NEURAL POTTS MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose the Neural Potts Model objective as an amortized optimization problem. The objective enables training a single model with shared parameters to explicitly model energy landscapes across multiple protein families. Given a protein sequence as input, the model is trained to predict a pairwise coupling matrix for a Potts model energy function describing the local evolutionary landscape of the sequence. Plausible couplings are predicted for train and validation sequences. A controlled ablation experiment assessing unsupervised contact prediction on sets of related protein families finds a gain from amortization for low-depth MSAs; the result is confirmed on a larger database with broad coverage of protein sequences.

1 INTRODUCTION

Recently, language modeling has emerged as a promising avenue for learning representations of protein sequences that are useful across a variety of protein modeling tasks (Rives et al., 2019; Alley et al., 2019; Rao et al., 2019; Heinzinger et al., 2019). The self supervision objectives from NLP studied for protein sequence modeling use the input sequence directly. In this paper we extend self-supervision to information from a set of evolutionarily related sequences to shape a model to predict a local evolutionary energy landscape for the input sequence.

The standard method for unsupervised contact prediction fits a linear or log linear model to a multiple sequence alignment (MSA) summarizing evolutionary variation around the input sequence (Balakrishnan et al., 2011; Ekeberg et al., 2013; Morcos et al., 2011; Marks et al., 2011; Jones et al., 2011; Kamisetty et al., 2013). To construct the MSA for an input sequence, a similarity query is performed across a large database to identify related sequences, and the returned set of sequences are aligned to each other. Coevolution of a pair of positions in the alignment is associated with a spatial contact between their amino acids in the folded structure (Göbel et al., 1994). Statistical couplings between positions in the MSA can be used to infer protein contacts (Weigt et al., 2009). These couplings can be extracted from a Potts Model trained with pseudolikelihood on the MSA (Balakrishnan et al., 2011; Ekeberg et al., 2013). Contact prediction performance depends on the depth of the MSA and is reduced when few related proteins can be used to fit the model.

We introduce the Neural Potts Model (NPM) objective. The objective is formally expressed as an amortized optimization problem across sequences and MSAs. A Transformer model is trained to predict the parameters of a Potts model energy function defined by the MSA of each input sequence. The hope is that the model will learn to generalize at test time to predict energy functions for sequences not seen during training or for which limited MSA information is available.

We investigate the approach in a controlled ablation experiment on a group of related protein families in PFAM (Bateman et al., 2004). In this artificial setting information can be generalized by the shared parameters to improve unsupervised contact prediction on a subset of the MSAs that have been artificially purged to reduce their number of sequences. We then study the model in the setting of a large dataset without artificial purging, training the model on MSAs for UniRef50 (Suzek et al., 2007) sequences. In this setting there is also an improvement on average for low depth MSAs both for sequence in the training set as well as for sequences held-out from training.

2 BACKGROUND

Multiple sequence alignments An MSA is a set of aligned protein sequences that are evolutionarily related. MSAs are constructed by retrieving related sequences from a sequence database and aligning

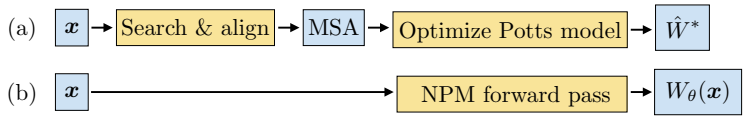

Figure 1: (a) Standard Potts model requires constructing an MSA and optimizing parameters W. (b) Neural Potts Model (NPM) predicts W in a single feedforward pass from a single sequence.

the returned sequences using a heuristic. An MSA can be viewed as a matrix where each row is a sequence, and columns contain aligned positions after removing insertions and replacing deletions with gap characters.

Potts model The generalized Potts model defines a Gibbs distribution over a protein sequence $(x_1, \ldots, x_L)$ of length L with the negative energy function:

$$-E(\boldsymbol{x}) = \sum_i h_i(x_i) + \sum_{ij} J_{ij}(x_i, x_j) \tag{1}$$

Which defines potentials h_i for each position in the sequence, and couplings J_{ij} for every pair of positions. The parameters of the model are $W = \{h, J\}$ the set of fields and couplings respectively. The distribution $p(\boldsymbol{x}; W)$ is obtained by normalization as $\exp\{-E(\boldsymbol{x}; W)\}/Z(W)$.

Since the normalization constant is intractible, pseudolikelihood is commonly used to fit the parameters (Balakrishnan et al., 2011; Ekeberg et al., 2013). Pseudolikelihood approximates the likelihood of a sequence $\boldsymbol{x}$ as a product of conditional distributions: $\ell_{\mathrm{PL}}(\boldsymbol{x}; W) = -\sum_i \log p(x_i | x_{-i}; W)$. To estimate the Potts model, we take the expectation:

$$\mathcal{L}_{\mathrm{PL}}(W) = \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{M}} [\ell_{\mathrm{PL}}(\boldsymbol{x}; W)] \tag{2}$$

over an MSA $\mathcal{M}$. In practice, we have a finite set of sequences $\hat{\mathcal{M}}$ in the MSA to estimate Eq. (2). L_2 regularization $\rho(W)$ is added, and sequences are reweighted to account for redundancy Morcos et al. (2011). We write the regularized finite sample estimator as:

$$\hat{\mathcal{L}}_{\mathrm{PL}}(W) = \frac{1}{M_{\mathrm{eff}}} \sum_{m=1}^{M} w^m [\ell_{\mathrm{PL}}(\boldsymbol{x}^m; W)] + \rho(W) \tag{3}$$

Which sums over all the M sequences of the finite MSA $\hat{\mathcal{M}}$, weighted with w^m summing collectively to M_{eff}. The finite sample estimate of the parameters $\hat{W}^*$ is obtained by minimizing $\hat{\mathcal{L}}_{\mathrm{PL}}$.

Idealized MSA Notice how in Eq. (2), we idealized the MSA $\mathcal{M}$ as a distribution, defined by the protein family. We consider the set of sequences actually retrieved in the MSA $\hat{\mathcal{M}}$ in Eq. (3) as a finite sample from this underlying idealized distribution. For some protein families this sample will contain more information than for others, depending on what sequences are present in the database. We will refer to W^* as a hypothetical idealized estimate of the parameters to explain how the Neural Potts Model can improve on the finite sample estimate $\hat{W}^*$ for low-depth MSAs.

2.1 Amortized Optimization

We review amortized optimization (Shu, 2017), a generalization of amortized variational inference (Kingma & Welling, 2013; Rezende et al., 2014) that uses learning to predict the solution to continuous optimization problems to make the computation more tractable and potentially generalize across problem instances. We are interested in repeatedly solving expensive optimization problems

$$W^*(x) = \arg\min_W \mathcal{L}(W; x), \tag{4}$$

where $W \in \mathbb{R}^m$ is the optimization variable, $x \in \mathbb{R}^n$ is the input or conditioning variable to the optimization problem, and $\mathcal{L} : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$ is the objective. We assume $W^*(x)$ is unique. We consider the setting of having a distribution over optimization problems with inputs $x \sim p(x)$, and the $\arg\min$ of those optimization problems $W^*(x)$.

Amortization uses learning to leverage the shared structure present across the distribution, *e.g.* a solution $W^*(x)$ is likely correlated with another solution $W^*(x')$. Assuming an underlying regularity of the data and loss $\mathcal{L}$, we can imagine learning to predict the outcome of the optimization problem with an expressive model $W_\theta(x)$ such that hopefully $W_\theta \approx W^*$. Modeling and learning $W_\theta(x)$ are the key design decisions when using amortization.

Modeling approaches. In this paper we consider models $W_\theta(x)$ that directly predict the solution to Eq. (4) with a neural network, which follows fully amortized variational inference models and the meta-learning method Mishra et al. (2017). The model can also leverage the objective information $\mathcal{L}(W; x)$ and gradient information $\nabla_W \mathcal{L}(W; x)$, *e.g.* by predicting multiple candidate solutions W and selecting the most optimal one. This is sometimes referred to as semi-amortization or unrolled optimization-based models and is considered in Gregor & LeCun (2010) for sparse coding, Li & Malik (2016); Andrychowicz et al. (2016); Finn et al. (2017) for meta-learning, and Marino et al. (2018); Kim et al. (2018) for posterior optimization.

Learning approaches. There are two main classes of learning approaches for amortization:

$$\arg\min_\theta \mathop{\mathbb{E}}_{p(x)} \mathcal{L}(W_\theta(x); x) \tag{5}$$

$$\arg\min_\theta \mathop{\mathbb{E}}_{p(x)} \|W_\theta(x) - W^*(x)\|_2^2 . \tag{6}$$

Gradient-based approaches leverage gradient information of the objective $\mathcal{L}$ and optimize Eq. (5) whereas *regression-based* approaches optimize a distance to ground-truth solutions W^*, such as the squared L^2 distance in Eq. (6). Prior work has shown that models trained with these objectives can learn to predict the optimal W^* directly as a function of x. Given enough regularity of the domain, if we observe new (test) samples $x' \sim p(x)$ we expect the model to generalize and predict the solution to the original optimization problem Eq. (4). Gradient-based approaches have the computational advantage of not requiring the expensive ground-truth solution W^* while regression-based approaches are less susceptible to poor local optima in $\mathcal{L}$. Gradient-based approaches are used in variational inference (Kingma & Welling, 2013), style transfer (Chen & Schmidt, 2016), meta learning (Finn et al., 2017; Mishra et al., 2017), and reinforcement learning, *e.g.* for the policy update in model-free actor-critic methods (Sutton & Barto, 2018). Regression-based approaches are more common in control for behavioral cloning and imitation learning (Duriez et al., 2017; Ratliff et al., 2007; Bain & Sammut, 1995).

3 Neural Potts Model

In Eq. (2) we introduced the Potts model for a single MSA $\mathcal{M}$ (aligned set of sequences $\boldsymbol{x}$), to optimize $W^* = \{h^*, J^*\} = \arg\min_W \mathbb{E}_{\tilde{\boldsymbol{x}} \sim \mathcal{M}}[\ell_{\mathrm{PL}}(\tilde{\boldsymbol{x}}; W)]$. As per Eq. (5) We will now introduce a neural network to estimate Potts model parameters from a single sequence: $\{h_\theta(\boldsymbol{x}), J_\theta(\boldsymbol{x})\} = W_\theta(\boldsymbol{x})$ with a single forward pass.

We propose minimizing the following objective for the NPM parameters θ, which directly minimizes the Potts model losses in expectation over our data distribution $\boldsymbol{x} \sim \mathcal{D}$ and their MSAs $\tilde{\boldsymbol{x}} \sim \mathcal{M}(\boldsymbol{x})$

$$\mathcal{L}_{\mathrm{NPM}}(\theta) = \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{D}} \left[\mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim \mathcal{M}(\boldsymbol{x})} \ell_{\mathrm{PL}}(\tilde{\boldsymbol{x}}; W_\theta(\boldsymbol{x})) \right] \tag{7}$$

To compute the loss for a given sequence $\boldsymbol{x}$ we compute the Potts model parameters $W_\theta(\boldsymbol{x})$, and evaluate its pseudo-likelihood loss ℓ_{PL} on a set of sequences $\tilde{\boldsymbol{x}}$ from the MSA constructed with $\boldsymbol{x}$ as query sequence. This fits exactly in "amortized optimization" in Section 2.1 Eq. (5): we train a model to predict the outcome of a set of highly related optimization problems. One key extension to the described amortized optimization setup is that the model W_θ estimates the Potts Model parameters from only the MSA query sequence $\boldsymbol{x}$ as input rather than the full MSA $\mathcal{M}(\boldsymbol{x})$. Thus, our model must learn to distill the protein energy landscape into its parameters, since it cannot look up related proteins during runtime. A full algorithm is given in Appendix A.

Similar to the original Potts model, we need to add a regularization penalty $\rho(W)$ to the main objective. For a finite sample of N different query sequences $\{\boldsymbol{x}_n\}$, and a sample of aligned sequences $\{\tilde{\boldsymbol{x}}_n^m\}$ from MSA $\hat{\mathcal{M}}(\boldsymbol{x}_n)$, the finite sample regularized loss, i.e. NPM training objective, becomes:

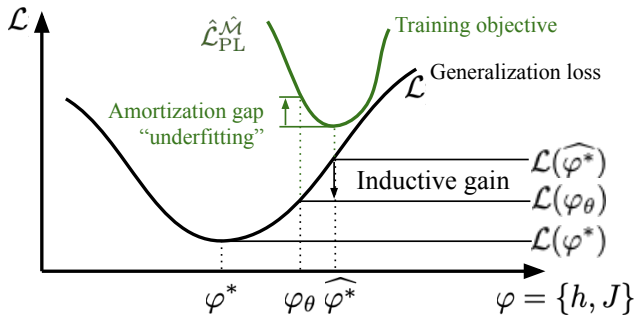

Figure 2: Inductive generalization gain. $\hat{W}^*$ is the standard Potts model, estimated on the finite observed MSA $\hat{\mathcal{M}}$. Though it minimizes the training objective, it does not achieve perfect generalization performance. However the Neural Potts Model $W_\theta(\boldsymbol{x})$ can generalize better than $\hat{W}^*$ through transfer learning from related samples, guided by the inductive bias of the model. We expect this especially when the estimate $\hat{W}^*$ is far from W^*, e.g. on small or biased MSAs.

$$\hat{\mathcal{L}}_{\text{NPM}}(\theta) = \sum_{n=1}^{N} \left[\frac{1}{M_{\text{eff}}(n)} \sum_{m} w_n^m \left[\ell_{\text{PL}}(\tilde{\boldsymbol{x}}_n^m; W_\theta(\boldsymbol{x}_n)) \right] + \rho(W_\theta(\boldsymbol{x}_n)) \right] \tag{8}$$

Inductive generalization gain (Fig. 2) is when the Neural Potts Model improves over the individual Potts model. Intuitively this is possible because the individual Potts Models are not perfect estimates (due to finite MSAs), while shared parameters can transfer information between related protein families.

Let us start with the normal amortized optimization setting, where we expect an amortization gap (Cremer et al., 2018). The amortization gap means that $W_\theta(x)$ will be behind the optimal W^* for the objective $\mathcal{L}$: $\mathcal{L}(W_\theta(x)) > \mathcal{L}(W^*)$. This is closely related to underfitting: the model W_θ is not flexible enough to capture $W^*(x)$. However, recall that in the Potts model setting, there is a finite-sample training objective $\hat{\mathcal{L}}$ (Eq. (8)), with minimizer $\hat{W}^*$. We can expect an amortization gap in the training objective; however this amortization gap can now be advantageous. Even if the amortized solution $W_\theta(\boldsymbol{x})$ is near-optimal on $\hat{\mathcal{L}}$, it can likely find a more generalizable region of the overparametrized domain W by parameter sharing of θ, allowing to transfer information between related instances. The inductive bias of $W_\theta(\boldsymbol{x})$ will allow the neural amortized estimate to generalize better, especially when the finite sample $\hat{\mathcal{M}}$ is poor. This inductive bias depends on the choice of model class for W_θ, as well as the shared structure between the protein families in the dataset. Concretely, when evaluating the generalization or validation loss $\mathcal{L}$ [1], we will show that for some samples $\mathcal{L}(W_\theta(\boldsymbol{x})) < \mathcal{L}(\hat{W}^*)$, i.e. there is an inductive generalization gain. This is visually represented in Fig. 2, and Table 1 compares between amortized optimization, NPM, and makes a connection to multi-task learning (Caruana, 1998). Additionally, we could frame NPM as a hypernetwork, a neural network that predicts the weights of second network (in this case the Potts model) as in, *e.g.*, Gomez & Schmidhuber (2005); Ha et al. (2016); Bertinetto et al. (2016).

In summary, the proposed NPM is a formulation to explicitly "distill" an ensemble of Potts models into a single feedforward model. From a self-supervised learning perspective, rather than supervising the model with the input directly, we use supervision from an energy landscape around the input.

4 Experiments

In Section 4.1 we present results on a small set of related protein domain families from PFAM, where we artificially purge sequences from a few families to study the inductive generalization gain from

[1] Validation loss $\mathcal{L}$ could refer to either the validation pseudo-likelihood loss (estimating the expectation $\mathbb{E}$ with heldout samples), or performance on downstream validation objectives like predicting contact which captures the underlying structure.

Table 1: Comparison between (A) "standard" amortized optimization, (B) Neural Potts Model, and (C) Multi-task learning. From row (A) amortized optimization to (B) Neural Potts Model, a finite-sample training loss is introduced which comes with considerations of generalization and regularization. This is related to multi-task learning, but with a major difference that (B) the solo optimization is over a single tensor W in the Potts model, but (C) a function f_θ in a learning problem. In the amortized/multi-task setting, the distribution over query sequences $\boldsymbol{x}$ in (B) NPM plays the role that different related tasks play in (C) MTL. In the NPM setting (B), W_θ takes $\boldsymbol{x}$ explicitly as argument, versus (C) MTL typically just has a separate output head per task.

	Solo objective Training	Solo objective Generalization	Amortized / Multi-task	Parametrization + model choices
(A) Optim→ Amortized	$\mathcal{L}(s; W)$	$\mathcal{L}(s; W)$ (= Training)	Amortized optim: $\mathbb{E}_{p(s)}\, \mathcal{L}(s; W_\theta(s))$	**Solo**: $W \in \mathbb{R}^n$ **Amor**: $W_\theta : \mathbb{R}^d \to \mathbb{R}^n$ +learner class
(B) Potts → NPM	PLL, finite MSA $\hat{M}$: $\hat{\mathcal{L}}(W) = \sum_m \ell_{\mathrm{PL}}(\tilde{\boldsymbol{x}}^m; W)$	Distr $\mathcal{L}(W) = \mathbb{E}[\ell_{\mathrm{PL}}(\tilde{\boldsymbol{x}}; W)]$ or Contact pred	Neural Potts $\mathbb{E}_{\boldsymbol{x}}[\hat{\mathcal{L}}^{\hat{M}}(W_\theta(\boldsymbol{x}))]$	**Solo**: $W \in \mathbb{R}^n$ +regularization **Amor**: $W_\theta : \mathbb{R}^d \to \mathbb{R}^n$ +learner class
(C) ML → MTL (Multi-task learning)	ERM: $\hat{\mathcal{L}}(f_\theta) = \sum_m \ell(f_\theta(x_m), y_m)$	$\mathcal{L}(f_\theta) = \mathbb{E}_{xy} \ell(f_\theta(x), y)$	Multi-task learning: $\sum_{t=1}^{T} [\hat{\mathcal{L}}^t(f_\theta^t)]$ for T related tasks	**Solo**: $f_\theta : \mathbb{R}^d \to \mathbb{R}$ +regularization +learner class **MTL**: $f_\theta^t : \mathbb{R}^d \to \mathbb{R}$ + param sharing f_θ^t

the related families. In Section 4.2 we present results on a large Transformer trained on MSAs for all of UniRef50.

For the main representation $g_\theta(\boldsymbol{x})$ we use a Transformer model (Vaswani et al., 2017). To compute the four-dimensional pairwise coupling tensor $J_\theta(\boldsymbol{x})$ from sequence embedding $g_\theta(\boldsymbol{x})$ we introduce the multi-head bilinear form in Appendix B. One can think of the multi-head bilinear form as the $L \times L$ self-attention maps of the Transformer's multi-head attention module but without softmax normalization. There are K^2 heads, one for every amino acid pair k, l. We discuss ways to symmetrize and tie the weights of those K^2 heads. We initialize $g_\theta(\boldsymbol{x})$ with a Transformer pre-trained with masked language modeling.

To evaluate Neural Potts Model energy landscapes, we will focus on proteins with structure in the Protein Data Bank (PDB), using the magnitude of the couplings after APC correction to rank contacts. The protocol is described in Appendix C.2.

4.1 PFAM P-LOOP-NTPASE CLAN

To study generalization in a controlled setting, we investigate a small set of structurally-related MSAs from the PFAM domain family database (El-Gebali et al., 2019). We expect that on a collection of related MSAs, information could be generalized to improve performance on low-depth MSAs. We select all 198 families in the P-loop NTPase clan. Families within a PFAM clan are linked by a distant evolutionary relationship, giving them related but not trivially-similar structure. We obtain contact maps for the sequences in each of the families where a structure is available in PDB. At test time we input the sequence and compare the generated couplings under the model to the corresponding structure.

We perform the experiment using a five-fold cross-evaluation scheme, in which we partition the 198 families into five equally-sized buckets of 40 families each. As in standard cross-validation, each bucket will eventually serve as an evaluation set. However, we do not remove the evaluation bucket, but artificially reduce the number of sequences in the MSAs in the validation bucket to a small fixed depth. MSAs in the remaining buckets remain unaltered. The goal of this setup is to check the model's ability to infer contacts on artificially limited sets of sequences. Both NPM and the baselines are evaluated according to this methodology, which guarantees that the Potts model

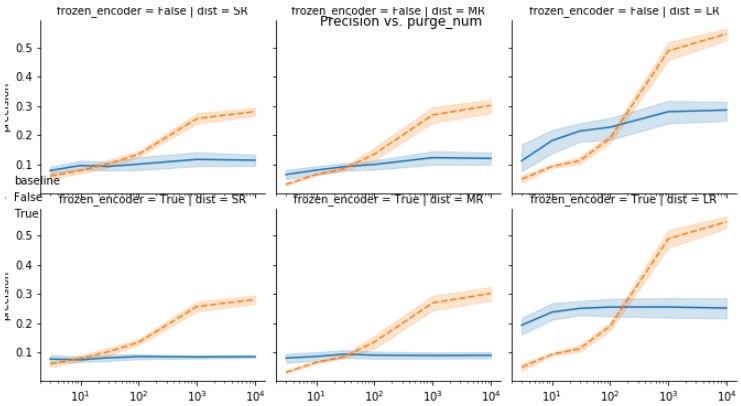

Figure 3: Contact prediction precision on validation bucket PFAM families at different levels of purging. Columns show (from left to right) short, medium and long-range precision for top-L/5 threshold. We show NPM with Transformer parameters either frozen or trained. In all cases, NPM outperforms the standard Potts model trained on the shallowest MSAs, but the effect is most pronounced for long range.

baseline and NPM have access to the same set of sequences. Note that while the Potts model uses the held-out purged MSAs directly at evaluation time, the NPM only has access to it during training. We perform five-fold cross validation, using each partition of the data as the evaluation-bucket, and report averages and standard deviations. We provide further details on model training in Appendix C.1 and on the PFAM dataset in Appendix C.3

Figure Fig. 6 shows the resulting contact prediction performance. We initialize a 12-layer Transformer from MLM pre-training, and consider either training the whole Transformer or only the multi-head bilinear form and single-site output head, i.e. keeping the Transformer parameters frozen. We empirically found that a multi-head bilinear form with weight tying was able to achieve relatively good precision in the first few 100 updates irrespective of purged MSA size, but saw performance degradation with continued training at high learning rates. With an untied head, NPM achieves stable training and outperforms a plain Potts model baseline for small MSA size. We see that NPM with frozen Transformer (lower capacity model) achieves more inductive gain for small purged MSAs, while when we train the full model (higher capacity model) the precision is better for large MSAs since it can fit the data better. For long range, NPM has the most pronounced advantage, outperforming plain CCMpred up until purged MSA size 100.

4.2 UniRef50

For a realistic evaluation setting for the NPM approach, we start from the UniRef50 (2018-03) database (Suzek et al., 2015), which provides full coverage of sequence space, clustered at 50% identity. We randomly partition the clusters in 90% train and 10% heldout sets. For each of the 27 million sequences in UniRef50 we compute the MSA, which is stored in compressed format on disk. During training, we iterate over all sequences and their MSAs on every epoch, and subsample to M=30 sequences per MSA. For contact prediction performance evaluation, the train and heldout split is preserved from NPM training. More details are found in Appendix C.4 (data), Appendix C.1 (model and training).

Figure 4 shows a comparison between the NPM predictions and individual Potts models fit from the MSA with CCMpred (Seemayer et al., 2014). The Neural Potts Model is given only the query sequence as input. On top-L/5 long range precision, NPM has better precision than CCMpred for 22.3% of train and 22.7% of heldout proteins. We indicate in red circles example proteins with low MSA-depth where NPM does better than the individual Potts model. These are visualized in Fig. 5. For shallow MSAs, the average performance of NPM is higher than the Potts model, suggesting an inductive generalization gain in this regime. On the heldout protein sequences, which were not inputs to the model during training, NPM is able to outperform CCMpred.

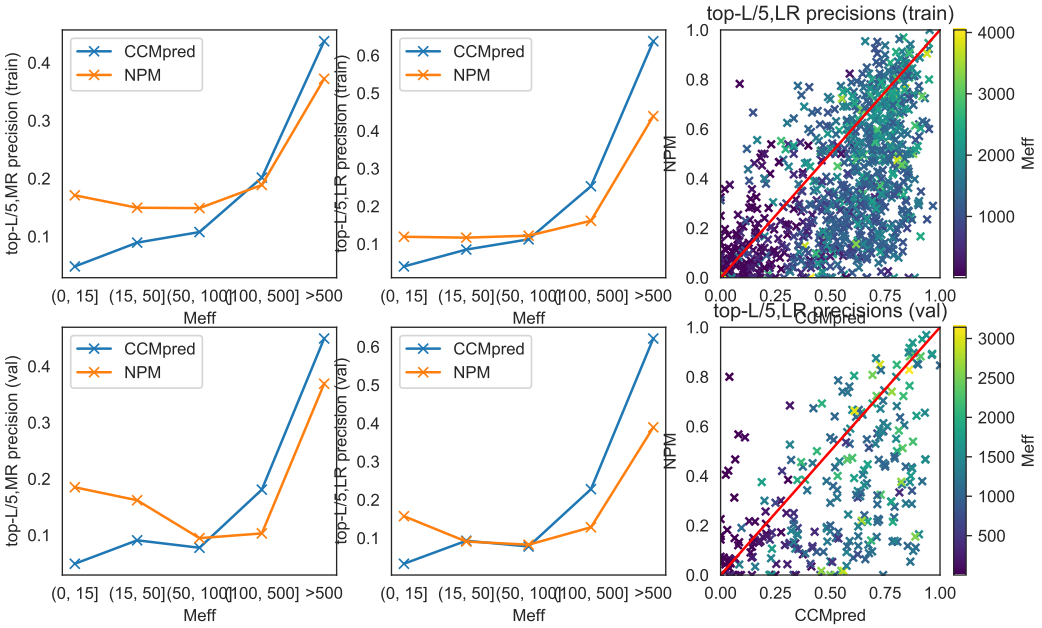

Figure 4: UniRef50: contact prediction precisions (higher is better) on medium range (left), long range (middle), binned by MSA size M_{eff}. Top row: sequences from the train set; bottom row: sequences from the held-out set. For shallow MSAs, average performance of the NPM is higher than the Potts model. (right) Scatter plot comparing long range precision from NPM vs standard Potts model (CCMpred), each point is a protein.

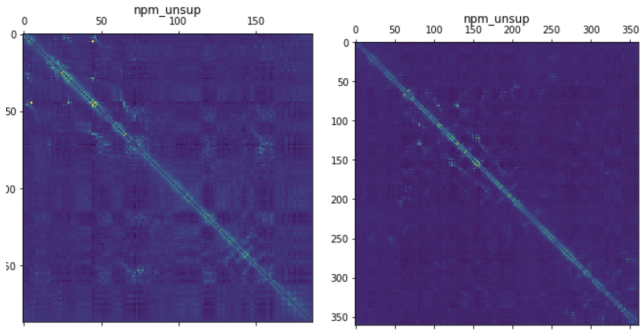

Figure 5: Example contact maps from the NPM model vs. the individually trained Potts models. NPM contact prediction (upper diagonal, black) compared to the standard Potts model prediction (lower diagonal, red). Ground truth contacts are indicated in gray. True and false hits are indicated with dots and crosses, respectively.

5 Discussion

This paper explores how a protein sequence model can be trained to produce a local energy landscape that is defined by a set of evolutionarily related sequences for each input. The training objective is cast as an amortized optimization problem. By learning to output the parameters for a Potts Model energy function across many sequences, the model may learn to generalize across the sequences. Extending self-supervision objectives for sequence modeling of proteins to richer sources of information that can be constructed from the underlying organization of the sequence space is a direction with many further possibilities.

We also formally and empirically investigate the generalization capability of models trained through amortized optimization. We consider the setting of training independent Potts models on the MSA of each sequence, in comparison with training a single model using the amortized objective to predict Potts model parameters for many inputs. Empirically the amortized objective appears to provide an inductive gain when few related sequences are available in the MSA for training the independent Potts Model.

REFERENCES

Ethan C Alley, Grigory Khimulya, Surojit Biswas, Mohammed AlQuraishi, and George M Church. Unified rational protein engineering with sequence-only deep representation learning. *bioRxiv*, pp. 589333, 2019.

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. *arXiv preprint arXiv:1606.04474*, 2016.

Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, pp. 103–129, 1995.

Sivaraman Balakrishnan, Hetunandan Kamisetty, Jaime G Carbonell, Su-In Lee, and Christopher James Langmead. Learning generative models for protein fold families. *Proteins: Structure, Function, and Bioinformatics*, 79(4):1061–1078, 2011.

Alex Bateman, Lachlan Coin, Richard Durbin, Robert D Finn, Volker Hollich, Sam Griffiths-Jones, Ajay Khanna, Mhairi Marshall, Simon Moxon, Erik LL Sonnhammer, et al. The pfam protein families database. *Nucleic acids research*, 32(suppl_1):D138–D141, 2004.

Luca Bertinetto, João F Henriques, Jack Valmadre, Philip Torr, and Andrea Vedaldi. Learning feed-forward one-shot learners. In *Advances in neural information processing systems*, pp. 523–531, 2016.

Rich Caruana. Multitask learning. In *Learning to learn*, pp. 95–133. Springer, 1998.

Tian Qi Chen and Mark Schmidt. Fast patch-based style transfer of arbitrary style. *arXiv preprint arXiv:1612.04337*, 2016.

Chris Cremer, Xuechen Li, and David Duvenaud. Inference suboptimality in variational autoencoders. *arXiv:1801.03558*, 2018.

Thomas Duriez, Steven L Brunton, and Bernd R Noack. *Machine learning control-taming nonlinear dynamics and turbulence*, volume 116. Springer, 2017.

Magnus Ekeberg, Cecilia Lövkvist, Yueheng Lan, Martin Weigt, and Erik Aurell. Improved contact prediction in proteins: using pseudolikelihoods to infer potts models. *Physical Review E*, 87(1):012707, 2013.

Sara El-Gebali, Jaina Mistry, Alex Bateman, Sean R Eddy, Aurélien Luciani, Simon C Potter, Matloob Qureshi, Lorna J Richardson, Gustavo A Salazar, Alfredo Smart, et al. The pfam protein families database in 2019. *Nucleic acids research*, 47(D1):D427–D432, 2019.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

Ulrike Göbel, Chris Sander, Reinhard Schneider, and Alfonso Valencia. Correlated mutations and residue contacts in proteins. *Proteins: Structure, Function, and Bioinformatics*, 18(4):309–317, 1994.

Faustino Gomez and Jürgen Schmidhuber. Evolving modular fast-weight networks for control. In *International Conference on Artificial Neural Networks*, pp. 383–389. Springer, 2005.

Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 399–406, 2010.

David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv:1609.09106*, 2016.

Michael Heinzinger, Ahmed Elnaggar, Yu Wang, Christian Dallago, Dmitrii Nechaev, Florian Matthes, and Burkhard Rost. Modeling aspects of the language of life through transfer-learning protein sequences. *BMC bioinformatics*, 20(1):723, 2019.

David T Jones, Daniel WA Buchan, Domenico Cozzetto, and Massimiliano Pontil. Psicov: precise structural contact prediction using sparse inverse covariance estimation on large multiple sequence alignments. *Bioinformatics*, 28(2):184–190, 2011.

Hetunandan Kamisetty, Sergey Ovchinnikov, and David Baker. Assessing the utility of coevolution-based residue–residue contact predictions in a sequence- and structure-rich era. *Proceedings of the National Academy of Sciences*, 110(39):15674–15679, 2013. ISSN 0027-8424. doi: 10.1073/pnas.1314045110. URL https://www.pnas.org/content/110/39/15674.

Yoon Kim, Sam Wiseman, Andrew C Miller, David Sontag, and Alexander M Rush. Semi-amortized variational autoencoders. *arXiv:1802.02550*, 2018.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *ICLR*, 2013.

Ke Li and Jitendra Malik. Learning to optimize. *arXiv:1606.01885*, 2016.

Joseph Marino, Yisong Yue, and Stephan Mandt. Iterative amortized inference. *arXiv:1807.09356*, 2018.

Debora S Marks, Lucy J Colwell, Robert Sheridan, Thomas A Hopf, Andrea Pagnani, Riccardo Zecchina, and Chris Sander. Protein 3d structure computed from evolutionary sequence variation. *PloS one*, 6(12), 2011.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *arXiv:1707.03141*, 2017.

Faruck Morcos, Andrea Pagnani, Bryan Lunt, Arianna Bertolino, Debora S Marks, Chris Sander, Riccardo Zecchina, José N Onuchic, Terence Hwa, and Martin Weigt. Direct-coupling analysis of residue coevolution captures native contacts across many protein families. *Proceedings of the National Academy of Sciences*, 108 (49):E1293–E1301, 2011.

Roshan Rao, Nicholas Bhattacharya, Neil Thomas, Yan Duan, Xi Chen, John Canny, Pieter Abbeel, and Yun S Song. Evaluating protein transfer learning with tape. *arXiv:1906.08230*, 2019.

Nathan Ratliff, J Andrew Bagnell, and Siddhartha S Srinivasa. Imitation learning for locomotion and manipulation. In *2007 7th IEEE-RAS International Conference on Humanoid Robots*, pp. 392–397. IEEE, 2007.

Danilo J Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1278–1286, 2014.

Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Demi Guo, Myle Ott, C Lawrence Zitnick, Jerry Ma, and Rob Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *bioRxiv*, pp. 622803, 2019. URL https://doi.org/10.1101/622803.

Stefan Seemayer, Markus Gruber, and Johannes Söding. Ccmpred—fast and precise prediction of protein residue–residue contacts from correlated mutations. *Bioinformatics*, 30(21):3128–3130, 2014.

Rui Shu. Amortized optimization. *http://ruishu.io/2017/11/07/amortized-optimization*, 2017.

Martin Steinegger, Markus Meier, Milot Mirdita, Harald Vöhringer, Stephan J Haunsberger, and Johannes Söding. Hh-suite3 for fast remote homology detection and deep protein annotation. *BMC bioinformatics*, 20 (1):1–15, 2019.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Baris E. Suzek, Hongzhan Huang, Peter McGarvey, Raja Mazumder, and Cathy H. Wu. UniRef: Comprehensive and non-redundant UniProt reference clusters. *Bioinformatics*, 23(10):1282–1288, 5 2007. ISSN 13674803. doi: 10.1093/bioinformatics/btm098. URL http://www.uniprot.org.

Baris E Suzek, Yuqi Wang, Hongzhan Huang, Peter B McGarvey, Cathy H Wu, and UniProt Consortium. Uniref clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics*, 31(6):926–932, 2015.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017.

Martin Weigt, Robert A White, Hendrik Szurmant, James A Hoch, and Terence 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.

Appendix

A Learning the Neural Potts Model

Algorithm 1 Learning the Neural Potts Model

Initialize the amortized model W_θ
while unconverged **do**
  Sample sequences $\boldsymbol{x}_n \sim \mathcal{D}$ and MSA sequences $\tilde{\boldsymbol{x}}_n^m \sim \mathcal{M}(\boldsymbol{x}_n)$
  grad_update$(\theta, \nabla_\theta \hat{\mathcal{L}}_{\text{NPM}}(\theta; \{\boldsymbol{x}_n\}, \{\tilde{\boldsymbol{x}}_n^m\})$ ▷ Optimize the regularized loss in Eq. (8)
end while

B Model Architecture: Multi-head Bilinear Form for Pairwise Couplings

In this Section, we describe the model architecture to compute a four-dimensional pairwise coupling tensor $J_\theta(\boldsymbol{x})$ from sequence embedding $g_\theta(\boldsymbol{x})$.

B.1 Multi-head Bilinear Form

We write sequence length L and amino acid vocabulary $K = 21$. The single site potentials $h \in \mathbb{R}^{L\times K}$, and the pairwise couplings are a four-dimensional tensor: $J \in \mathbb{R}^{L\times K\times L\times K}$ indexed as $J_{ij}(k, l)$.

We start with a sequence-level model to produce the embedding e of the sequence (typically final hidden layer output): $e = g_\theta(\boldsymbol{x}) \in \mathbb{R}^{L\times d}$. The estimator for single-site potential $h_\theta(\boldsymbol{x})$ is a linear projection layer on the embedding; $h_\theta(\boldsymbol{x}) = g_\theta(\boldsymbol{x})W^h$ with $W^h \in \mathbb{R}^{d\times K}$. Now we discuss how to parametrize the estimator $J_\theta(\boldsymbol{x}) \in \mathbb{R}^{L\times K\times L\times K}$.

We introduce a *multi-head bilinear form* on the embedding e; i.e. for every pair k, l of amino acids we have a bilinear form, parametrized with a learned interaction matrix $W^{kl} \in \mathbb{R}^{d\times d}$ connecting the hidden states at positions $e_i, e_j \in \mathbb{R}^{1\times d}$. So we compute the K^2 bilinear forms for amino acid pairs (k, l) between $L \times L$ position pairs (i, j): $J_{ij}(k, l) = e_i W^{kl} e_j^\top$. We use a low-rank decomposition $W^{kl} = U^{kl}V^{kl\top}$ with both $U^{kl}, V^{kl} \in \mathbb{R}^{d\times d'}$, so the bilinear form becomes the inner product in the lower-dimensional space with d' the projection dimension: $(e_i U^{kl})(e_j V^{kl})^\top$. We can interpret this as an inner product of embeddings i, j after linear projection to a space specific to amino acid pair (k, l). This low-rank multi-head bilinear form is similar to the multi-head attention mechanism introduced in Vaswani et al. (2017), but without softmax normalization.

Notation-wise, our parameters θ include the parameters of the transformer that produces the embedding and the components of the interaction matrix $\{U^{kl}, V^{kl}\}$.

Tied projection One way to reduce the number of parameters in the multi-head bilinear form, is for the low-rank decompositions of the K^2 heads W^{kl} to share their decomposition per k, l. Instead we can *share/tie* the projection matrices between amino acids k, l: $U^{kl} = U^k$ and $V^{kl} = V^l$, such that head $W^{kl} = U^k V^{l\top}$. Note that the dot product in this case is after a linear projection specific to single-site amino acid k and l separately; $J_{ij}(k, l) = e_i W^{kl} e_j^\top = (e_i U^k)(e_j V^l)^\top$.

Symmetry we can/should parametrize the estimator of J to be symmetrical against interchanging both i,j and k,l: $J_{ij}(k, l) = J_{ji}(l, k)$, i.e. no difference between the order of considering interaction between AA k at position i with AA l at position j. This does not mean symmetry of each interaction matrix! We ask that

$$J_{ij}(k, l) = e_i W^{kl} e_j^\top = e_j W^{lk} e_i^\top = e_i W^{lk\top} e_j$$

The second equality is the symmetry, the last equality by transposing the bilinear form. From $W^{kl} = W^{lk\top}$ it follows that

$$U^{kl}V^{kl\top} = V^{lk}U^{lk\top}$$
$$U^{kl} = V^{lk}$$

The last equality the obvious choice. In the tied parametrization, this simply becomes $U^{kl} = U^k = V^k = V^{lk}$ such that $W_{kl} = U^k U^{l\top}$. Once again, note that the dot product now becomes $J_{ij}(k,l) = (e_i U^k)(e_j U^l)^\top$. We present a Tensor decomposition perspective on this multi-head bilinear form in Appendix B.2.

B.2 Tensor decomposition view on multi-head bilinear form

We can see the multi-head bilinear form as a tensor decomposition of J, for which we will use Einstein notation to indicate that any pair of indices appearing both in subscript and superscript are summed over their range. Let us write the tensor collecting the U^{kl} matrices as $\mathcal{U} \in \mathbb{R}^{K\times K\times d\times d'}$; and index into $\mathcal{U}$ in the same notation as for U: $\mathcal{U}^{kl}_{\alpha r} = U^{kl}_{\alpha r}$. With $\alpha, \beta \in [1\dots d]$, $r \in [1\dots d']$ The same for $\mathcal{V}$. Now the J estimate in the full untied asymmetric case, written as tensor, becomes

$$J^{kl}_{ij} = e^\alpha_i \, \mathcal{U}^{kl}_{\alpha r} \, \mathcal{V}^{klr}_\beta \, e^\beta_j$$

or the symmetric ($\mathcal{U}^{kl}_{\alpha r} = \mathcal{V}^{lk}_{\alpha r}$) and tied ($\mathcal{U} \in \mathbb{R}^{K\times d\times d'}$) version:

$$J^{kl}_{ij} = e^\alpha_i \, \mathcal{U}^{k}_{\alpha r} \, \mathcal{U}^{lr}_\beta \, e^\beta_j$$

Note that the $\mathcal{U}, \mathcal{V}$ are shared across proteins, while the embeddings $e = g_\theta(\boldsymbol{\sigma})$ are specific per protein, based on a high-capacity sequence level model.

C Experiment Details

C.1 Training details

We summarize the precise model architecture and optimization settings in this table:

		PFAM	UniRef50
Transformer model size	Parameters	87M (X trained)	519M
	Layers	12 layers	32 layers
	Embed dim	768	1024
	Attention heads	12	16
	mhbf tied	Yes	No
	mhbf projection dim	128	512
MLM pre-training	Number of epochs (UR50)	21 epochs / X updates	50 epochs / x updates
	Learningrate	"4e-4"	"3e-5"
NPM training	Number of updates	7.5k	135k
	Adam learningrate	1e-5/3e-5	"3e-5"
	λ_J, λ_h	1e-3, 1e-2	1e-3, 1e-2
	M (samples per MSA)	100	30
	Batch size (at max-len=512)	64	2048
	Base model frozen	Yes	No

Potts model fitting is done with MSA's sampled down to 100 sequences. Each sequence is selected according to its sequence weight w^m. In both the PFAM and Uniref experiments, we enforce a max sequence length of 512 via random contiguous crops of positions.

C.2 Validation details

To compute precisions, we convert the pairwise couplings $J \in \mathbb{R}^{L\times K\times L\times K}$ to an $L \times L$ pairwise coupling score by by (1) zeroing all positions in J corresponding to gap characters, (2) computing the "magnitude" via Frobenius norm over the $K \times K$ matrix J_{ij} for every pair of positions i, j, and (3) zeroing the diagonal of the final matrix.

C.3 PFAM training data and setup

Data Selection. For the PFAM experiments in 4.1, we retrieved the PFAM database (El-Gebali et al., 2019) version 28.0. All MSAs in the P-loop NTPase clan, (n=198) clan were parsed from the multiple

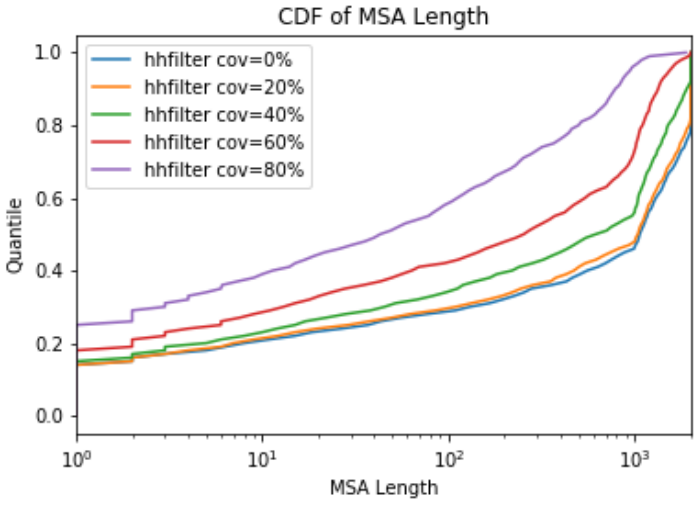

Figure 6: A random sample of 1000 MSAs were analyzed from our Uniref50 MSA dataset, containing 27M total MSAs. Graph shows cumulative density plots with MSA depth in log-scale on the x axis, for different coverage requirements specified by a call hhfilter, applied to each MSA. The percentage of MSAs with depth < 100 spans from 25% - 60%, depending on the desired coverage parameter. The prevalence of such shallow MSAs on this very large scale dataset, which spans evolutionary diversity, motivates the need for a method that can make use of shallow MSAs.

alignment file Pfam-A.full. PFAM clans are collections of evolutionarily related families with similar, but not trivially-similar structure. We chose the P-loop-NTPase clan, as we found that the largest clan in PFAM, HTH, had MSAs which yielded poor long-range contact predictions on the CCMPred baseline.

For contact precision evaluation, we use the pdbmap file included in the PFAM28.0 release. This file contains associations between proteins in PFAM families and PDB files, when they exist..

Dataset splits. MSAs in the P-loop NTPase clan are mostly deep (90% > 2000 sequences). To provide opportunity for observing inductive generalization gain, we partititon the dataset into 5 equally-sized buckets of MSA's, and purge down the number of sequences in the fifth MSA bucket, similar to holding out a bucket during cross-validation. Like cross-validation, we rotate over which bucket is purged, plot results as an average with single standard deviation error bars.

MSA Processing. For perfomance reasons, we load the first 10,000 sequences from each PFAM MSA in the P-loop NTPase clan. Then, all MSA's in our setup go through a common set of transforms. HHfilter from the HHSuite3 (Steinegger et al., 2019) toolset is applied to each MSA with default settings. We found filtering to greatly improve contact prediction accuracy of our baseline, CCMPred.

Full multiple sequence alignments between positions in all sequences in a PFAM family necessarily include many gap positions. To avoid giving NPM sequence with a very high gap fraction, at training time, we randomly select a sequence in an MSA, and drop all MSA columns in positions where that sequence has gaps. To ensure sequences stay constrained in length, if an MSA has more than 512 positions, we take a random contiguous crop of columns. Finally, we subsample the MSA down to 100 sequences according to sequence weights w^m.

At test time, we iterate through all MSA's in the purge-bucket that have have a contact map, and recenter them to the contact map's associated sequence. At this stage, sequence length can vary greatly. For simplicity, if a (seq, MSA) pair has sequence dimension greater than 512, we take a random 512-length, contiguous crop of positions/columns in the (seq, MSA) pair. Finally, all MSA's are randomly subsampled to 100 sequences according to sequence weights.

C.4 UNIREF50 TRAINING DATA AND SETUP

For the experiments in Section 4.2, we retrieved the UniRef-50 (Suzek et al., 2015) database dated 2018-03 (27M sequences) The UniRef50 clusters were partitioned randomly in 90% train and 10% heldout. For all sequences, we construct the MSAs using HHblits (Steinegger et al., 2019) against the UniClust30_2017_10 database. HHblits was performed for 3 iterations, with an e-value of 0.001.

It's important to note that given this MSA generation procedure, validation sequences can be included in MSAs of train sequences. However, we are are guaranteed that validation sequences are not trained on as input to NPM.

Contact precision targets In order to show out-of-distribution generalization on contacts, we are motivated to create a dataset where the training and validation set splits also respect the pretraining data splits. First we query PDB to obtain a list of all protein structures submitted with a resolution of at most 2.5 Å, a length greater than 40 residues, and submitted before May 1, 2020.

To be sure that we respect pre-training splits, we search each PDB sequence against UniRef50 using the MMSeqs2 software suite. We run the `easy-search` routine with a minimum sequence identity of 50% at 80% coverage - using the target coverage mode, meaning that 80% of the UniRef50 sequence must be more than 50% identical to be counted as a hit. This labels each sequence in the structural dataset with a list of UniRef50 hits. A protein structure is included in the training set only if the list of hits do not include any sequences in the UniRef50 validation set. Similarly, a structure is included in the validation set if the list of hits do not include any sequences in the training set. If we detect hits in both UniRef50 training and validation sets, these structures are discarded in the resulting dataset. We then cluster the datasets at 50% sequence similarity, resulting in a dataset of 11040 training structures and 211 validation structures.

To predict contact on these structures, NPM only needs the corresponding primary sequence. For the plain Potts model baseline, we construct an MSA for the structures by searching Uniclust30 (2017) with HHblits with n=3 iterations and the default E-value cutoff 0.001.