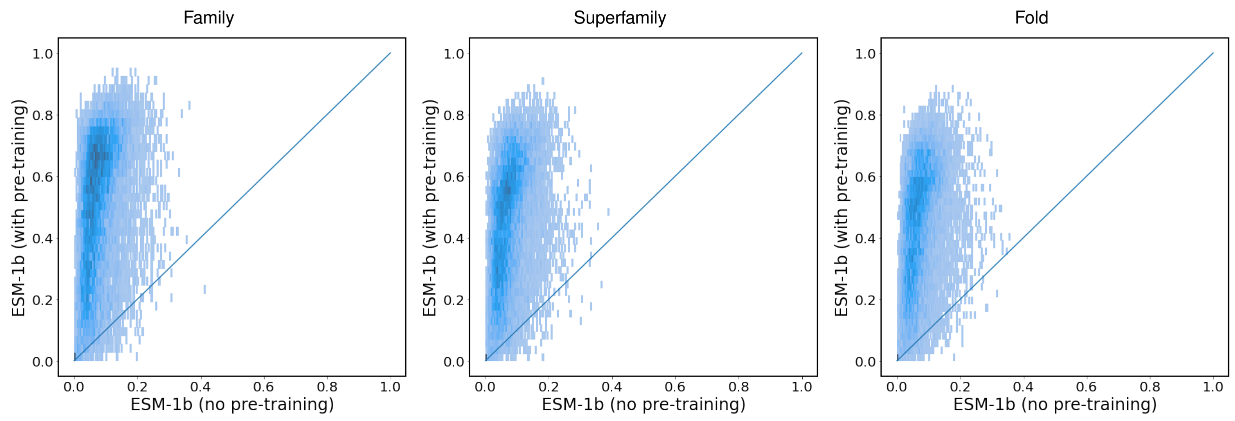


(a) Linear projections pre-training vs no pre-training

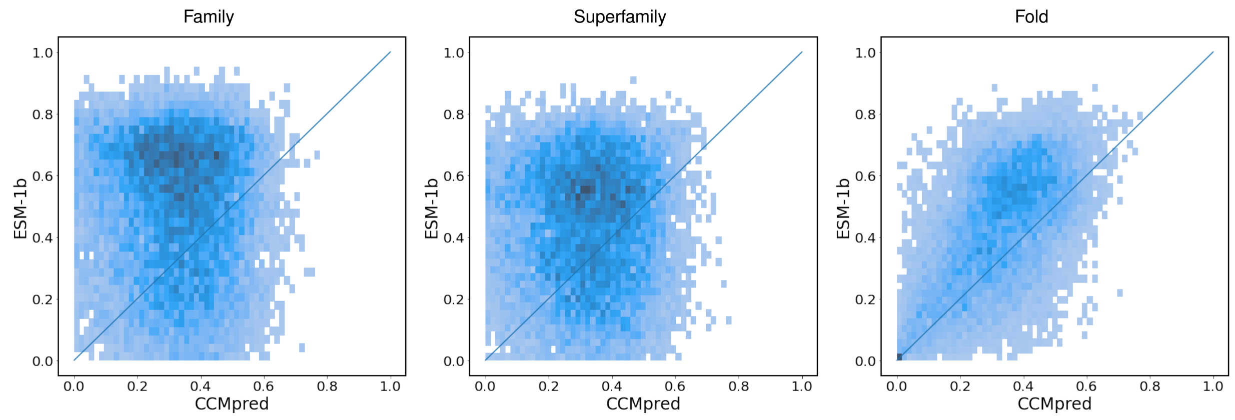


(b) Linear projections ESM1-b vs HMM Profile

Fig. S2. Secondary structure prediction 8-class accuracy distributions for linear projections. (a) Comparison with and without pretraining; (b) comparison of ESM-1b Transformer representations with HMM sequence profiles. Density is indicated by color.



(a) Linear projections pre-training vs no pre-training



(b) Linear projections ESM1-b vs CCMpred

Fig. S3. Contact prediction Top-L long-range precision distributions for linear projections. (a) Comparison with and without pretraining; (b) comparison of ESM-1b Transformer representations with CCMpred predictions. Density is indicated by color.

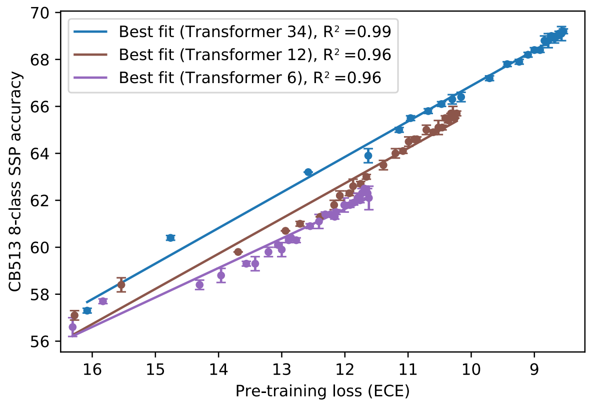


Fig. S4. Eight-class secondary structure prediction accuracy as a function of pre-training ECE. A deep secondary structure predictor is trained using features from Transformer models over the course of pre-training on UR50/S. The Netsurf training sequences and CB513 test set are used. Averages across three seeds of the downstream model per pre-training checkpoint are plotted, with line of best fit for each Transformer.

Metric: top-		L/5,LR	L,LR	L/5,MR	L,MR	L/5,SR	L,SR
Test	(a) RaptorX	84.3 $\pm$ .2	59.4 $\pm$ .2	74.4 $\pm$ .1	33.0 $\pm$ .0	71.6 $\pm$ .1	25.8 $\pm$ .0
	(b) +direct	86.7 $\pm$ .3	61.7 $\pm$ .4	76.5 $\pm$ .3	33.8 $\pm$ .1	**73.6 $\pm$ .2**	**26.1 $\pm$ .1**
	(c) +avg	**87.7 $\pm$ .3**	62.9 $\pm$ .4	77.4 $\pm$ .2	**34.0 $\pm$ .1**	**73.7 $\pm$ .3**	**26.1 $\pm$ .1**
	(d) +cov	**87.8 $\pm$ .3**	**63.3 $\pm$ .2**	**77.6 $\pm$ .2**	**34.0 $\pm$ .1**	**73.7 $\pm$ .2**	**26.1 $\pm$ .0**
CASP11	(a) RaptorX	77.5 $\pm$ .4	53.8 $\pm$ .3	75.0 $\pm$ .5	35.6 $\pm$ .2	72.1 $\pm$ .6	28.6 $\pm$ .2
	(b) +direct	78.3 $\pm$ .1	55.0 $\pm$ .1	76.2 $\pm$ .4	35.9 $\pm$ .2	74.0 $\pm$ .5	28.8 $\pm$ .2
	(c) +avg	**80.4 $\pm$ .5**	56.6 $\pm$ .4	**76.5 $\pm$ .4**	36.3 $\pm$ .2	**73.8 $\pm$ .5**	28.8 $\pm$ .1
	(d) +cov	**80.3 $\pm$ .3**	**56.8 $\pm$ .2**	**76.6 $\pm$ .4**	**36.5 $\pm$ .2**	**74.0 $\pm$ .4**	**29.0 $\pm$ .0**
CASP12-AVG	(a) RaptorX	72.7 $\pm$ .6	51.1 $\pm$ .2	68.3 $\pm$ .5	31.2 $\pm$ .3	66.5 $\pm$ .2	26.3 $\pm$ .1
	(b) +direct	74.0 $\pm$ .8	51.5 $\pm$ .5	70.7 $\pm$ .7	**32.4 $\pm$ .3**	68.9 $\pm$ .9	**27.2 $\pm$ .2**
	(c) +avg	74.4 $\pm$ 1.4	52.4 $\pm$ .5	**71.7 $\pm$ .6**	**32.2 $\pm$ .3**	**70.1 $\pm$ .2**	26.9 $\pm$ .2
	(d) +cov	**76.6 $\pm$ .7**	**53.0 $\pm$ .3**	70.1 $\pm$ .3	31.9 $\pm$ .3	69.1 $\pm$ .5	26.6 $\pm$ .1
CASP12-ENS	(a) RaptorX	77.1 $\pm$ .9	54.5 $\pm$ .4	70.6 $\pm$ .6	32.4 $\pm$ .4	68.6 $\pm$ .4	27.0 $\pm$ .1
	(b) +direct	77.0 $\pm$ .6	53.5 $\pm$ .6	**71.9 $\pm$ .9**	**33.1 $\pm$ .3**	69.8 $\pm$ .7	**27.6 $\pm$ .2**
	(c) +avg	76.7 $\pm$ 1.4	54.4 $\pm$ .7	**74.1 $\pm$ .8**	**33.0 $\pm$ .3**	**71.5 $\pm$ .3**	27.4 $\pm$ .2
	(d) +cov	**79.7 $\pm$ .8**	**55.3 $\pm$ .2**	72.7 $\pm$ .5	32.7 $\pm$ .3	71.0 $\pm$ .6	27.2 $\pm$ .2
CASP13-AVG	(a) RaptorX	68.0 $\pm$ .9	43.4 $\pm$ .4	71.3 $\pm$ .4	36.5 $\pm$ .3	68.8 $\pm$ 1.0	28.4 $\pm$ .0
	(b) +direct	67.4 $\pm$ .8	43.7 $\pm$ .4	69.5 $\pm$ .9	35.5 $\pm$ .4	68.1 $\pm$ .4	28.3 $\pm$ .2
	(c) +avg	68.1 $\pm$ 1.6	**44.8 $\pm$ .8**	**73.0 $\pm$ .6**	**36.9 $\pm$ .1**	**71.2 $\pm$ .6**	**28.6 $\pm$ .2**
	(d) +cov	**70.3 $\pm$ 1.3**	**45.2 $\pm$ .5**	**73.5 $\pm$ 1.4**	**37.0 $\pm$ .2**	70.2 $\pm$ .8	**28.6 $\pm$ .3**
CASP13-ENS	(a) RaptorX	**72.1 $\pm$ .8**	46.3 $\pm$ .5	73.6 $\pm$ .4	**38.1 $\pm$ .3**	71.0 $\pm$ 1.4	**29.6 $\pm$ .1**
	(b) +direct	68.0 $\pm$ .7	45.0 $\pm$ .6	71.4 $\pm$ 1.2	36.6 $\pm$ .4	70.2 $\pm$ .2	29.1 $\pm$ .2
	(c) +avg	70.8 $\pm$ 2.2	46.4 $\pm$ 1.1	**75.4 $\pm$ .5**	**38.1 $\pm$ .2**	**73.6 $\pm$ .5**	29.3 $\pm$ .2
	(d) +cov	71.9 $\pm$ 1.9	**47.2 $\pm$ .4**	**75.2 $\pm$ 1.5**	**38.3 $\pm$ .4**	**72.3 $\pm$ .8**	29.3 $\pm$ .2

Table S3. Additional metrics. Feature combination for supervised contact prediction. AVG corresponds to the average of the metrics over the different MSAs, while in ENS the probabilities are averaged (ensembled) over the different MSA predictions before computing the metrics.

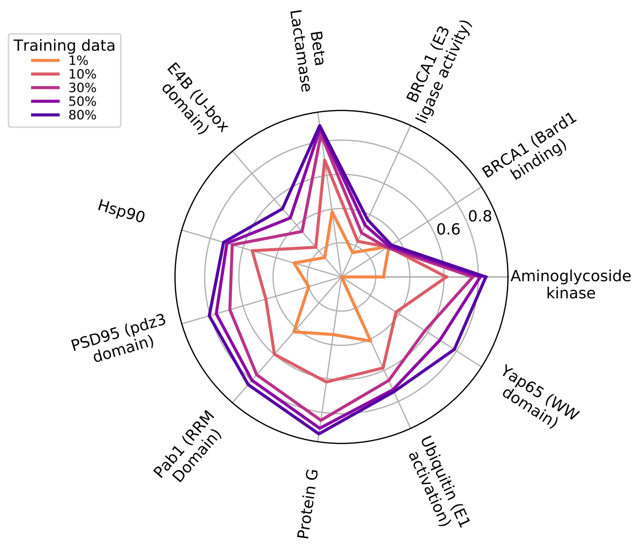


Fig. S5. After pre-training, the Transformer can be adapted to predict mutational effects on protein function. The 34-layer Transformer model pre-trained on UR50/S is fine-tuned on mutagenesis data. Spearman ρ on each protein when supervised with smaller fractions of the data.

Amount of training data Protein	1% data	10% data	30% data	50% data	80% data
Aminoglycoside kinase	0.25 ± 0.07	0.61 ± 0.02	0.77 ± 0.01	0.81 ± 0.01	0.84 ± 0.01
BRCA1 (Bard1 binding)	0.33 ± 0.02	0.32 ± 0.01	0.32 ± 0.03	0.33 ± 0.03	0.35 ± 0.03
BRCA1 (E3 ligase activity)	0.16 ± 0.01	0.23 ± 0.03	0.28 ± 0.07	0.33 ± 0.05	0.37 ± 0.04
Beta Lactamase	0.39 ± 0.03	0.69 ± 0.01	0.84 ± 0.01	0.88 ± 0.01	0.89 ± 0.01
E4B (U-box domain)	0.15 ± 0.03	0.23 ± 0.03	0.35 ± 0.01	0.46 ± 0.03	0.53 ± 0.04
Hsp90	0.29 ± 0.02	0.54 ± 0.01	0.67 ± 0.01	0.70 ± 0.02	0.72 ± 0.05
PSD95 (pdz3 domain)	0.20 ± 0.02	0.46 ± 0.05	0.68 ± 0.01	0.76 ± 0.01	0.81 ± 0.02
Pab1 (RRM Domain)	0.42 ± 0.07	0.60 ± 0.02	0.76 ± 0.01	0.80 ± 0.01	0.83 ± 0.02
Protein G	0.34 ± 0.15	0.62 ± 0.03	0.85 ± 0.02	0.89 ± 0.02	0.93 ± 0.01
Ubiquitin (E1 activation)	0.41 ± 0.06	0.58 ± 0.02	0.67 ± 0.02	0.72 ± 0.02	0.74 ± 0.02
Yap65 (WW domain)		0.38 ± 0.06	0.58 ± 0.06	0.68 ± 0.05	0.78 ± 0.05

Table S4. Aggregate spearman ρ measured across models and datasets. Mean and standard deviations of spearman ρ performance for the fine-tuned Transformer-34 on intraprotein tasks. Performance was assessed on five random partitions of the validation set. Model pre-trained on UR50/S.

dataset model	spearmanr Envision	Envision (LOPO)	DeepSequence
Transformer	0.71 ± 0.20	0.51	0.70 ± 0.15
LSTM biLM (Large)	0.65 ± 0.22		0.62 ± 0.19
Gray, et al. 2018	0.64 ± 0.21	0.45	
Riesselman, et al. 2018			0.48 ± 0.26

Table S5. Aggregate spearman ρ measure across models and datasets. 34-layer Transformer pre-trained on UR50/S. For intraprotein models, the train/valid data was randomly partitioned five times. The mean $\pm$ standard deviation across the five runs is reported. No standard deviations are reported for LOPO experiments, as the evaluation is performed across all proteins.

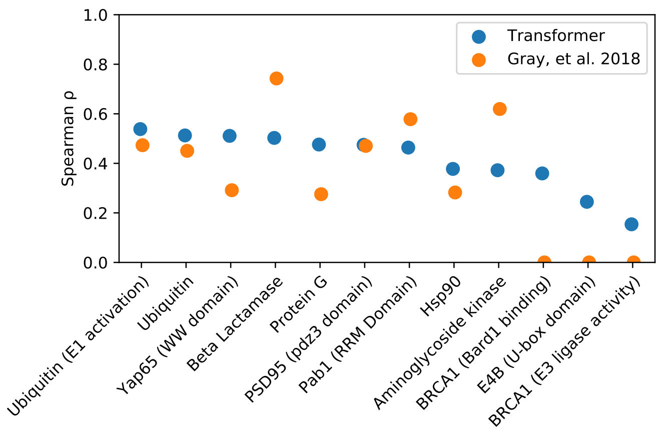
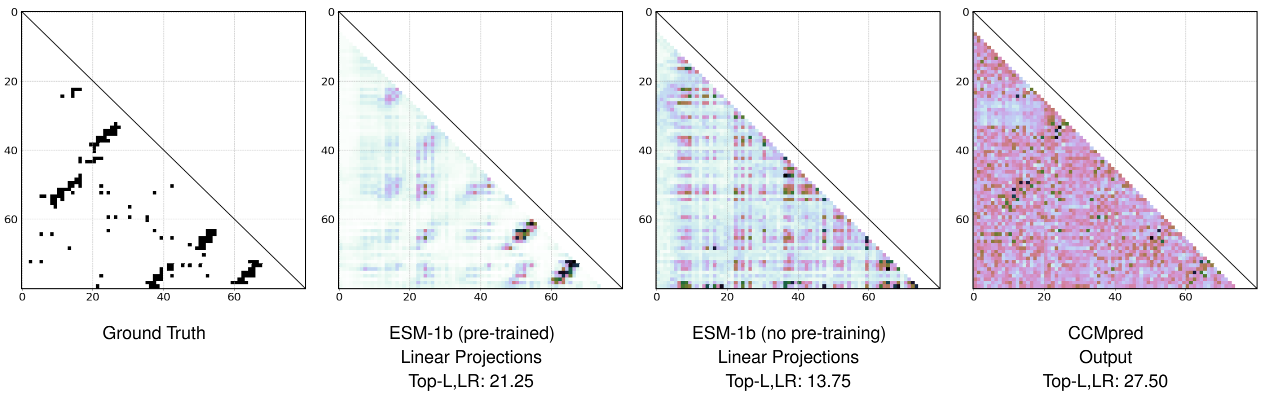
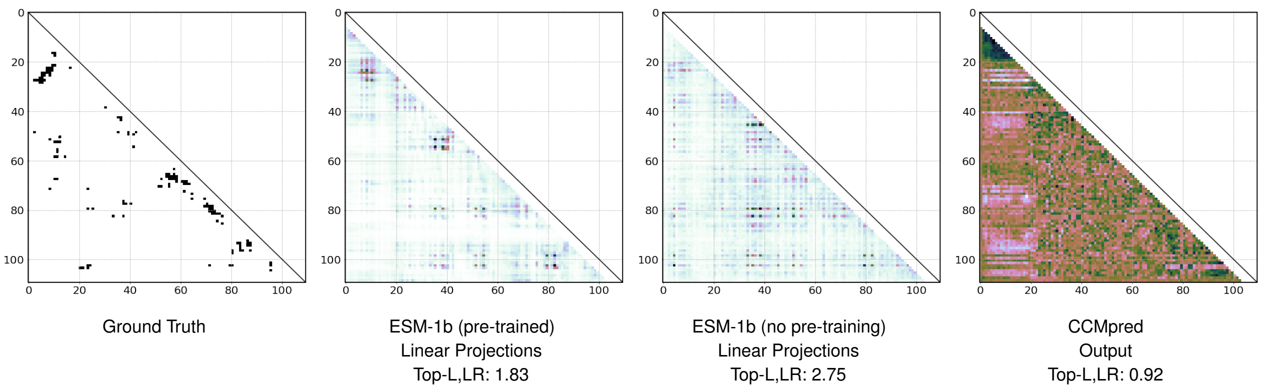


Fig. S6. Leave-one-out experiment on Envision dataset (11). Pre-training improves the ability of the Transformer to generalize to the mutational fitness landscape of held-out proteins. All mutagenesis data from the protein selected for evaluation are held out, and the model is supervised with data from the remaining proteins. For each evaluation protein, a comparison is shown for the 34-layer Transformer pre-trained on UR50/S.

Model	Pre-Training	Params	SSP CB513	SSP CASP13	Contact Test	Contact CASP11	Contact CASP12	Contact CASP13
Transformer-34	(None)	669.2M	56.8	60.0	16.3	17.7	14.8	13.3
UniRep (LSTM)		18.2M	58.4	60.1	21.9	21.4	16.8	14.3
SeqVec (LSTM)		93M	62.1	64.0	29.0	25.5	23.6	17.9
TAPE (Transformer)		38M	58.0	61.5	23.2	23.8	20.3	16.0
LSTM (S)	UR50/S	28.4M	60.4	63.2	24.1	23.6	19.9	15.3
LSTM (L)	UR50/S	113.4M	62.4	64.1	27.8	26.4	24.0	16.4
Transformer-6	UR50/S	42.6M	62.0	64.2	30.2	29.9	25.3	19.8
Transformer-12	UR50/S	85.1M	65.4	67.2	37.7	33.6	27.8	20.7
Transformer-34	UR100	669.2M	64.3	66.5	32.7	28.9	24.3	19.1
Transformer-34	UR50/S	669.2M	69.1	70.7	50.2	42.8	34.7	30.1
ESM-1b	UR50/S	652.4M	**71.6**	**72.5**	**56.9**	**47.4**	**42.7**	**35.9**

Table S6. Comparison to related methods. Top-L long-range contact precision. Predictions are directly from protein sequence, no coevolutionary features or MSAs used. Test is RaptorX test set of Wang et al. (9). Model weights for related work are obtained and evaluated in our codebase with same downstream architecture, training, and test data.


(a) d1bcoa1 (mu transposase, C-terminal domain) (80 residues)


(b) d1gyoa (Multiheme cytochromes) (109 residues)

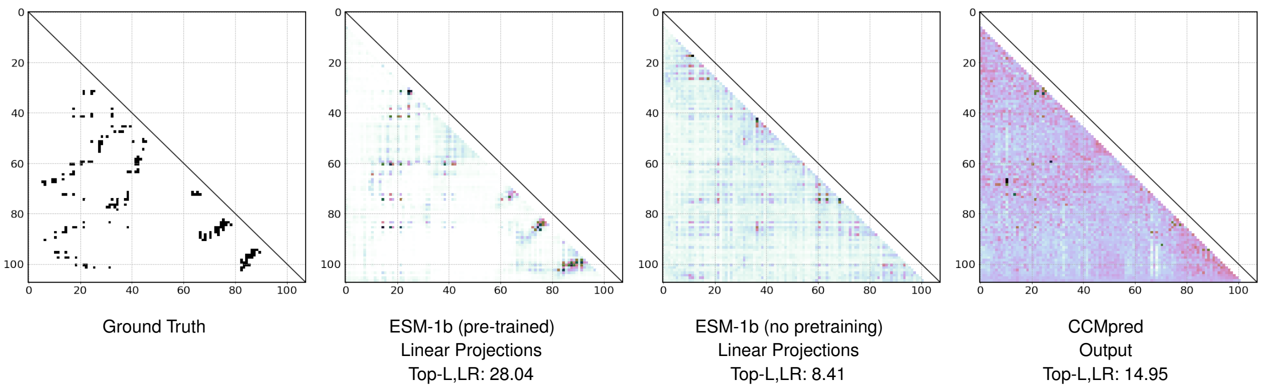
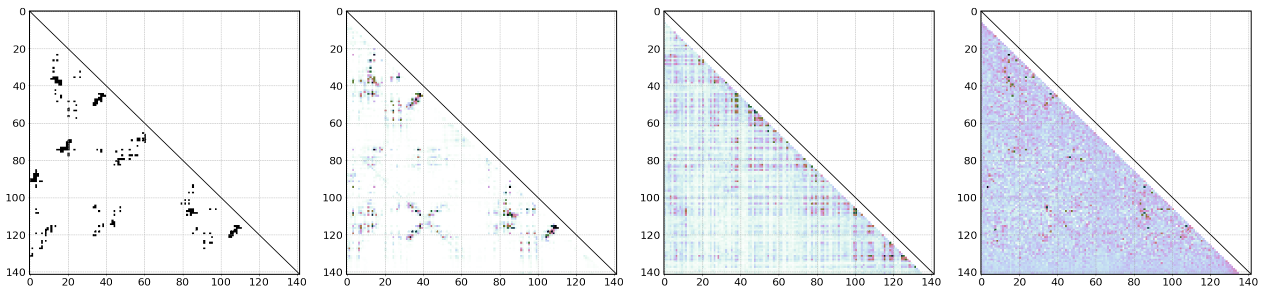

(c) d1t92a1(Pili subunits) (107 residues)

Fig. S7. Comparison of ground truth contact map, projections from ESM-1b with and without pre-training, and CCMpred output. Labels indicate SCOPe domain, fold name, and number of residues. Eight domains randomly sampled from fold-level test sets are shown.

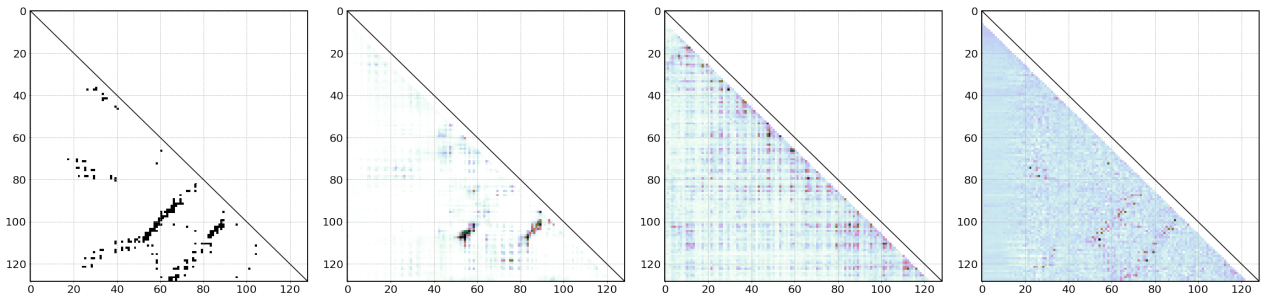

Ground Truth

ESM-1b (pre-trained)
Linear Projections
Top-L,LR: 38.30

ESM-1b (no pre-training)
Linear Projections
Top-L,LR: 7.09

CCMpred
Output
Top-L,LR: 31.91

(d) d3ddja1 (CBS-domain pair) (141 residues)

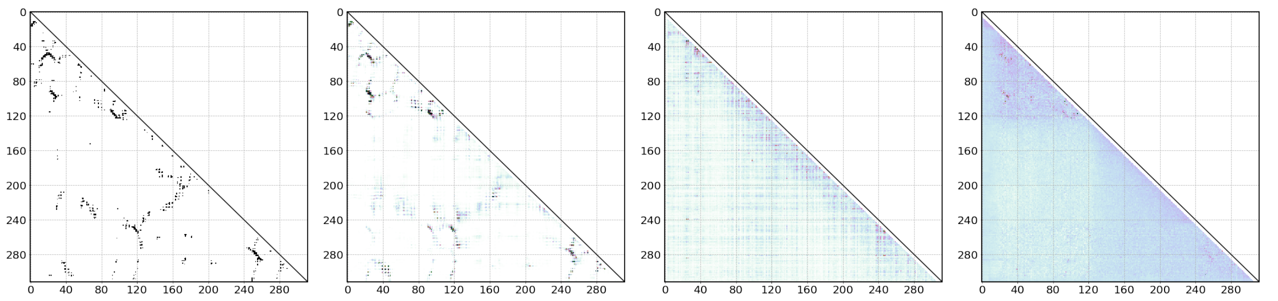

Ground Truth

ESM-1b (pre-trained)
Linear Projections
Top-L,LR: 49.22

ESM-1b (no pre-training)
Linear Projections
Top-L,LR: 6.25

CCMpred
Output
Top-L,LR: 64.06

(e) d3paja1 (alpha/beta-Hammerhead) (128 residues)


Ground Truth

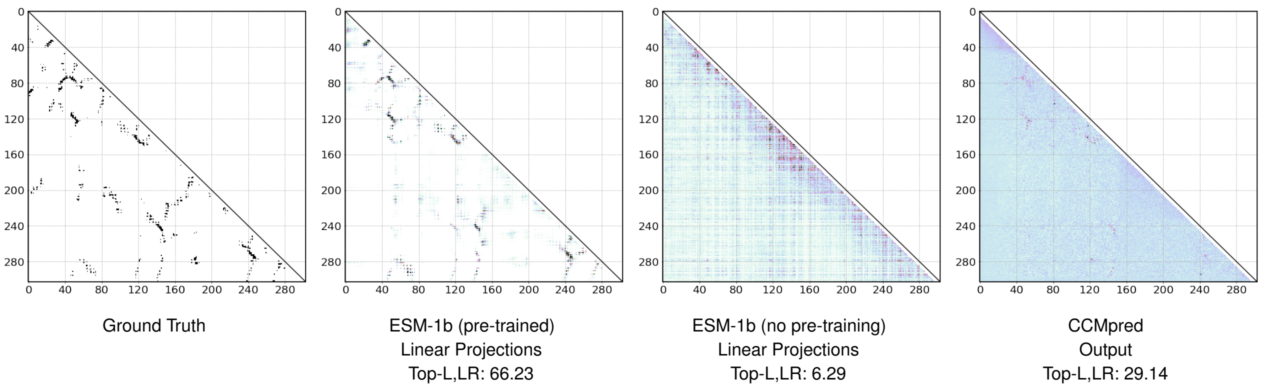
ESM-1b (pre-trained)
Linear Projections
Top-L,LR: 60.13

ESM-1b (no pre-training)
Linear Projections
Top-L,LR: 3.22

CCMpred
Output
Top-L,LR: 29.26

(f) d4jnca1 (alpha/beta-Hydrolases) (311 residues)

Fig. S7. (continued from above)



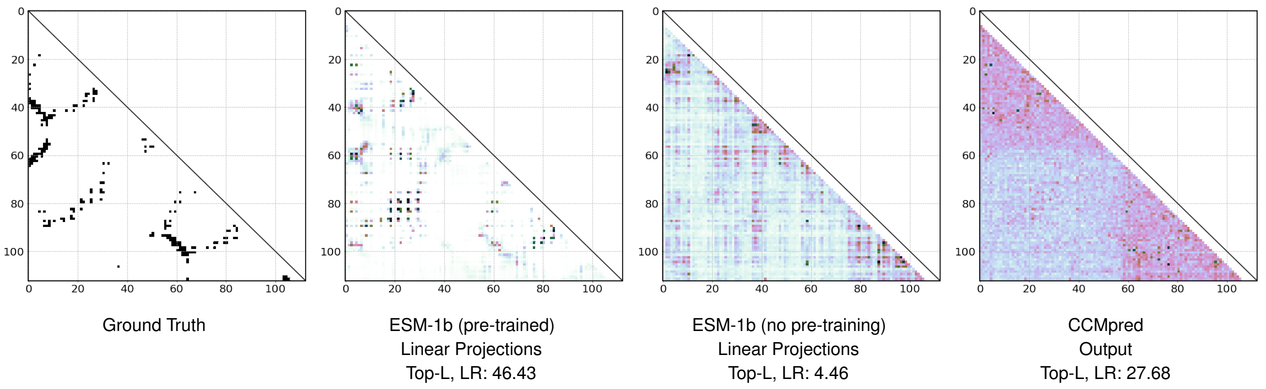
Ground Truth

ESM-1b (pre-trained) Linear Projections Top-L,LR: 66.23

ESM-1b (no pre-training) Linear Projections Top-L,LR: 6.29

CCMpred Output Top-L,LR: 29.14

(g) d5esra1 (alpha/beta-Hydrolases) (302 residues)



Ground Truth

ESM-1b (pre-trained) Linear Projections Top-L, LR: 46.43

ESM-1b (no pre-training) Linear Projections Top-L, LR: 4.46

CCMpred Output Top-L, LR: 27.68

(h) d3b64a (Tautomerase/MIF) (112 residues)

Fig. S7. (continued from above)

Supplemental Sources

1. The UniProt Consortium. The universal protein resource (uniprot). *Nucleic Acids Research*, 36(suppl_1):D190–D195, 11 2007. ISSN 0305-1048. .
2. B. E. Suzek, Y. Wang, H. Huang, P. B. McGarvey, C. H. Wu, and U. Consortium. Uniref clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics*, 31(6):926–932, 2015.
3. N. K. Fox, S. E. Brenner, and J.-M. Chandonia. SCOPe: Structural Classification of Proteins—extended, integrating SCOP and ASTRAL data and classification of new structures. *Nucleic Acids Research*, 42(D1): D304–D309, January 2014. ISSN 0305-1048, 1362-4962. . URL https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkt1240.
4. M. S. Klausen, M. C. Jespersen, H. Nielsen, K. K. Jensen, V. I. Jurtz, C. K. Sonderby, M. O. A. Sommer, O. Winther, M. Nielsen, B. Petersen, and P. Marcatili. NetSurfP-2.0: Improved prediction of protein structural features by integrated deep learning. *Proteins*, 87(6):520–527, 06 2019.
5. M. Steinegger and J. Söding. Mmseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature biotechnology*, 35(11):1026–1028, 2017.
6. M. Mirdita, L. von den Driesch, C. Galiez, M. J. Martin, J. Söding, and M. Steinegger. Uniclust databases of clustered and deeply annotated protein sequences and alignments. *Nucleic acids research*, 45(D1):D170–D176, 2017.
7. J. A. Cuff and G. J. Barton. Evaluation and improvement of multiple sequence methods for protein secondary structure prediction. *Proteins: Structure, Function, and Bioinformatics*, 34(4):508–519, 1999.
8. W. Kabsch and C. Sander. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22(12):2577–2637, 1983.
9. S. Wang, S. Sun, Z. Li, R. Zhang, and J. Xu. Accurate de novo prediction of protein contact map by ultra-deep learning model. *PLoS computational biology*, 13(1):e1005324, 2017.
10. J. Xu. Distance-based protein folding powered by deep learning. *arXiv preprint arXiv:1811.03481*, 2018.
11. V. E. Gray, R. J. Hause, J. Luebeck, J. Shendure, and D. M. Fowler. Quantitative missense variant effect prediction using large-scale mutagenesis data. *Cell systems*, 6 (1):116–124, 2018.
12. A. J. Riesselman, J. B. Ingraham, and D. S. Marks. Deep generative models of genetic variation capture the effects of mutations. *Nature Methods*, 15(10):816–822, 2018. ISSN 1548-7105. .
13. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.
14. J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.
15. U. Khandelwal, H. He, P. Qi, and D. Jurafsky. Sharp nearby, fuzzy far away: How neural language models use context. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 284–294, 2018.
16. S. Hochreiter, Y. Bengio, and P. Frasconi. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. 2001.
17. A. S. Lapedes, B. G. Giraud, L. Liu, and G. D. Stormo. Correlated mutations in models of protein sequences: phylogenetic and structural effects. *Lecture Notes-Monograph Series*, pages 236–256, 1999.
18. J. Thomas, N. Ramakrishnan, and C. Bailey-Kellogg. Graphical models of residue coupling in protein families. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 5(2):183–197, 2008.
19. M. Weigt, R. A. White, H. Szurmant, J. A. Hoch, and T. Hwa. Identification of direct residue contacts in protein–protein interaction by message passing. *Proceedings of the National Academy of Sciences*, 106(1):67–72, 2009.
20. K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.
21. J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
22. A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.
23. D. Hendrycks and K. Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.
24. M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.
25. S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. .
26. M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237, 2018. URL https://aclanthology.info/papers/N18-1202/n18-1202.
27. J. Huerta-Cepas, S. K. Forslund, P. Bork, A. Hernández-Plaza, C. von Mering, D. Szklarczyk, D. Heller, H. Cook, L. Jensen, D. R. Mende, I. Letunic, and T. Rattei. eggNOG 5.0: a hierarchical, functionally and phylogenetically annotated orthology resource based on 5090 organisms and 2502 viruses. *Nucleic Acids Research*, 47(D1):D309–D314, 11 2018. ISSN 0305-1048. .
28. J. Söding and M. Remmert. Protein sequence comparison and fold recognition: progress and good-practice benchmarking. *Current opinion in structural biology*, 21(3): 404–411, 2011.
29. M. Remmert, A. Biegert, A. Hauser, and J. Söding. Hhblits: lightning-fast iterative protein sequence searching by hmm-hmm alignment. *Nature Methods*, 9:173 EP, 12 2011. .
30. J. Ma, S. Wang, Z. Wang, and J. Xu. Mrfalign: protein homology detection through alignment of markov random fields. In *International Conference on Research in Computational Molecular Biology*, pages 173–174. Springer, 2014.
31. J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with gpus. *CoRR*, abs/1702.08734, 2017. URL

http://arxiv.org/abs/1702.08734.

32. A. Bateman, A. Heger, E. L. L. Sonnhammer, J. Mistry, J. Clements, J. Tate, K. Hetherington, L. Holm, M. Punta, P. Coggill, R. Y. Eberhardt, S. R. Eddy, and R. D. Finn. Pfam: the protein families database. *Nucleic Acids Research*, 42(D1):D222–D230, 11 2013. ISSN 0305-1048. .
33. R. P. Joosten, T. A. Te Beek, E. Krieger, M. L. Hekkelman, R. W. Hooft, R. Schneider, C. Sander, and G. Vriend. A series of pdb related databases for everyday needs. *Nucleic acids research*, 39(suppl_1):D411–D419, 2010.
34. S. Seemayer, M. Gruber, and J. Söding. Ccmpred—fast and precise prediction of protein residue–residue contacts from correlated mutations. *Bioinformatics*, 30(21):3128–3130, 2014.
35. S. Balakrishnan, H. Kamisetty, J. G. Carbonell, S.-I. Lee, and C. J. Langmead. Learning generative models for protein fold families. *Proteins: Structure, Function, and Bioinformatics*, 79(4):1061–1078, 2011.
36. M. Ekeberg, C. Lövkvist, Y. Lan, M. Weigt, and E. Aurell. Improved contact prediction in proteins: using pseudolikelihoods to infer potts models. *Physical Review E*, 87(1): 012707, 2013.
37. I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.
38. B. Adhikari. DEEPCON: protein contact prediction using dilated convolutional neural networks with dropout. *Bioinformatics*, 36(2):470–477, 07 2019.
39. A. Senior, J. Jumper, and D. Hassabis. AlphaFold: Using AI for scientific discovery, 12 2018. URL https://deepmind.com/blog/alphafold/.
40. Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. 7 2019. URL http://arxiv.org/abs/1907.11692.
41. J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.
42. A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.
43. R. Child, S. Gray, A. Radford, and I. Sutskever. Generating long sequences with sparse transformers. *URL https://openai.com/blog/sparse-transformers*, 2019.
44. M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism, 2020.
45. C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2020.
46. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.