Metalic: Meta-Learning In-Context with Protein Language Models

Jacob Beck*, **Shikha Surana, Manus McAuliffe, Oliver Bent, Thomas D. Barrett, Juan Jose Garau Luis, & Paul Duckworth**
InstaDeep
Boston, MA, USA & London, UK
*Jacob_Beck@alumni.brown.edu, FirstInitial.LastName@instadeep.com

Abstract

Predicting the biophysical and functional properties of proteins is essential for *in silico* protein design. Machine learning has emerged as a promising technique for learning such prediction tasks. However, the relative scarcity of *in vitro* annotations means that these models often have little, or no, specific data on the desired fitness prediction task. As a result of limited data, protein language models (PLMs) are typically trained on general protein sequence modeling tasks, and then fine-tuned, or applied zero-shot, to protein fitness prediction. When no task data is available, the models make strong assumptions about the correlation between the protein sequence likelihood and fitness scores. In contrast, instead of restricting the representations, we propose meta-learning over a distribution of standard fitness prediction tasks, and demonstrate positive transfer to unseen fitness prediction tasks. Our method, called *Metalic* (Meta-Learning In-Context), makes use of in-context learning and fine-tuning, when data is available, to adapt to new tasks. Crucially, the fine-tuning enables considerable generalization, even though it is not accounted for during meta-training. The fine-tuned models achieve strong results with 18 times fewer parameters than state-of-the-art models. Moreover, our method sets a new state-of-the-art on ProteinGym, an established fitness-prediction benchmark. We believe that meta-learning across protein fitness tasks will play a vital role in advancing protein fitness prediction methods.

1 Introduction

The accurate prediction of the functional and biophysical properties of proteins, such as stability or binding affinity, is a critical challenge in the physical sciences with far-reaching implications for drug discovery, medical research and agriculture. Protein design seeks to optimize one, or a collection of these properties, referred to as fitness. While protein fitness can be measured *in vitro*, the process is often laborious and time-consuming. Consequently, machine learning models have emerged as a powerful tool to accelerate protein design by predicting fitness directly from amino acid sequences *in silico*. However, due to the complex interplay between protein sequences and fitness, and the limited availability of high-quality data, accurate prediction is a challenging task.

In recent literature, protein language models (PLMs) are commonly used to model protein fitness [Madani et al., 2020, Rives et al., 2021, Rao et al., 2021, Lin et al., 2022, Notin et al., 2023, Truong Jr and Bepler, 2024]. While PLMs are not directly trained to predict fitness, they are trained to model the likelihood of naturally occurring proteins, which is assumed to correlate strongly with their fitness. By predicting masked amino acids, or subsequent amino acids, over known proteins at scale, PLMs can capture much of the structure and resultant properties deriving from the amino acid sequence. In practice, PLMs are further fine-tuned on protein fitness data. While this paradigm is highly effective,

Foundation Models for Science Workshop,38th Conference on Neural Information Processing Systems (NeurIPS 2024).

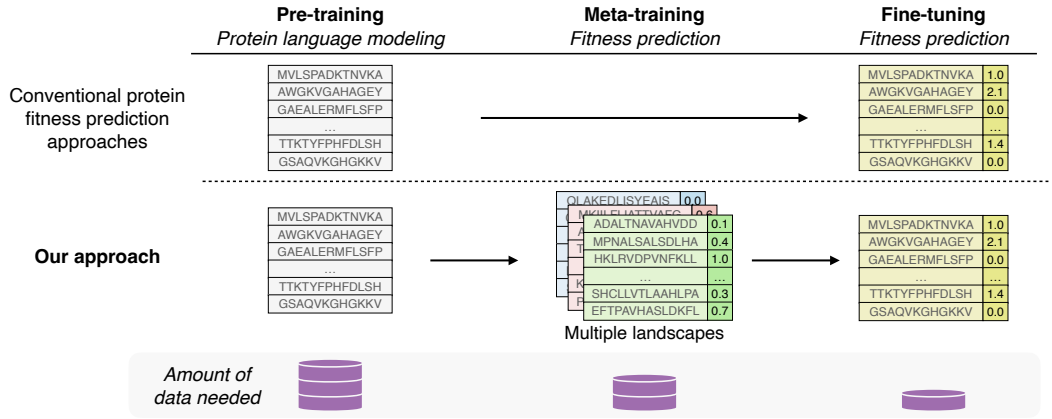


Figure 1: An overview of meta-learning paradigm for protein fitness prediction. PLMs are trained over massive quantities of unlabeled data, whereas meta-learning is trained over a smaller quantity of labelled fitness data. Using this extra data is critical given the limited data for fine-tuning at test time.

when there is severely limited data, or no data, as in *de novo* design, PLMs provide limited utility. For example, in order to use a PLM when there is no data available, it must be assumed that the protein fitness is solely a function of protein likelihood [Truong Jr and Bepler, 2024, Hawkins-Hooker et al., 2024], and for masked language models, it may also be assumed that each amino acid contributes independently to the fitness [Meier et al., 2021, Hawkins-Hooker et al., 2024, Notin et al., 2024].

While limited or no data may be available for specific protein fitness prediction tasks, there are often other related protein prediction tasks that can serve as a valuable source of additional data. Advances in high-throughput assays, such as deep mutational scanning, have enabled the compilation of large datasets containing over one hundred distinct fitness prediction tasks [Notin et al., 2024]. Training over a distribution of related tasks, to reason about new tasks, is a process called *meta-learning*.

To address the challenge of limited data in protein fitness prediction, we propose *Metalic*, a method that integrates in-context meta-learning, protein language models, and fine-tuning. Meta-learning, or learning to learn, aims to improve the learning process by leveraging experience from other learning problems. *Metalic* makes use of PLMs and fine-tuning, but adds an additional step of meta-learning over additional protein tasks. The additional meta-learning phase is depicted in Fig. 1, and is critical given the limited protein fitness data available in each task for inference at test-time. Meta-learning shifts the burden of handcrafting assumptions and inductive biases for our algorithms to collecting additional tasks for learning [Beck et al., 2023]. In the setting of protein fitness design, for example, when no data is available for fine-tuning, meta-learning over a distribution of tasks enables us to forgo the assumption that fitness is solely a function of protein likelihood. Instead, *Metalic* learns the relationship between protein embeddings and fitness through meta-learning, setting a new state-of-the-art (SOTA) on the ProteinGym benchmark [Notin et al., 2024].

We present *Metalic*, an in-context meta-learning approach to tackle the problem of protein fitness prediction, and make the following contributions:

- We introduce a method that combines in-context meta-learning with PLMs and fine-tuning for protein fitness prediction.
- We advance SOTA for zero-shot protein fitness prediction on the ProteinGym benchmark.
- We attain strong performance for few-shot protein fitness prediction with 18 times fewer parameters.
- We ablate each component of our method to understand the contributions of each part and underscore their necessity.
- We empirically validate the superiority of our method to alternative forms of meta-learning.

2 Related Work

Meta-Learning Meta-learning aims to create a sample-efficient learning algorithm by training over a distribution of tasks. The goal is to learn algorithms such that they can rapidly adapt to new tasks during inference. This inference-time adaptation is often called the *inner loop*, for which there are

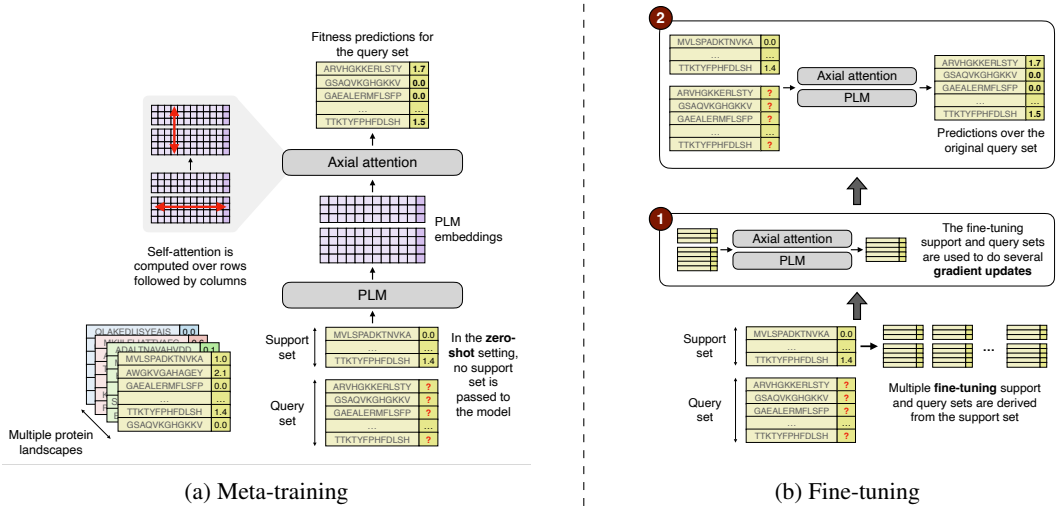
two primary forms found in the literature: gradient-based meta-learning [Finn et al., 2017, Zintgraf et al., 2019] and in-context meta-learning [Santoro et al., 2016, Mishra et al., 2017, Nguyen and Grover, 2022].

Gradient-based algorithms and in-context algorithms differ both in their computational efficiency and capacity for out-of-distribution generalization. Gradient-based approaches explicitly adapt model parameters within in the inner loop using standard gradient-based learning. Commonly, the model initialization are the only parameters learned over the task distribution. The meta-training process explicitly computes gradients through this adaptation process, effectively learning a parameter initialization that can be adapted to new tasks with only a few gradient steps at inference [Finn et al., 2017, Zintgraf et al., 2019, Vuorio et al., 2019]. However, this comes with considerable computational overhead – in particular when calculating meta-gradients through the inner loop process – which makes gradient-based meta-learning less suitable for large models. Alternatively, in-context meta-learning adapts at inference time by conditioning on a task-specific dataset in context, i.e., conditioning on the data points over which gradient based approaches would train. Typically, this is achieved by modelling the inner loop with a sequence model that can explicitly condition on the provided data [Santoro et al., 2016, Mishra et al., 2017, Ni et al., 2022, Nguyen and Grover, 2022, Beck et al., 2024a,b]. Such methods have been found to typically be more sample- and compute-efficient than gradient-based adaption, but can perform worse when provided with out-of-distribution tasks given the lack of explicit gradient-based learning in the inner-loop [Beck et al., 2023].

In this paper, we only train for in-context meta-learning, but find that this is still compatible with task-specific fine-tuning at inference. In the the meta-reinforcement learning setting, this combination has been shown to be possible by increasing task-specific data for fine-tuning [Xiong et al., 2021]. In contrast, we evaluate in the supervised setting and do not give increased data at inference time. While prior work has combined gradient-based and in-context meta-learning without restrictions [Vuorio et al., 2019, Rusu et al., 2018], these works compute expensive meta-gradients to learn how to account for fine-tuning. Despite not explicitly meta-learning gradient-based adaptation, we find that in-context meta-learning alone provides a strong foundation for subsequent fine-tuning and that both aspects are critical for achieving high performance.

Likelihood-Based Fitness Prediction with PLMs Leveraging PLMs is standard practice in protein fitness prediction [Rives et al., 2021, Notin et al., 2023, Rao et al., 2021, Truong Jr and Bepler, 2024]. In the few-shot setting, PLMs intended for sequence generation are repurposed by fine-tuning for protein fitness prediction [Rives et al., 2021]. In the zero-shot setting, it is assumed that the fitness correlates with the likelihood of the proteins associated sequence of amino acids, as predicted by a PLM [Meier et al., 2021, Truong Jr and Bepler, 2024]. Furthermore, if using a masked PLM, it is often assumed that each amino acid contributes independently to the fitness [Meier et al., 2021]. In this work, we likewise leverage PLMs for protein fitness prediction. However, in contrast, we make use of additional data in the form of additional fitness prediction tasks on other proteins. Specifically, we meta-learn how to use a PLM for protein fitness prediction, rather than relying on assumptions. Only after meta-learning, do we fine-tune our model. Leveraging meta-learning over multiple tasks enables us to avoid restrictive constraints on the model representation and achieve SOTA performance. While multi-task learning has been leveraged previously for protein fitness prediction [Xu et al., 2022], that work did not leverage any sort of meta-learning, and had limited success. We will show that in-context meta-learning is necessary in order to achieve strong results.

In-Context PLMs We build upon existing PLMs that make use of in-context data for protein fitness prediction [Notin et al., 2022, Truong Jr and Bepler, 2024, Notin et al., 2023, Rao et al., 2021]. However, these methods do not *meta-learn* how to make use of their context. These methods either learn to use the context only for protein language modelling, and then assume that the likelihood from the generative model correlates with fitness [Truong Jr and Bepler, 2024, Notin et al., 2022, Rao et al., 2021], or these models make use of the context with fitness information for protein fitness prediction, but not by meta-learning over multiple protein tasks [Notin et al., 2023]. Of these ProteinNPT [Notin et al., 2023] is the most related to our method, since we use the same attention architecture to condition on fitness information about related proteins in-context, and it makes use of gradient steps to fine-tune to the target task. In comparison, our method meta-learns over many more tasks how to make use of the fitness information, which we find to be critical (Section 4). Additionally, our method is the first to make use of the aforementioned procedure to allow fine-tuning and in-context conditioning on the very same context at inference time.



(a) Meta-training

(b) Fine-tuning

Figure 2: An overview of *Metalic*. Meta-training our model over many protein prediction tasks enables in-context learning (left (a)). Fine-tuning the in-context learning on the support set requires sub-sampling smaller support and query sets, and enables generalization at test time (right (b)).

3 Methods

3.1 Problem Setting

We consider a fitness prediction task, $\mathcal{T}$, to be defined by a dataset of the form $\mathcal{D}_\mathcal{T} = \{(x_i, y_i \equiv f_\mathcal{T}(x_i))\}_{i=1}^{N}$, where x_i is a sequence of amino acids, and $y_i \in \mathbb{R}$ is the associated scalar fitness value assigned by the (unknown) underlying fitness function $f_\mathcal{T}$. In the few-shot setting, the task-specific data is typically split into non-overlapping support and query sets: $\mathcal{D}_\mathcal{T}^{(\mathrm{S})} = \{(x_i^{(\mathrm{S})}, y_i^{(\mathrm{S})})\}_{i=1}^{N^{(\mathrm{S})}}$ and $\mathcal{D}_\mathcal{T}^{(\mathrm{Q})} = \{(x_i^{(\mathrm{Q})}, y_i^{(\mathrm{Q})})\}_{i=1}^{N^{(\mathrm{Q})}}$ such that $\mathcal{D}_\mathcal{T}^{(\mathrm{S})} \cap \mathcal{D}_\mathcal{T}^{(\mathrm{Q})} = \emptyset$. The support set provides data for task-specific adaptation, and the query set provides data for evaluating the adapted performance. The size of the support set is referred to as the *shot*. Note that even when the support set is empty (*zero-shot*), the model can still adapt to the task using information from the query set, i.e., other sequences in the protein landscape that do not have a provided fitness score.

Meta-learning for protein fitness prediction requires not just a single task, but multiple tasks, $\mathcal{D} = \mathcal{D}_1, \ldots \mathcal{D}_\mathrm{T}$ over which to learn. The full dataset of tasks, $\mathcal{D}$, can be seen as defining a distribution of tasks which can be split into training and test tasks in the usual way. Concretely, for in-context meta-learning the goal is to learn a function with parameters θ conditioned on the full support set and unlabelled inputs from query set, $f_\theta(\{x_i^{(\mathrm{Q})}\}_{i=1}^{N^{(\mathrm{Q})}}, \mathcal{D}_\mathcal{T}^{(\mathrm{S})})$. In our case, rather than directly predicting the fitness values the query set, we instead follow prior works that use a preference-based objective that aims to correctly rank the query set in order of fitness [Krause et al., 2022, Brookes et al., 2023, Hawkins-Hooker et al., 2024].

3.2 Metalic

Architecture Our work leverages the ProteinNPT architecture proposed by Notin et al. [2023] as an in-context PLM for fitness prediction. Here we briefly summarise the key elements as illustrated in Fig. 2, but defer the reader to Appendix A.3 and the original paper for full details.

Protein sequences in both the support and query set are converted to per-residue embeddings using a pre-trained PLM; in our case we take the third layer of ESM2-8M [Lin et al., 2022]. Fitness scores in the query set are set to zero. All fitness scores are then projected to match the dimension of the residue embedding with a simple linear layer, summed with a learned embedding denoting if the row belongs to the support or query set, and finally concatenated along the sequence dimension. This tensor is then processed via axial attention blocks [Ho et al., 2019]., each of which applies self-attention separately along the sequences and across the sequences. Axial attention reduces the computational complexity of self-attention from $O(K^2L^2)$ to $O(K^2 + L^2)$, where K is the shot and L is the length

of a protein. Finally, a linear layer conditions on the fitness embedding and mean-pooled sequence embedding of to predict each query value, i.e. $\{v_i^{(\mathrm{Q})}\}_{i=1}^{N^{(\mathrm{Q})}} = f_\theta(\{x_i^{(\mathrm{Q})}\}_{i=1}^{N^{(\mathrm{Q})}}, \mathcal{D}_\mathcal{T}^{(\mathrm{S})})$, which is then used to rank them by fitness.

Meta-Training Following prior works that use a preference-based objective [Krause et al., 2022, Brookes et al., 2023, Hawkins-Hooker et al., 2024], we reframe the relative score prediction of the neural network on two sequences as parameterizing a binary classifier that sequence $x_i^{(\mathrm{Q})}$ has a higher fitness value that sequence $x_j^{(\mathrm{Q})}$:

$$p\left(y_i^{(\mathrm{Q})} > y_j^{(\mathrm{Q})}\right) = \sigma\left(v_i^{(\mathrm{Q})} - v_j^{(\mathrm{Q})}\right), \tag{1}$$

where σ is a sigmoid function and the dependency of the query values on θ has been dropped for brevity. This classifier is optimized with respect to every pairwise comparison between sequences in the query set corresponding to the loss function

$$\mathcal{L}(\theta, \mathcal{D}_\mathcal{T}^{(\mathrm{Q})}, \mathcal{D}_\mathcal{T}^{(\mathrm{S})}) = -\sum_{i=1}^{N^{(\mathrm{Q})}} \sum_{\substack{j=1 \\ j\neq i}}^{N^{(\mathrm{Q})}} \mathbb{I}\left(y_i^{(\mathrm{Q})} > y_j^{(\mathrm{Q})}\right) \log \sigma\left(v_i^{(\mathrm{Q})} - v_j^{(\mathrm{Q})}\right), \tag{2}$$

where $\mathbb{I}$ is an indicator function. Intuitively, this is optimising $N^{(\mathrm{Q})} \times (N^{(\mathrm{Q})} - 1)$ binary classification problems. Note that we only compute the loss over the query set to avoid encouraging memorization of the support set. Adapting this to meta-learning (Fig. 2a), the objective becomes to find the parameterization that minimises the loss across the task distribution,

$$\mathcal{J}(\theta, \mathcal{D}) = -\mathbb{E}_{\mathcal{D}_\mathcal{T} \in \mathcal{D}} \mathbb{E}_{\mathcal{D}_\mathcal{T}^{(S)}, \mathcal{D}_\mathcal{T}^{(Q)} \in \mathcal{D}_\mathcal{T}} \mathcal{L}(\theta, \mathcal{D}_\mathcal{T}^{(\mathrm{Q})}, \mathcal{D}_\mathcal{T}^{(\mathrm{S})}). \tag{3}$$

Fine-Tuning *Metalic* uses fine-tuning, during inference alone, in order to enable generalization, without having to account for the fine-tuning procedure during meta-training. This process is depicted in Fig. 2b.

The combination of in-context learning and fine-tuning creates a unique problem. Since fine-tuning occurs at inference time, labels for the query set, $\{y_i^{(\mathrm{Q})}\}_{i=1}^{N^{(\mathrm{Q})}}$, are not available for training. While labels for the support set, $\{y_i^{(\mathrm{S})}\}_{i=1}^{N^{(\mathrm{S})}}$, are available, propagating gradients from the support set would encourage memorization of the support set, since the labels are also passed as input in-context. While prior methods that combine in-context and gradient-based meta-learning [Rusu et al., 2018, Vuorio et al., 2019] would encounter this issue, this problem is exacerbated for *Metalic*. Whereas prior methods compress inputs to a representation with a constant number of dimensions, *Metalic* uses self-attention, which scales with the number of inputs, allowing them to be stored without compression. Moreover, whereas prior methods use meta-gradients that could adjust the gradient update procedure so as to be useful for generalization and not memorization, *Metalic* does not take into account the fine-tuning process during meta-training.

We address the issue of memorization by sub-sampling from the support set. The fine-tuning procedure is the same as during meta-training, with the exception that the support set is sub-sampled. In order to compute updates on a single support set, the support set is sub-sampled into multiple smaller support and query sets, $\mathcal{D}_\mathcal{T}^{(\mathrm{S}')} \subseteq \mathcal{D}_\mathcal{T}^{(\mathrm{S})}$ and $\mathcal{D}_\mathcal{T}^{(\mathrm{Q}')} \subseteq \mathcal{D}_\mathcal{T}^{(\mathrm{S})}$. Concretely, this corresponds to fine-tuning on unseen data using the objective,

$$\mathcal{J}(\theta, \mathcal{D}_\mathcal{T}^{(\mathrm{S})}) = -\mathbb{E}_{\mathcal{D}_\mathcal{T}^{(S')}, \mathcal{D}_\mathcal{T}^{(Q')} \in \mathcal{D}_\mathcal{T}^{(\mathrm{S})}} \mathcal{L}(\theta, \mathcal{D}_\mathcal{T}^{(\mathrm{Q}')}, \mathcal{D}_\mathcal{T}^{(\mathrm{S}')}). \tag{4}$$

After fine-tuning, *Metalic* then conditions on the complete support set in-context, allowing no data to go to waste.

Using *Metalic*'s unique combination of in-context meta-learning followed by fine-tuning, we enable the generalization of extensive fine-tuning, while also precluding expensive computation. If a typical gradient-based meta-learning method requires $O(m)$ meta-gradients and $O(mn)$ regular gradients for meta-training, *Metalic* requires no meta-gradients and $O(m)$ regular gradients, constituting a linear reduction with superior performance to efficient alternatives, as demonstrated in Appendix A.2.

4 Experiments

In this section we evaluate *Metalic* on fitness prediction tasks from the benchmark ProteinGym [Notin et al., 2024]. We evaluate in the few-shot setting, with limited support data, and the zero-shot setting, with no support data. To establish SOTA results in the zero-shot setting, we first compare to given predictions provided by ProteinGym. To establish strong performance in the few-shot setting, since predictions are not provided, we train baselines from Hawkins-Hooker et al. [2024], and establish that *Metalic* matches or exceeds the performance of all baselines in all domains. We then perform ablations of *Metalic*, to show the benefits of meta-learning, in-context learning, and fine-tuning. Finally, we compare to the gradient-based method, Reptile Nichol et al. [2018], to show that taking account of gradients during training is an unnecessary complication.

4.1 Experimental Setup

In our experiments, we focus on ProteinGym deep mutational scans. There are 121 single-mutant tasks available, which we augment with multi-mutant tasks from ProteinGym as well, bringing the total to 185 tasks. We evaluate over eight held-out single-mutant tasks, following Hawkins-Hooker et al. [2024]. All fitness values are standardized by subtracting the mean and dividing by the standard deviation by task. Additionally, we remove any multi-mutant tasks that have overlapping proteins with single-mutant tasks to increase the difficulty of single-mutant prediction. To fit the backward pass on an Nvidia A100 device, we limit to tasks in which the maximum protein length is ≤ 750.

We use a query set size of 100, and the size of the support set is determined by our evaluation setting and is one of three sizes: 0, 16, or 128. We also use an additional set of 128 points just for early stopping of the fine-tuning process, for all models except ProteinNPT, following the implementation of Hawkins-Hooker et al. [2024]. We then evaluate remaining points in the task, with a maximum of 2,000 points total, by dividing the data into multiple query sets. If the model, can fit a larger query size, such as baselines that do not involve meta-learning, then we pass the remaining data as a single query set. If the data is not divisible by the query set size, it is left out from evaluation. However, we sample the support data over three independent samples, avoiding systemic exclusion.

All evaluation uses the Spearman rank correlation, in line with prior work [Notin et al., 2023, Truong Jr and Bepler, 2024, Hawkins-Hooker et al., 2024]. We compute the Spearman correlation per task, and then average over tasks. For all evaluations of our models, we compare over three seeds for training and report the mean and standard deviation. Each context, consisting of a support and query set, consists of $\leq$ 171,000 tokens. We meta-train for 100,000 updates and fine-tune for 100. Fine-tuning uses the same procedure, including an Adam optimizer and cosine learning rate scheduler. Using a single Nvidia A100, training our model takes roughly 3 to 13 days per seed, depending on support size and frequency of fine-tuned evaluation.

4.2 Zero-Shot

The first setting we evaluate is the zero-shot performance of our model, when there is no support set for fine-tuning. In Table 1 we report predictions provided by ProteinGym for each baseline to compute the zero-shot Spearman correlation (ρ). We compare to provided predictions, on our data splits, to enable a fair comparison to the strongest models available without retraining each baseline from scratch ourselves. We include the best performing model, and notable models, as baselines. We find that *Metalic* outperforms every reported baseline and is SOTA at zero-shot prediction.

Our method significantly outperforms strong baselines with many more parameters, such as ESM1-v-650M. The 8×10^6 parameter PLM used by our method, ESM2-8M, without our method, achieves a score of only .121 ρ, demonstrating the large contribution of our meta-learning procedure. The next strongest method after ours is VESPA [Marquet et al., 2022]. Note, unlike our method, VESPA is specific to single variant mutants and specific to zero-shot context. Similar to our method, VESPA learns to make zero-shot predictions using features derived from pre-trained PLMs.

The strong performance of our method in the zero-shot setting can be attributed to meta-learning. Since there is no data for fine-tuning, the zero-shot performance increase over ESM2-8M derives entirely from our meta-training procedure. In this case, other methods generally assume that the likelihood of a PLM correlates with fitness in order to make fitness predictions, whereas our model learns to make use of the information contained in PLM embeddings to make predictions zero-shot.

Model Name	$\mathbf{n = 0}$
Metalic	**.475** (max) **.465** $\pm$ .007 (mean $\pm$ std)
VESPA	.464
TranceptEVE-Medium	.457
ESM1-v-650M	.437
Tranception-Medium	.427
Progen2-Medium	.419
ESM2-650M	.399
MSA Transformer	.398
ESM2-8M	.121

Table 1: Spearman correlation in the zero-shot setting. Results are computed using predictions provided by ProteinGym. Most baselines use a pre-trained PLM, so the process is deterministic in the zero-shot setting. For other baselines, we interpret the predictions provided by Notin et al. [2024] as indicating a highly performant model. For comparison, we report our highest performing model, and also provide the mean and standard deviation, for quantifying the variation in meta-training. *Metalic* achieves SOTA performance in either case.

Moreover, our method still conditions on an unlabeled query set, and the protein embeddings in that query set, which allow for meta-learning a form of in-context unsupervised adaptation.

4.3 Fine-Tuning Results

Model Name	$\mathbf{n = 0}$	$\mathbf{n = 16}$	$\mathbf{n = 128}$
Metalic	**.465** $\pm$ .007	**.490** $\pm$ .003	.547 $\pm$ .006
Metalic-NoFT	**.465** $\pm$ .007	.476 $\pm$ .004	.468 $\pm$ .006
ESM1-v-650M	.384 $\pm$ .000	.452 $\pm$ .000	.553 $\pm$ .000
ESM2-8M	.105 $\pm$ .000	.226 $\pm$ .000	.406 $\pm$ .000
PoET	.416 $\pm$ .003	.475 $\pm$ .026	**.588** $\pm$ .006
ProteinNPT (ESM1-v)	N/A	.321 $\pm$ .009	.473 $\pm$ .002

Table 2: Spearman correlation for the 0, 16, and 128-shot setting. All results are re-computed for this paper with standard deviation over seeds reported to show spread. We additionally report *Metalic* without fine-tuning (*Metalic*-NoFT). *Metalic* matches or exceeds all baselines.

In Table 2 we report Spearman correlation with a support set of size 0, 16, and 128, and we compare to baselines that we train and evaluate ourselves over multiple seeds. We re-train these methods using the models provided by Hawkins-Hooker et al. [2024] to provide a comparison over multiple seeds between these methods in a range of practical settings. All models use the same preference-based loss function as *Metalic*, for a fair comparison.

Again, we find that *Metalic* has the strongest performance evaluated in the 0-shot and 16-shot settings, and has comparably strong performance in the 128-shot settings, with 18 times fewer parameters for the underlying embedding model. From these results we also see that fine-tuning drastically improves our model, in the 16-shot and 128-shot settings, by comparing to our method without fine-tuning (No FT). Moreover, we see that our model is also able to outperform contemporary models, such as PoET [Truong Jr and Bepler, 2024], that make use of multi-sequence alignment and in-context learning. Note that for a fair comparison, none of these methods, including our own, make use of ensembling.

Consistent with the motivation of meta-learning, results are strongest when the data is most limited. Meta-learning adds an additional training stage to learn prior beliefs and inductive biases from related data. The more fine-tuning data is restricted, the more prior data is useful.

4.4 Ablations

In Table 3 we report ablations of *Metalic*. Spearman correlation is reported in the zero-shot and fine-tuned 128-shot settings. The results justify all components of our method.

Model Name	$\mathbf{n = 0}$	$\mathbf{n = 128}$
Metalic	**.465** $\pm$ .007	**.547** $\pm$ .006
Metalic-NoFT	**.465** $\pm$ .007	.468 $\pm$ .006
Metalic-NoICL	.443 $\pm$ .008	.495 $\pm$ .011
Metalic-NoMetaTrain	-.047 $\pm$ .044	.163 $\pm$ .018
Metalic-NoAug	.459 $\pm$ .004	.522 $\pm$ .009

Table 3: Ablations in the 0 and 128-shot setting. Results show the importance of fine-tuning, in-context learning, meta-training, and additional training tasks as augmentation.

Most detrimental to performance was removing meta-training our method (NoMetaTrain). This ablation is identical to *Metalic*, with the exception that there is no additional meta-learning stage over multiple protein landscapes. We see that without meta-learning, the initial zero-shot predictions have near zero correlation with the fitness and that the 128-shot predictions are critically impaired.

We also ablate the ability to attend to the rest of the proteins in context by turning off the column attention in the axial attention layers (NoICL), we ablate the fine-tuning stage of training (NoFT), and we ablate the multi-mutant data augmentation by restricting $\mathcal{D}$ to single-mutant variants (NoAug). Removing in-context learning decreases performance in both settings, indicating an ability to adapt in an unsupervised fashion even from the query set alone in the zero-shot setting. Removing fine-tuning decreased performance in the 128-shot setting, indicating the need for fine-tuning for generalization to held-out data. Finally, ablating the multi-mutant augmentation decreased performance in both settings, indicating the advantage of additional meta-training data.

4.5 Gradient-Based Meta-Learning

Model Name	$\mathbf{n = 128}$
Metalic	**.510** $\pm$ .004
Reptile-3-3	.430 $\pm$ .013
Reptile-3-100	.469 $\pm$ .011
Metalic-Reptile	.487 $\pm$ .001

Table 4: Comparison to the gradient-based Reptile [Nichol et al., 2018] method after 16,000 steps. Results show that accounting for fine-tuning during meta-training is unnecessary.

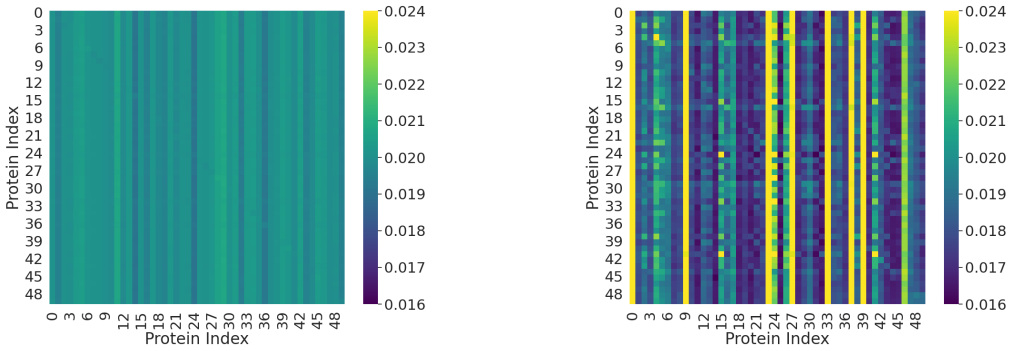
In this section, we compare *Metalic* to the same architecture but trained with Reptile [Nichol et al., 2018], an efficient method for gradient-based meta-learning. Unlike *Metalic*, Reptile does not use in-context learning and modifies the outer-loop during meta-training to take into account the subsequent fine-tuning. While accounting for the fine-tuning during meta-training comes with increased computational costs and can increase bias and variance [Vuorio et al., 2021], Reptile provides a simplified algorithm that can be run more efficiently. The primary differences between Reptile and *Metalic* are that Reptile adjusts the meta-learning loss to account for fine-tuning during meta-training, while *Metalic* has in-context learning. Details are provided in Appendix A.2.

Even though Reptile is more compute-efficient than other gradient-based methods, it is still not efficient. Our method makes use of 100 updates on the support data for fine-tuning. Reptile uses gradient updates during both meta-training and fine-tuning. Due to compute limitations, we cannot use 100 steps during each forward pass of meta-training. We therefore evaluate Reptile with 3 gradient steps during meta-training and 3 during fine-tuning, so that train and test match (Reptile-3-3), and we evaluate Reptile with 3 steps during meta-training and 100 during fine-tuning, so that test time matches our method (Reptile-3-100). Note that even Reptile-3-3 uses three times the compute as *Metalic*. Finally, we evaluate Reptile-3-100 with *Metalic*, to evaluate whether the benefits of each are orthogonal and whether they can be used in conjunction (*Metalic*-Reptile).

Results are reported in Table 4. We evaluate in the 128-shot setting. Note that Reptile is not applicable in the zero-shot setting, since it requires some data on which to fine-tune. We train all methods for 16,000 steps, given the increased computation. We observe that *Metalic* is superior to Reptile, indicating that gradient-based meta-learning to be unnecessary. We also observe that *Metalic* used in conjunction with Reptile (*Metalic*-Reptile) performs better than Reptile, but worse than *Metalic* alone.

Our results are in line with prior work showing that accounting for gradient updates during training (in their case, without in-context learning) can be detrimental when there are a limited number of tasks for meta-training or limited data for the inner-loop [Gao and Sener, 2020, Triantafillou et al., 2020]. Collectively, this evidence further justifies the choice of fine-tuning only after meta-training.

5 Discussion



(a) Untrained Attention

(b) Trained Attention

Figure 3: Axial attention maps over the query set in the zero-shot setting. Each row shows attention to other proteins in a context, normalized to 1 over the row, and averaged over both layers and mutation location. Attention map halfway through training (left) and at the end of training (right). Each protein learns to pay attention to itself, while still attending to other significant proteins in context.

In this section we briefly investigate the in-context learning abilities of *Metalic* through the use of attention maps. In section 4 we show that in-context learning is vital to *Metalic* by ablating the attention between proteins and showing decreased performance. Notably, the in-context learning was beneficial not only in the few-shot setting, but also in the zero-shot setting. This suggests an interesting phenomenon: The emergence of unsupervised in-context learning from the query set alone. To confirm this phenomenon, in Fig. 3 we show the attention maps in the axial attention layers between proteins in the query set. Over the course of training, we observe the emergence of a diagonal line from the top left to the bottom right of the maps. This line indicates that each protein learns to pay attention to itself to predict fitness. (This diagonal is more apparent when averaging over multiple contexts, as depicted in Appendix A.1.) Additionally, we observe bright vertical lines, which demonstrates that there exist some significantly informative proteins that all others proteins attend to. Finally, we observe that no rows that are entirely dark off the diagonal entries. This demonstrates that no protein pays attention to just itself. Taken together, these results further corroborate the effect and necessity of in-context meta-learning. Additional attention maps can be found in Appendix A.1.

6 Conclusion

In this paper, we have demonstrated new state-of-the-art results on a standard protein fitness prediction benchmark dataset when fine-tuning data is limited. In order to do so, we proposed *Metalic*, which makes use of both in-context meta-learning and subsequent fine-tuning. Critically, we have demonstrated the ability of meta-learning to take advantage of additional data from other proteins fitness prediction tasks, while remaining computationally tractable by deferring fine-tuning to test time alone. Unique within the meta-learning literature, we show that in-context meta-learning provides a useful initialization for further fine-tuning, and provides a method that can make use of test time data for both fine-tuning and in-context learning. *Metalic* additionally demonstrates the ability to learn from the query set alone (zero-shot), performing unsupervised in-context learning. Future work could investigate leveraging *Metalic*'s unique in-context learning ability to act as an auto-regressive fitness model (i.e., a world model in the meta-reinforcement learning setting [Beck et al., 2023]), for optimally trading off exploration and exploitation when designing novel proteins. Given its efficacy at leveraging additional data, we believe that meta-learning will play a crucial role in advancing protein fitness prediction methods, with *Metalic* being a foundational first step in that direction.

References

Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn, and Shimon Whiteson. A survey of meta-reinforcement learning. *arXiv*, 2023.

Jacob Beck, Matthew Jackson, Risto Vuorio, Zheng Xiong, and Shimon Whiteson. Splagger: Split aggregation for meta-reinforcement learning. *Reinforcement learning conference*, 2024a.

Jacob Beck, Risto Vuorio, Zheng Xiong, and Shimon Whiteson. Recurrent hypernetworks are surprisingly strong in meta-rl. *Advances in Neural Information Processing Systems*, 2024b.

David H Brookes, Jakub Otwinowski, and Sam Sinai. Contrastive losses as generalized models of global epistasis. *arXiv*, 2023.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *International conference on machine learning*, 2017.

Katelyn Gao and Ozan Sener. Modeling and optimization trade-off in meta-learning. *Advances in neural information processing systems*, 2020.

Alex Hawkins-Hooker, Jakub Kmec, Oliver Bent, and Paul Duckworth. Likelihood-based fine-tuning of protein language models for few-shot fitness prediction and design. In *ICML 2024 Workshop on Efficient and Accessible Foundation Models for Biological Discovery*, 2024.

Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *arXiv*, 2019.

Ben Krause, Nikhil Naik, Wenhao Liu, and Ali Madani. Don't throw away that linear head: Few-shot protein fitness prediction with generative models, 2022. URL `https://openreview.net/forum?id=hHmtmT58pSL`.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Sal Candido, and Alexander Rives. Language models of protein sequences at the scale of evolution enable accurate structure prediction. *bioRxiv*, 2022.

Ali Madani, Bryan McCann, Nikhil Naik, Nitish Shirish Keskar, Namrata Anand, Raphael R Eguchi, Po-Ssu Huang, and Richard Socher. Progen: Language modeling for protein generation. *arXiv*, 2020.

Céline Marquet, Michael Heinzinger, Tobias Olenyi, Christian Dallago, Kyra Erckert, Michael Bernhofer, Dmitrii Nechaev, and Burkhard Rost. Embeddings from protein language models predict conservation and variant effects. *Human genetics*, 2022.

Joshua Meier, Roshan Rao, Robert Verkuil, Jason Liu, Tom Sercu, and Alex Rives. Language models enable zero-shot prediction of the effects of mutations on protein function. *Advances in neural information processing systems*, 2021.

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

Tung Nguyen and Aditya Grover. Transformer neural processes: Uncertainty-aware meta learning via sequence modeling. In *International conference on machine learning*, 2022.

Tianwei Ni, Benjamin Eysenbach, and Ruslan Salakhutdinov. Recurrent model-free rl can be a strong baseline for many pomdps. In *International Conference on Machine Learning*, 2022.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv*, 2018.

Pascal Notin, Mafalda Dias, Jonathan Frazer, Javier Marchena-Hurtado, Aidan N Gomez, Debora Marks, and Yarin Gal. Tranception: protein fitness prediction with autoregressive transformers and inference-time retrieval. In *International Conference on Machine Learning*, pages 16990–17017. PMLR, 2022.

Pascal Notin, Ruben Weitzman, Debora Marks, and Yarin Gal. Proteinnpt: Improving protein property prediction and design with non-parametric transformers. *Advances in Neural Information Processing Systems*, 2023.

Pascal Notin, Aaron Kollasch, Daniel Ritter, Lood Van Niekerk, Steffanie Paul, Han Spinner, Nathan Rollins, Ada Shaw, Rose Orenbuch, Ruben Weitzman, et al. Proteingym: Large-scale benchmarks for protein fitness prediction and design. *Neural Information Processing Systems*, 2024.

Roshan M Rao, Jason Liu, Robert Verkuil, Joshua Meier, John Canny, Pieter Abbeel, Tom Sercu, and Alexander Rives. Msa transformer. In *International Conference on Machine Learning*. PMLR, 2021.

Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C Lawrence Zitnick, Jerry Ma, et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 2021.

Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. In *International conference on learning representations*, 2018.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, 2016.

Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, et al. Meta-dataset: A dataset of datasets for learning to learn from few examples. In *International Conference on Learning Representations*, 2020.

Timothy Truong Jr and Tristan Bepler. Poet: A generative model of protein families as sequences-of-sequences. *Advances in Neural Information Processing Systems*, 2024.

Risto Vuorio, Shao-Hua Sun, Hexiang Hu, and Joseph J Lim. Multimodal model-agnostic meta-learning via task-aware modulation. *Advances in neural information processing systems*, 2019.

Risto Vuorio, Jacob Austin Beck, Gregory Farquhar, Jakob Nicolaus Foerster, and Shimon Whiteson. No dice: An investigation of the bias-variance tradeoff in meta-gradients. In *Deep RL Workshop NeurIPS 2021*, 2021.

Zheng Xiong, Luisa M Zintgraf, Jacob Austin Beck, Risto Vuorio, and Shimon Whiteson. On the practical consistency of meta-reinforcement learning algorithms. In *Fifth Workshop on Meta-Learning at the Conference on Neural Information Processing Systems*, 2021.

Minghao Xu, Zuobai Zhang, Jiarui Lu, Zhaocheng Zhu, Yangtian Zhang, Ma Chang, Runcheng Liu, and Jian Tang. Peer: a comprehensive and multi-task benchmark for protein sequence understanding. *Advances in Neural Information Processing Systems*, 2022.

Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. In *International conference on machine learning*, 2019.

A Appendix

A.1 Attention Maps

In this section we provide additional attention maps beyond those presented in Section 5. The attention maps in Section 5 demonstrated that proteins attend to themselves, in addition to other significant proteins in the context, halfway through training and at the end of training. Here, we provide an attention map at the beginning of training for comparison in Fig. 4. Unsurprisingly, the attention map at the beginning of training is uniform, since no meta-learning has occurred to enable in-context learning. In the main body, the attention maps used a query set of size 50 to make the maps easier to interpret, and evaluated a single context. Here, we additionally provide the map over a

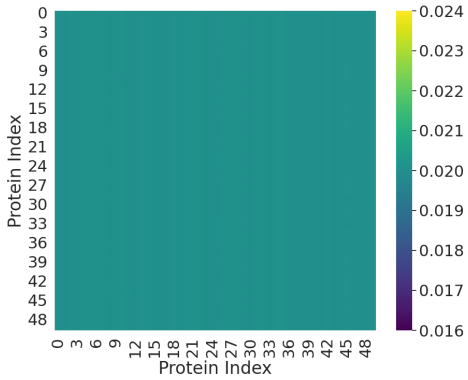
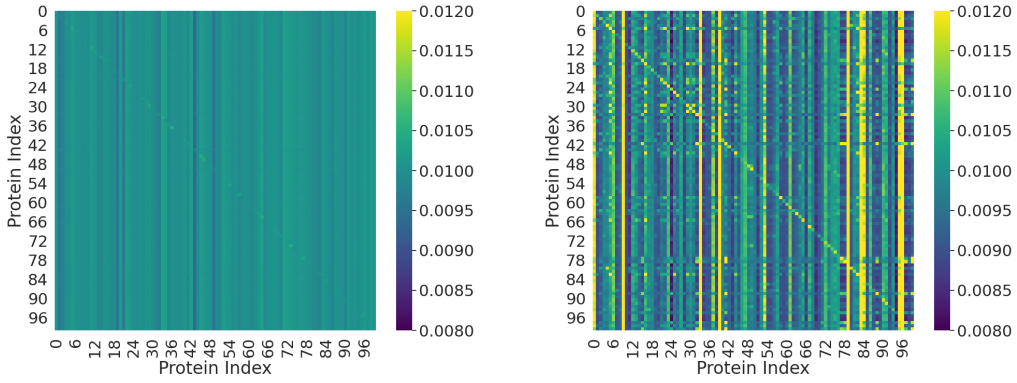


Figure 4: Axial attention maps over the query set in the zero-shot setting at the beginning of training.



(a) Untrained Attention

(b) Trained Attention

Figure 5: Axial attention maps over the query set in the zero-shot setting with query size 100. Attention map halfway through training (left) and attention map at the end of training (right).

query size of 100, as this is the query size used for training all models, and we average over the batch of size 4. These maps are depicted in Fig. 5. Here, the diagonal attention is more apparent in these maps, since the relevant proteins vary between contexts, but the diagonal, averaged over contexts, does not vary. Additionally, we see the same pattern of bright vertical lines, indicating the presence of some significantly informative proteins to which all the others attend. No protein pays attention just to itself, indicating that *Metalic* learns to leverage in-context learning.

A.2 Reptile Details

This section provides additional details on how Reptile [Nichol et al., 2018] works. In order to account for fine-tuning during meta-training, gradient-based methods generally compute a *meta-gradient* that requires the computation of higher order derivatives, which can be computationally intractable for a large model. The costs can be especially burdensome when many gradient steps are needed for out of distribution adaptation. For this reason, Reptile avoids meta-gradients by changing optimization during meta-training. Here, the new parameters are updated, not by gradient descent, but rather by moving toward the mean, after each theta is adapted to a task, $\theta_{\mathcal{T}}$, by fine-tuning on that task. From time-step t to $t+1$ of this outer-loop optimization process, Reptile can be written:

$$\theta^{t+1} = \theta^t + \beta \mathbb{E}_{\mathcal{D}_{\mathcal{T}} \in \mathcal{D}} \left[\theta^t_{\mathcal{T}} - \theta^t \right]. \tag{5}$$

We had to choose several implementation details for Reptile. First, note that Reptile sub-samples the support set during fine-tuning and has no distinct query set. In our implementation, we sub-sample

mini-batches of size 50 to match the query size of *Metalic* for sub-sampling in the 128-shot setting. Reptile also has several options for the outer loop optimization. We choose to use the batched version with a batch size of 4 to match *Metalic*. We also choose to use the direct update given in Equation (5), rather than taking the difference over the learning rate, $(\theta_{\mathcal{T}}^t - \theta^t)/\alpha$ as an approximation of a gradient to be used with the Adam optimizer. We do so to avoid complications from having to tune the $\beta 1$ parameter of Adam, for which *Metalic* uses .9, but the original Reptile implementation uses 0, and to avoid complications regarding when to reset Adam's momentum statistics.

A.3 Hyper-Parameters

Since we build upon the axial attention of ProteinNPT [Notin et al., 2023], we follow their choice for most hyper-parameters, with a few exceptions. Most notably, in our experiments, we use the third layer of ESM2 as an embedding for each protein, given the strong performance and reduced number of parameters. Additionally, we use a ranking loss from Hawkins-Hooker et al. [2024], and do not use the CNN or additional inputs (such as zero-shot predictions) from ProteinNPT. We likewise found conditioning on the wild-type unhelpful. The same set of hyper-parameters are used for each setting: 0-shot, 16-shot, and 128-shot. We used the same learning rate for fine-tuning *Metalic* as for meta-training. We found the default ProteinNPT learning rate to be too large, and decreasing by a factor of 10 to be sufficient. Complete details on hyper-parameters used are available in Table 5. We tuned relatively few of the hyper-parameters of our method, and mostly tuned over a single seed. There is likely room for improvement in the hyper-parameter selection of *Metalic*.

Hyper-Parameter	**Description**	**Value**
Training Steps	The total number of training steps in meta-training	100,000
Warm-Up Fraction	The fraction of total time steps spent linearly warming up, preceding cosine decay	0.05
Batch Size	The number of contexts evaluated per training step. Note that gradient accumulation is used for each context in the batch, so this scales training time linearly.	4
Weight Decay	Weight decay applied to non-bias parameters only	5e-3
Learning Rate	The learning rate for meta-training and fine-tuning	3e-5
Min LR Fraction	The minimum fraction of the LR maintained during the cosine decay in learning rate scheduling	1e-5
Adam Eps	The epsilon value for the Adam optimizer	1e-8
Adam Beta1	The beta1 value for the Adam optimizer	0.9
Adam Beta2	The beta2 value for the Adam optimizer	0.999
Gradient Clip Value	The maximum norm allowed for the gradient	1.0
ESM embed model	The full name for the ESM2 model used	esm_t6_8M_UR50D
ESM embed layer	The layer from the ESM2 model used as an embedding	3
Number Fine-tune Steps	The number of gradient updates for fine-tuning after meta-training	100
Num ProteinNPT Layers	The number of layers using axial attention, as in ProteinNPT	5
Condition on Pooled Sequence	Whether each sequence is pooled or ignored after axial attention	True
MLP Layer Sizes	The number and size of fully connected layers after axial attention	[768,]
Embed Dim	The embedding dimension for all inputs including the protein sequences and fitness values	768
Axial Forward Embed Dim	The hidden size of the feed-forward layer within the ProteinNPT layer	400
Attention Heads	The number of heads in self-attention	4
Dropout Prob	The probability of dropout during training and fine-tuning for layers other than axial attention	0.0
Attention Dropout	The probability of dropout during training and fine-tuning for axial attention layers	0.1

Table 5: Hyper-Parameters for *Metalic*.

For the majority of baselines no tuning was required, other than Reptile. For the baselines, we used reference predictions for Table 1 and reference implementations for Table 2, neither of which required tuning. Our implementation of Reptile in Table 4 did introduce one new hyper-parameter: β in Equation (5), which is the outer-loop learning rate. Unlike in *Metalic*, the outer loop (i.e., meta-training) update magnitude, and thus the outer loop learning rate, depend on the inner loop update magnitude. For Reptile, we use the same inner loop learning rate as *Metalic*, but we tune the outer loop learning rate. Given the increased computational cost of Reptile, we use a single seed over three learning rates for 10,000 steps. We tune over the following learning rates: the learning rate for *Metalic*, which is $3e^{-5}$; a learning rate of 1, which corresponds to the learning rate of *Metalic* if you interpret $(\theta^t_{\mathcal{T}} - \theta^t)/\alpha$ as the gradient [Nichol et al., 2018]; and a learning rate of $\frac{1}{n}$, where n is the number of inner-loop updates (3 in this case), which corresponds to the learning rate of *Metalic* if you interpret $(\theta^t_{\mathcal{T}} - \theta^t)/(\alpha n)$ as the gradient. We found a learning rate of 1 to perform best, in line with the outer loop learning rate of *Metalic* and the interpretation of the gradient from Nichol et al. [2018]. Results of the learning rate tuning are presented in table Table 6.

Model Name	1.	.333	$3e^{-5}$
Reptile-3-3	**.420**	.350	.091
Reptile-3-100	**.444**	.396	.181
Metalic-Reptile	**.476**	.411	.190

Table 6: Reptile tuning results for the 128-shot setting. Results show over 10,000 steps for one seed each.

A.4 Results for All Seeds

In this section we report each seed individually for each experiment conducted in the main body.

Model Name	$\mathbf{n = 0}$
Metalic	.47, .46, .46
VESPA	.464
TranceptEVE-Medium	.457
ESM1-v-650M	.437
Tranception-Medium	.427
Progen2-Medium	.419
ESM2-650M	.399
MSA Transformer	.398
ESM2-8M	.121

Table 7: Spearman zero-shot results for each seed in Table 1.

Model Name	$\mathbf{n = 0}$	$\mathbf{n = 16}$	$\mathbf{n = 128}$
Metalic	.47, .46, .46	.49, .49, .49	.54, .55, .55
Metalic-NoFT	.47, .46, .46	.48, .48, .47	.48, .46, .46
ESM1-v-650M	.38, .38, .38	.45,.45,.45	.55, .55, .55
ESM2-8M	.11, .11, .11	.23, .23, .23	.41, .41, .41
PoET	.41, .42, .42	.48, .50, .44	.60, .59, .58
ProteinNPT (ESM1-v)	N/A	.31, .31, .33	.47, .47, .48

Table 8: Spearman results for the 0, 16, and 128-shot setting for each seed in Table 2.

Model Name	$\mathbf{n = 0}$	$\mathbf{n = 128}$
Metalic	.47, .46, .46	.54, .55, .55
Metalic-NoFT	.47, .46, .46	.48, .46, .46
Metalic-NoICL	.44, .45, .43	.51, .49, .49
Metalic-NoMetaTrain	-.079, .016, -.079	.19, .15, .15
Metalic-NoAug	.46, .46, .46	.53, .51, .52

Table 9: Spearman results for ablations for each seed in Table 3.

Model Name	$\mathbf{n = 128}$
Metalic	.51, .51, .51
Reptile-3-3	.44, .42, .43
Reptile-3-100	.44, .49, .48
Metalic-Reptile	.49, .49, .49

Table 10: Spearman results for Reptile experiments for each seed in Table 4. Results are given after 16,000 training steps.