

Graph neural processes for molecules: An evaluation on docking scores and strategies to improve generalization

Miguel García-Ortegón^{1,2,3*}, Srijit Seal⁴, Carl Rasmussen²,
Andreas Bender³, Sergio Bacallado¹

¹Statistical Laboratory, University of Cambridge, Wilberforce Rd,
Cambridge, CB3 0WA, United Kingdom.

²Department of Engineering, University of Cambridge, Trumpington St,
Cambridge, CB2 1PZ, United Kingdom.

³Department of Chemistry, University of Cambridge, Lensfield Rd,
Cambridge, CB2 1EW, United Kingdom.

⁴Imaging Platform, Broad Institute of MIT and Harvard, 415 Main
Street, Cambridge, 02142, MA, US.

*Corresponding author(s). E-mail(s): mg770@cam.ac.uk;

Abstract

Neural processes (NPs) are models for meta-learning which output uncertainty estimates. So far, most studies of NPs have focused on low-dimensional datasets of highly-correlated tasks. While these homogeneous datasets are useful for benchmarking, they may not be representative of realistic transfer-learning. In particular, applications in scientific research may prove especially challenging due to the potential novelty of meta-testing tasks. Molecular property prediction is one such research area that is characterized by sparse datasets of many functions on a shared molecular space. In this paper, we study the application of graph NPs to molecular property prediction with DOCKSTRING, a diverse dataset of docking scores. Graph NPs show competitive performance in few-shot learning tasks relative to supervised learning baselines common in chemoinformatics, as well as alternative techniques for transfer learning and meta-learning. In order to increase meta-generalization to divergent test functions, we propose fine-tuning strategies that adapt the parameters of NPs. We find that adaptation can substantially increase NP’s regression performance while maintaining good calibration of uncertainty estimates. Finally, we present a Bayesian optimization

experiment which showcases the potential advantages of NPs over Gaussian Processes in iterative screening. Overall, our results suggest that NPs on molecular graphs hold great potential for molecular property prediction in the low-data setting.

1 Introduction

A major difficulty in the application of machine learning (ML) to molecular property prediction in drug discovery is the scarcity of labeled data. Experimental assays are expensive and time-consuming, and data collection is biased towards certain bioactivities (e.g. protein targets deemed medically relevant or commercially profitable) or molecules (e.g. those that are easier to acquire or synthesize). As a result, chemoinformatic datasets are highly sparse and non-overlapping. In a typical pharmaceutical company’s chemical library, it is estimated that less than 1% of all the compound-assay pairs have been measured [1]. Even more strikingly, public databases are as little as 0.05% complete [1, 2].

Meta-learning, or “learning to learn”, is a machine-learning paradigm that attempts to achieve fast adaptation to novel tasks given a small number of labeled datapoints [3]. The meta-learning setting is similar to transfer learning in that it attempts to take advantage of existing information to improve predictions on downstream tasks. However, instead of transferring knowledge from a single pre-training task with many labels, meta-learning attempts to transfer knowledge from multiple meta-training tasks with few labels each. Later, during meta-testing, the model is evaluated on unseen tasks, using a few labeled datapoints from each task as examples. These example points encode information about the meta-test task and are called the contexts. In turn, the query points of interest that we want to predict for each meta-task are called the targets. The ability to learn from meta-training tasks without overfitting and generalizing to novel meta-testing tasks is called meta-generalization.

The meta-learning setting may be appropriate in molecular property prediction [4], since measurements from many different molecular tasks have been collected historically and could be used for meta-training. Examples of molecular tasks are physicochemical properties, protein binding affinities, phenotypic assays or ADMET endpoints [5, 6]. Typically, each task comprises too few datapoints to train a large neural model, but collectively a large set of bioactivities may be useful to learn biases of molecular functions, as well as molecular representations. However, given the sheer diversity of molecular tasks that are available, extra care should be taken to ensure that the biases learnt during meta-training are adequate for meta-testing. For example, tasks related to physicochemical properties, which are intrinsic to molecules, may be very different from cell assays, which depend on the complex interplay between molecules and a biological system [7].

In addition to data efficiency and learning from sparse datasets, another feature that is desirable is the ability to produce uncertainty estimates [8]. Well-calibrated uncertainty estimates are helpful in settings that involve molecular selection and subsequent experimental validation, such as Bayesian optimization (BO) or virtual

screening (VS), since they allow users to balance exploration and exploitation in a principled manner. For example, the selected set could combine some novel but uncertain molecules with others more conservative but certain. Such strategies could help prevent committing to the wrong set of molecules early in development, which is extremely costly [9]. Neural processes (NPs) [10, 11] are a family of models for probabilistic meta-learning that can estimate the uncertainty of each prediction.

Meta-learning for molecular property prediction is a relatively new but rapidly growing area in ML research. Nguyen et al. [4] was an early study that showed the benefits of model-agnostic meta-learning (MAML) for bioactivity classification. However, MAML suffers from lack of robustness during meta-training [12], and the model used in this study did not provide uncertainty estimates, which reduces interpretability and prevents BO for molecular optimization. More recently, Chen et al. [13] presented ADKF-IFT, a deep kernel Gaussian process (GP) for molecular meta-learning. Like NPs, deep kernel GPs are neural models that produce uncertainty estimates. However, the ADKF-IFT method was benchmarked on FS-Mol [14], which is susceptible to overfitting due to random splitting, and it is comparatively difficult to implement, which prevents re-training on novel datasets by chemoinformatics practitioners. NPs for molecules were recently explored for the first time [15–17], yielding promising results. However, the benchmarking experiments in these early studies were limited, not assessing uncertainty calibration, applicability domain or the effect of choosing FP or MG molecular representations. In general, hand-engineered representations like FP are considered advantageous in single-task, low-data settings, whereas data-driven representations derived from MGs may provide benefits in single-task, high-data settings. However, it is unclear whether FPs or MGs should be preferred in the multi-task, low-data setting of meta-learning. The NP model by Chan et al. [17] addressed function heterogeneity in bioactivity datasets by clustering similar assays, effectively denoising predictions from across-assay variability. This strategy could be complementary to ours, which is based on parameter adaptation.

In chemoinformatics research, transfer learning in the low-data setting has been previously approached from the perspective of imputation of sparse datasets [18, 19]. Alchemite [1] is a commercial model for imputation of bioactivities that resembles NPs in that information about the test task is encoded implicitly in the context points. In principle, this makes NPs and Alchemite applicable to any molecular task as long as some molecules have been measured and can be used as context points. In contrast, a related family of models that rely on explicit protein representations are proteochemometric models, which predict affinity values from protein-ligand pairs [20, 21]. Proteochemometric models are less general since they can only be applied to tasks associated with a single known protein, since they depend on explicit protein representations.

In this work, we study the application of NPs to molecular properties and compare their performance to single-task, transfer-learning and meta-learning baselines that are popular in chemoinformatics or in molecular ML, using either a fingerprint (FP) or molecular graph (MG) representation. We conduct our evaluation on DOCK-STRING, a dataset of docking scores and benchmarking framework for regression and molecular optimization that includes labels for 260k molecules and 58 molecular tasks

corresponding to 58 protein targets from diverse protein families [22]. DOCKSTRING provides a controlled environment where each molecule is annotated for every task, and where task similarity can be objectively quantified as the correlation between docking scores of protein targets. These qualities make it ideal for benchmarking molecular meta-learning, since it becomes possible to test few-shot learning (FSL) across a range of dataset sizes, and to assess meta-generalization to novel tasks across a range of similarity to training tasks. We observe that NPs on molecular graphs outperform other models in FSL experiments, and provide well-calibrated uncertainty estimates that may inform the reliability of their predictions. We study the applicability domain of NPs and observe that their performance in distant regions of chemical space remains higher than that of other models. In addition, we suggest a simple yet effective fine-tuning strategy that improves meta-generalization. Finally, we perform a Bayesian optimization (BO) experiment that showcases the potential of NPs for finding optimal molecular candidates within a chemical library.

2 The challenge of meta-generalizing to real-world tasks

One potential obstacle to using meta-learning for molecular property prediction, especially for bioactivity prediction, is the heterogeneity of molecular tasks [17, 23]. Meta-learning attempts to transfer knowledge from a set of tasks with already existing data (the meta-training tasks) to another set of novel tasks with little to no existing data (the meta-test tasks). Crucially, the meta-training tasks must be similar enough to the meta-test tasks, and the heterogeneity among meta-training tasks must be of similar magnitude to the heterogeneity between meta-train and meta-test tasks, for the knowledge transfer to help predict meta-test tasks. However, many meta-learning benchmarks in the ML literature are limited to tasks of low diversity, which may inflate the perceived robustness of meta-learning methods. For example, MAML for regression was evaluated on a simple dataset of 1D sinusoid functions of the same frequency [3], and NPs were evaluated on the MNIST dataset of handwritten digits [10, 11], splitting images randomly into training and test sets even though images in the same digit class (0 to 9) are clearly highly homogeneous.

For an empirical illustration of the challenge of meta-generalization even in simple functions, we consider the set of 1D sinusoids used to evaluate MAML regression [3]. In this benchmark, a task is defined by predicting the value of a function of the form $\sin(\rho(x - B))$ at different values of the input x . Different tasks have different values of B , but the same unit frequency $\rho = 1$. We performed a simple experiment to show that even a slight modification to the meta-test tasks, such as a small change in sinusoid frequency, is enough to compromise meta-generalization in both MAML and conditional NPs (CNPs) (Figure 1 and Supplementary Section F for details). If a small change in a simple function class over a 1D input space can disrupt meta-generalization, mismatches between meta-training and meta-testing in the real world, where one may encounter highly heterogeneous functions over high-dimensional input spaces, are likely to cause problems too.

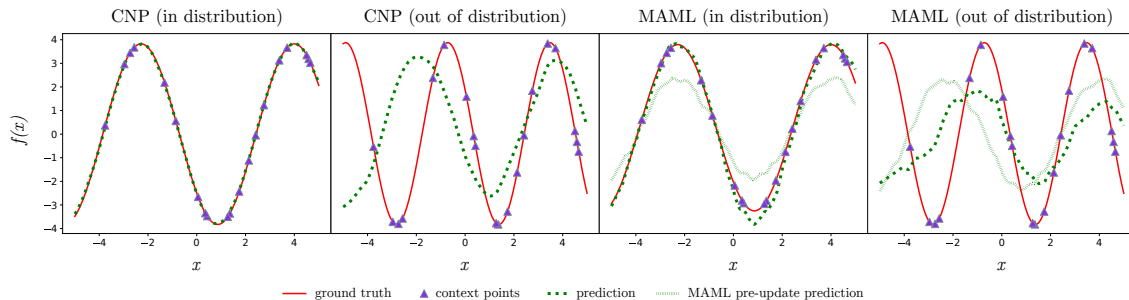


Fig. 1: Meta-generalization experiment on 1D sinusoids. CNPs (left) and MAML (right) meta-trained on sinusoids with frequency $\rho = 1$ generalize to similar test functions. However, changing the frequency to $\rho = 1.5$ leads to catastrophic loss of generalization. Quantitative results are shown in Table F.1. This figure is inspired by Finn et al. [3].

Scientific applications such as bioactivity prediction in preclinical studies may require meta-generalizing to divergent meta-test tasks, since mismatches between meta-training and meta-testing may be unavoidable in a research context. First, by definition research involves an attempt to acquire new knowledge, which means that novel tasks of interest (e.g. novel protein targets, novel cell assays, etc) may not have any known close relatives available for meta-training. Second, if the overlap of labeled datapoints among tasks is limited due to sparsity, it may be difficult to identify the closest task relatives available, which in turn may make it difficult to build an optimal meta-training set. Finally, public bioactivity databases integrate data from a large variety of sources and assays, which leads to extreme task heterogeneity even among seemingly comparable tasks [17, 23]. This heterogeneity was recently quantified by Landrum and Riniker, who developed a “maximal curation” strategy to try to identify affinity assays in ChEMBL that were equivalent, with a view to combining them [23]. In addition to checking that assays were carried out against the exact same protein target and variant (e.g. splice variant, mutant, etc), they also ensured that assays employed the same technology (as per the BioAssay Ontology [24]) and that proteins were expressed in the same strain and purified from the same subcellular fraction. Even after such curation, 48% of overlapping measurements differed by more than the median experimental error. This result points at high heterogeneity even within seemingly equivalent assays. Interestingly, this high variability was observed not only in IC50 measurements, but also in K_i , which is traditionally believed to be more robust [23]. Similar results have been independently reported by others: Chan et al found that out of 190 pairs of ChEMBL IC50 assays against the same protein, only 34 (17.9%) had a Pearson correlation greater than 0.7 [17]. In light of this heterogeneity, meta-learning methods which fail to meta-generalize to divergent test tasks may be of limited practical value to bioactivity prediction, a challenge that current meta-learning benchmarks may overlook.

While these difficulties complicate the application of meta-learning to molecular property prediction, transferring knowledge across very different, even seemingly unrelated tasks has been previously achieved through parameter adaptation [25]. Specifically for molecules, recent work by Formont et al. [26] has shown that fine-tuning may be more robust to shifts in the distributions of molecular labels than traditional meta-learning. The authors noted that this feature increases their practical utility and is currently overlooked in the literature [26]. In this paper, we explore a fine-tuning strategy to adapt the parameters of NPs to novel tasks, as a way to improve generalization during meta-testing. As a benchmarking task, we focus on the prediction of docking scores from the DOCKSTRING dataset, where tasks are different protein targets [22]. This provides us with a controlled environment where the divergence between tasks can be quantified exactly, which enables studying meta-generalization across a range of divergences.

3 Methods

3.1 Neural processes (NPs)

Consider a meta-training dataset with observations of real-valued functions or tasks $f_1, \dots, f_n$, $f_i : \mathcal{X} \rightarrow \mathbb{R}$ (we use the terms *function* and *task* interchangeably in this section, which are not to be confused with the notion of *targets* described below). In this paper, $\mathcal{X}$ represents the space of chemically feasible molecules, and $x \in \mathcal{X}$ refers to a single molecule represented either as a fingerprint vector (FP) or as a molecular graph (MG). Each molecular function f_i is observed at a set of O_i input points $x_o^i \in \mathcal{X}^{O_i}$, with known labels $y_o^i = (y_{o,1}^i, \dots, y_{o,O_i}^i)$, where $y_{o,j}^i = f_i(x_{o,j}^i)$. Additionally, consider a *meta-test* function f , observed at a small set of C *context* points $(x_c, y_c) = ((x_{c,1}, y_{c,1}), \dots, (x_{c,C}, y_{c,C}))$. Our goal is to predict the values y_t of f at a set of T *target* locations $x_t \in \mathcal{X}^T$ as accurately and efficiently as possible, using the example context points (x_c, y_c) and observations from the example functions $f_1, \dots, f_n$ or *meta-training data*.

A neural process (NP) is a model for meta-learning that aims to describe the epistemic or Bayesian predictive distribution of the target outputs given the target inputs, context data, and meta-training data. This predictive distribution will be denoted $q(y_t | x_c, y_c; x_t)$, suppressing the dependence on meta-training data. We use semicolon notation to differentiate contexts and targets, e.g., $q(y_t | x_c, y_c; x_t)$ in a predictive density or $\mathcal{L}(y_t | x_c, y_c; x_t)$ in an objective function. This distinction will be helpful in later sections, where the context outputs could themselves be predicted, e.g., $q(y_c | x_c, y_c; x_c)$ or $\mathcal{L}(y_c | x_c, y_c; x_c)$.

NPs assume conditional independence between target points and a Gaussian predictive distribution:

$$q(y_t | x_c, y_c; x_t) = \prod_{j=1}^T \mathcal{N}(y_{t,j}; \mu_\theta(x_{t,j}, x_c, y_c), \sigma_\theta^2(x_{t,j}, x_c, y_c)),$$

where the mean and variance, μ_θ and σ_θ^2 , are neural functions whose parameters θ will be fit using the meta-training data. For any input $x \in \mathcal{X}$, the functions $\mu_\theta(x, x_c, y_c)$ and $\sigma_\theta^2(x, x_c, y_c)$ are computed in three steps. First, each context point $(x_{c,j}, y_{c,j})$ is mapped by an encoder network h_θ to a local datapoint representation r_j . Then, all context encodings r_j are combined into a global function encoding r through a commutative operation, usually the sum or the mean. Commutativity guarantees invariance to contexts' permutations. Finally, a decoder network g_θ maps the function encoding r and the input location x to the predictive mean and variance, respectively.

In this paper we use two flavors of NPs: the conditional NP (CNP) [10] and the latent NP (LNP) [11]. In the CNP, the decoding step is deterministic. The latent LNP is slightly more complex; decoding involves sampling a latent random variable z from an approximate posterior $\tilde{q}_\phi$, which is then fed as an input instead of r to the decoder network g_θ . A more precise description of the encoding and decoding process is provided in Supplementary Section A.1.

The parameters of the CNP $\psi = \{\theta\}$ are trained by backpropagation from the predictive log-likelihood $\mathcal{L}_\psi(y_t | x_c, y_c; x_t) = \log q_\theta(y_t | x_c, y_c; x_t)$. During meta-training, each meta-train function f_i is seen once every epoch, but not all observations (x_o^i, y_o^i) are used at each iteration. Rather, the O_i observations are randomly subsampled to create two disjoint sets: a context set (x_c^i, y_c^i) and a target set (x_t^i, y_t^i) , with sizes C_i and T_i respectively, $C_i + T_i \leq O_i$. The predictive log-likelihood on the current targets is optimized given the current contexts. Therefore, the final objective is

$$\mathbb{E} \left[\frac{1}{n} \sum_{i=1}^n \mathcal{L}_\psi(y_t^i | x_c^i, y_c^i; x_t^i) \right], \quad (1)$$

where the expectation is with respect to the random sampling procedure. C_i and T_i can themselves be stochastic: in our experiments, we sample them uniformly from $[20, 150)$ at each iteration. We find that this randomization is key to uncertainty quantification, making the model robust to varying context and target sizes at test time. In Section 4.4 we investigate the influence of these hyperparameters on the generalization of NPs.

The parameters of the LNP $\psi = \{\theta, \phi\}$ are trained by backpropagation from a function $\mathcal{L}_\psi(y_t | x_c, y_c; x_t) = \log q_\theta(y_t | x_c, y_c; x_t) + \rho(\phi, x_o, y_o)$, where a regularization term $\rho(\phi, x_o, y_o)$ reduces the sensitivity of the encoder to any given sample (Supplementary Section A.2). This regularization is motivated by a variational Bayesian argument [11]. Meta-training is performed by randomising the context and target sets, as outlined previously, yielding an objective of the form (1).

3.2 Computational considerations of NPs

A forward pass of N datapoints through a NN with fixed architecture has runtime $\mathcal{O}(N)$. A forward pass of T target points through a NP with fixed architecture, conditioned on C context points, requires evaluating the encoder h_θ on the contexts and the decoder g_θ on the targets, which results in runtime $\mathcal{O}(C + T)$. If the NP is evaluated on $\mathcal{T}$ tasks simultaneously, the complexity becomes $\mathcal{O}(\mathcal{T}(C + T))$.

In practice, forward propagation of NNs and NPs is parallelized so that a whole batch is processed at the same time, with the size of the batch B chosen so as to fit the memory of the GPU. A batch for a NN will contain B_N of the N datapoints. Assuming each datapoint is a vector of length M , a NN batch will have dimensions $B_N \times M$ (2D). In contrast, a batch for a NP will contain $B_{\mathcal{T}}$ of the $\mathcal{T}$ tasks, so it will have dimensions $B_{\mathcal{T}} \times (C + T) \times M$ (3D). Higher-dimensional datapoints (e.g. 2D adjacency matrices if representing molecules as graphs) will result in even higher-dimensional batches.

The higher dimension of NP batches with respect to NN batches may result in high memory usage, which may make it difficult to fit the GPU. In practice, running out of memory can be avoided by choosing a lower batch size $B_{\mathcal{T}}$ or by implementing automated checks in the NP encoder h_{θ} and decoder g_{θ} so that, if the context size C or the target size T in a batch are high, contexts and targets in that batch are chunked and processed in several passes rather than in a single pass.

3.3 Effective epochs

Since only some observations of a function f_i are sampled as contexts or targets every epoch, how often an individual observation is seen will depend on the sample size relative to the total number of observations for that function O_i , which for molecular datasets can vary widely (Supplementary Section B). In order to homogenize training across varying sample sizes and observed sets, we introduce the concept of *effective epochs* e_e , which we define as the average number of times an observed datapoint is seen during training. This quantity is calculated as

$$e_e^i = e \frac{\bar{n}}{O_i},$$

where e is the number of epochs and $\bar{n}$ is the average sample size (sample size could itself be random). In our experiments, the average number of contexts and targets while training is the same in every experiment, so effective epochs refer to views of an observation both as context and target.

3.4 Parameter adaptation during meta-testing

When a NP is applied to a test function f with contexts (x_c, y_c) , its predictions on the contexts themselves may be inaccurate. In particular, if a NP fails to meta-generalize, the predicted context density

$$q(y_c \mid x_c, y_c; x_c) = \prod_j^C \mathcal{N}(y_{c,j}; \mu_{\theta}(x_{c,j}, x_c, y_c), \sigma_{\theta}^2(x_{c,j}, x_c, y_c))$$

may be inadequately low, even though the contexts are given as input. In this situation, the loss on the context predictions $\mathcal{L}_{\psi}(y_c \mid x_c, y_c; x_c)$ can be exploited to adapt the weights to the test function f , potentially improving meta-generalization. We studied two strategies for parameter adaptation by backpropagation during meta-testing:

fine-tuning and a single step of gradient descent on a NP trained with model-agnostic meta-learning (MAML) [3]. MAML is a meta-learning training regime that involves nested gradient descent and is notorious for displaying instability during training [12]. Although we implemented improvements from the successor MAML++ [12] to increase robustness, we still observed high variance in MAML-trained models, and average performance was lower than that of fine-tuning. Therefore, for the remainder of this work we will focus on fine-tuning. See Supplementary Section C for a description of MAML and to view MAML results.

We fine-tuned NPs by mimicking the meta-training procedure on the test function f (Algorithm 1). To this end, at each epoch we split the test function’s contexts (x_c, y_c) into new contexts and new targets, as if they were the observations of a meta-train function. Then, we used the new contexts $(x_{c'}, y_{c'})$ and targets $(x_{t'}, y_{t'})$ to evaluate the NP’s objective $\mathcal{L}_\psi(y_{t'} \mid x_{c'}, y_{c'}; x_{t'})$ and backpropagate as usual. In this way, fine-tuning during meta-testing resembled meta-training, but instead of iterating over many train functions every epoch, it focused on a single test function. As in meta-training, the new contexts and targets were disjoint, with sizes $C' + T' \leq C$. We set the number of desired effective epochs e_e as a hyperparameter, and calculated the required number of actual epochs e required to achieve the desired e_e based on the original context size C and the number of new targets T' .

In our experiments, we adapted the weights for 20 effective epochs, sampling $C' = T' = 20$ new contexts and targets every iteration. The only exception from this protocol was the few-shot learning (FSL) experiment with 20 observations, where we used $C' = T' = 5$. To minimize the risk of overfitting, we fine-tuned the last layers and froze the rest. Specifically, we always adapted the last two layers of the decoder network g_θ , and in LNPs we also adapted the last two layers of the encoder g_ϕ (see Appendices A.1 and E for architectural notation and details). The latter allowed us to optimize the regularization term of the LNP objective $\mathcal{L}_{\theta, \phi}$, which depends on the encoder (Supplementary Section A.2).

3.5 Molecular graph neural processes (MG-NPs)

In order to apply NPs to molecular graphs (MGs), we expanded the encoder module with a graph neural network (GNN) that processed atom and bond features and implemented message passing with attention. Our architecture was inspired by the Attentive FP model [27], with the differences that we processed atom and bond features with a small network before and after message passing, we removed gated recurrent units (GRUs) [28] and we changed the attention mechanism to query-key-value (QKV) [29]. These changes were aimed at speeding computation while maintaining expressivity.

Prior to message-passing, atoms and bonds were processed with a small fully-connected neural network (FNN, 2 layers and 50 hidden neurons) to encode atomic features into vectors of length 25 and bond features for each atom i into matrices of size $n_i \times 25$, where n_i is the number of neighbours of atom i . Then, bond feature matrices were summed along the row dimension to yield a single bond vector of length 25 for each atom. This was concatenated to the previous atomic encoding, such that the final atomic encoding (before message passing) was a vector of length (50). Then, we updated the atomic encodings with 3 iterations of message passing from direct

Algorithm 1 Fine-tuning: Adapting NP parameters to test function f

Require: e_e : number of effective epochs

Require: (x_c, y_c) : C context points from test function f

Require: C', T' : number of new context and target points to sample from (x_c, y_c) .

Ensure: $C' + T' \leq C$

Require: NP with parameters ψ meta-trained on functions other than f .

Require: $\mathcal{L}_\psi(y \mid x_{c'}, y_{c'}; x)$: loss function

Require: Gradient-based optimization method and hyperparameters

- 1: $e \leftarrow e_e \frac{C}{T'}$ $\triangleright$ Number of epochs e needed to achieve the desired effective epochs e_e
 - 2: **repeat**
 - 3: Sample C' new context points $(x_{c'}, y_{c'})$ from (x_c, y_c) .
 - 4: Sample T' new target points $(x_{t'}, y_{t'})$ from $(x_c, y_c) \setminus (x_{c'}, y_{c'})$.
 - 5: Evaluate gradient $\nabla_\psi \mathcal{L}_\psi(y_{t'} \mid x_{c'}, y_{c'}; x_{t'})$.
 - 6: Update ψ with gradient-based method. $\triangleright$ Optionally, can update a subset of ψ .
 - 7: **until** reached e iterations
-

atomic neighbours, calculating attention coefficients with a QKV mechanism. The queries and keys are computed with a single-layer FNN. After the initial 3 iterations of message passing, a second stage began where an imaginary “superatom” connected to all atoms was added to the graph, and another 3 iterations of message-passing were performed. The encoding of the superatom is treated as the representation for the whole molecule. As a final step, the superatom encoding was passed through a small FNN to yield the final molecular representation, also of length 50.

Implementation details of the MG-NP and all other baseline models are provided in Supplementary Section E.

3.6 Dataset and data splits

DOCKSTRING is a bundle for benchmarking ML models for molecules using docking scores as objective properties [22]. It consists of a dataset of scores and poses and a set of evaluation tasks that reflect the model’s performance in regression, virtual screening and multiobjective molecular optimization. The dataset includes Autodock Vina scores for every ligand-target pair in 260k molecules from ExCAPE [30] and 58 targets from DUD-E [31]. While other popular benchmark datasets in ML for drug discovery contain sparse annotations, in the sense that not every molecule included is labeled for every property considered (e.g. MoleculeNet [32], FS-Mol[14]), the DOCKSTRING ligands have docking scores for every target, which makes DOCKSTRING suitable to the design of new benchmark tasks in multi-task learning, transfer learning or meta-learning. Note that docking scores in DOCKSTRING are not intended for downstream applications, but rather as proxy properties for benchmarking that are loosely reflective of the 3D interactions between a molecule and a protein structure. Thus, the purpose of DOCKSTRING scores is not to perfectly reflect binding affinity

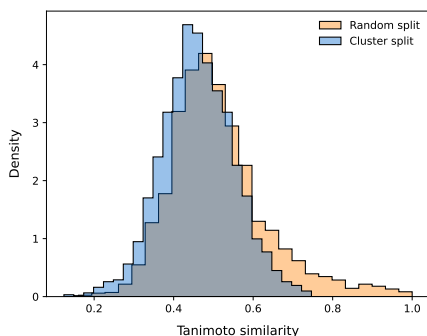


Fig. 2: Distribution of the Tanimoto similarity of every molecule in the test set to its closest neighbour in the training set, either in the DOCKSTRING split ($dtrain$ and $dtest$) or in an example random split. Tanimoto similarity values were computed on Morgan fingerprints of length 1024 and radius 3, calculated with RDKit.

or to improve docking algorithms, but to enable a reproducible and fair comparison of ML models on the same set of benchmark tasks. To this end, ligand preparation and docking were performed with fixed random seeds so as to maximize repeatability across different runs, and each benchmark task has well-defined training and test sets to ensure a fair model comparison. In the regression benchmark, roughly 220k compounds were assigned to the training set and 40k were assigned to the test set. They were split by chemical scaffolds to reduce the risk of data leakage from chemical analogues [33–35]. The regression benchmark focuses on 5 diverse proteins, some of which have scores that are relatively easy to predict, and some which are harder, so as to reflect a range of prediction difficulty.

Few-shot learning (FSL)

For our meta-learning evaluation, we needed to create splits not only across datapoints ($dtrain$ and $dtest$) but also across tasks or functions ($ftrain$ for meta-train and $ftest$ for meta-test). Regarding datapoints, in the few-shot learning (FSL) experiments, we respected the scaffold splits of the DOCKSTRING regression benchmark, sampling 2.5k molecules from the training set and 2.5k molecules from the test set for our $dtrain$ and $dtest$ respectively. To confirm that this split was more stringent than a random one, we visualized (Figure 2) the distribution of the Tanimoto similarity of every molecule in $dtest$ to its closest neighbour in $dtrain$ (blue histogram), and compared it to the homologous distribution arising from a random split (orange histogram). As expected, the random split led to chemical analogues spread across the training and test set, as reflected by the heavy tail of high Tanimoto similarities. In contrast, the DOCKSTRING split was more strict and prevented such analogues between $dtrain$ and $dtest$. We also visualized a random sample of datapoints from $dtest$ and their closest three molecules in $dtrain$ according to Tanimoto similarity (Figure 3). We

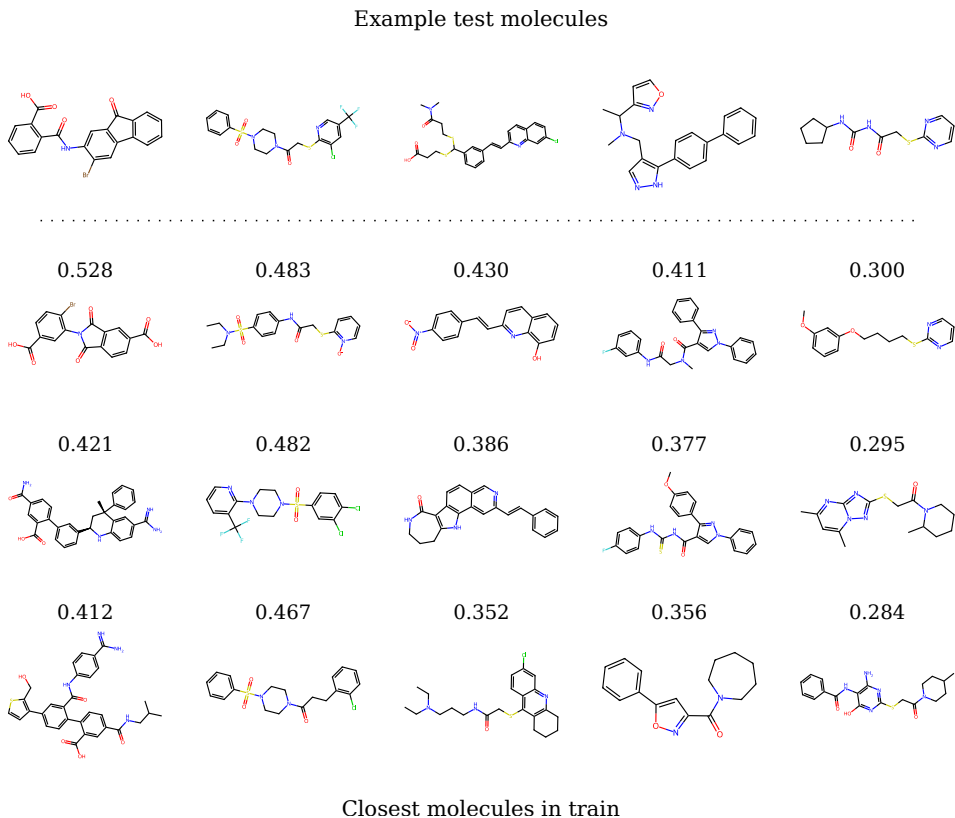


Fig. 3: 5 random molecules from the test set and their three closest neighbours in the training set. Tanimoto similarity values were computed on Morgan fingerprints of length 1024 and radius 3, calculated with RDKit.

found that the similarity values of the closest neighbours were relatively low, and no neighbours shared the same scaffold across training and test sets.

Regarding functions, we selected three proteins from the regression benchmark (PARP1, ESR2 and PGR) as meta-test f_{test} and kept the 53 training proteins from the benchmark as meta-train f_{train} (Figure 4). The three proteins chosen for meta-testing reflected a range of similarity (high, medium and low) to other proteins, as indicated by the Pearson correlation of their docking scores on the full dataset of 260k molecules (Figure 5). This allowed us to examine the generalization of transfer learning and meta-learning methods to novel tasks which were either close or distant to previously seen tasks.

The protein and datapoint splits of the FSL experiment were used by each model family as follows:

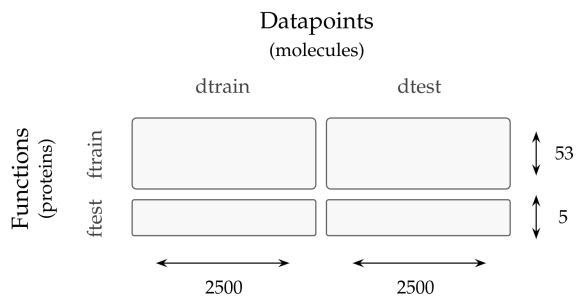


Fig. 4: Data split in the FSL experiments. Meta-learning models were trained on $ftrain, dtrain$ and tested on $ftest, dtest$, using points from $ftest, dtrain$ as contexts.

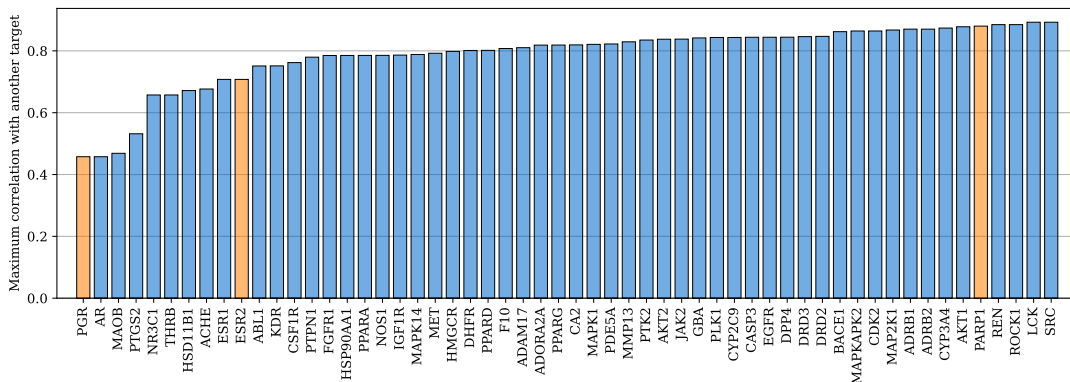


Fig. 5: Maximum Pearson correlation of the docking scores of each protein in DOCKSTRING to those of any other protein. The maximum correlation of each protein suggests how challenging it would be for meta-generalization. We identified three proteins from the DOCKSTRING regression benchmark (orange bars) that represented a range of dissimilarity to other proteins: PARP1 (highly correlated, very similar), ESR2 (medium correlation and similarity) and PGR (low correlation and similarity). We selected these proteins as our meta-test set, and all proteins not in the regression benchmark as meta-training set. In this way we aimed to illustrate meta-generalization across a range of correlations.

- Meta-learning models were meta-trained on $ftrain, dtrain$. During meta-testing, they used random points from $ftest, dtrain$ as contexts and all points from $ftest, dtest$ as targets. Metrics were calculated on all points from $ftest, dtest$.
- Transfer-learning models were pre-trained on $ftrain, dtrain$ and fine-tuned on random points from $ftest, dtrain$. Metrics were calculated on all points from $ftest, dtest$.

- Single-task models were trained on points randomly sampled from *ftrain*, *dtrain*. Metrics were calculated on all points from *ftrain*, *dtest*.

As an alternative for benchmarking FSL other than DOCKSTRING, we also considered using FS-Mol [14], a dataset for molecular meta-learning that includes bioactivity values from ChEMBL. However, FS-Mol only considers classification tasks, whereas NPs are usually used for regression; its labels are non-overlapping across molecules, which precludes flexible subsampling; and it splits molecules randomly, which may lead to data leakage and an overestimation of performance [33–35]. In contrast, DOCKSTRING is a complete matrix with full overlap of annotations across tasks, which allows flexible sampling of variably-sized subsets, and it splits molecules by scaffold, which minimizes the risk of data leakage from chemical analogues (Figures 2 and 3).

Bayesian optimization (BO)

We created a library of 60k compounds for the BO experiments by sampling 30k molecules from the DOCKSTRING regression training set and 30k from the test set. As objective functions to optimize, we used the druglike F2 and the selective JAK2 objectives from the DOCKSTRING optimization benchmark (Supplementary Section H). These objective functions were treated as meta-test tasks by the meta-learning models: at each iteration of BO, all molecules seen up to that point were used as contexts and all 60k molecules in the library were used as targets (at each iteration, we selected the best predicted molecules from these targets which had not been seen yet). For meta-training tasks, we created an augmented dataset where scores were transformed either linearly (scalar multiplication and linear combination) or non-linearly (by taking the minimum or the maximum between a given score and the median of the score distribution for the corresponding protein target), and were combined with the quantitative estimate of drug-likeness (QED). We created 869 such transformed tasks; this odd number arose from creating an augmented dataset of 1000 transformations and discarding those which included F2, JAK2 or LCK, since those three proteins were part of the DOCKSTRING objective functions. We meta-trained meta-learning models on the 869 augmented functions. We trained two different MG-CNPs: one on the whole 60k molecules of the BO library and one on the 2.5k *dtrain* molecules from the FSL split.

3.7 Molecular representations

All models used either a fingerprint (FP) or molecular graph (MG) representation. We computed binary Morgan FPs [36] of length 1024 and radius 3 with RDKit [37]. MGs were obtained from SMILES strings [38], using RDKit to extract the connectivity graph and the atom and bond features. Connectivities were represented in PyTorch as adjacency matrices. Atom features consisted of one-hot vectors indicating the atom type, the number neighbouring hydrogens, the number of neighbouring heavy atoms, the formal charge, the hybridisation type, whether the atom is placed within a ring, whether the atom is in an aromatic region and whether the atom is chiral. Atom features also included, as real numbers, the atomic mass, the Van der Waals radius

and the covalent radius. Bond features included one-hot vectors with the bond type (single, double, tripe or aromatic), whether the bond is conjugated, whether it is part of a ring, and the stereoisometry type, if any.

All models labeled with "FP" used the fingerprint representation and all models labeled with "MG" used the molecular graph representation. GNNs also used the MG representation. The ADKF-IFT baseline model came pre-trained and used its own FPs (Morgan FPs of length 2048 and radius 2). The dummy regressor simply predicted the mean of the training molecules and did not use any representation.

4 Results

4.1 Few-shot learning (FSL)

We evaluated few-shot learning by MG-NPs on docking scores from the DOCK-STRING dataset and compared their performance to a variety of baselines, including single-task, transfer learning and meta-learning models. Few-shot learning (FSL) is a ML paradigm where models are trained using just a handful of datapoints, typically between 5 and 50. In the meta-learning setting, FSL is achieved by first meta-training the model on several tasks, each of them with a low-to-medium number of labels, and then adapting the predictions of the model to a novel meta-test task for which just a few datapoints are available. In the context of molecular property prediction, the goal of FSL is to predict a new molecular task for which just a few molecules are annotated.

To recreate a low-data setting, we created a small training set by sampling a subset of 2.5k molecules from the DOCKSTRING training set, and a test set by sampling the same number from the test set, which we call the *dtrain* and *dtest* respectively (Section 3.6). We trained all models on the 2.5k molecules from the training set. Single-task models were trained on *ftest*, *dtrain* and meta-learning models were meta-trained on *fttrain*, *dtrain*. At test time, we evaluated all models on *ftest*, *dtest*, with meta-learning models using context points sampled randomly from *ftest*, *dtrain* (Section 3.6 and Supplementary Section E). We inspected a range of numbers of context points, from 20 to 1000. In this way, we aimed to reflect the information available in bioactivity datasets, where the number of observations per task may fluctuate considerably (Supplementary Section B). The range of contexts also allowed us to examine the robustness of NPs, and assess whether their uncertainty estimates were well calibrated over a number of observations.

Table 1 shows the coefficient of determination (R^2) for the prediction of the three meta-test proteins: PARP1, ESR2 and PGR. These proteins represent a range of similarity to the meta-training tasks, from high to low, as shown by the correlation of their docking scores to other proteins (Figure 5). The lower the similarity of a meta-task to meta-training tasks, the more challenging it will be for meta-learning models, and the greater the potential benefit of parameter adaptation. When the mismatch between meta-training and meta-testing was low (as for PARP1), all meta-learning methods outperformed single-task models in the low-data regime, as expected. However, when the mismatch grew (ESR2 and PGR), the biases learnt during meta-training could become detrimental to prediction, leading to poor performance by meta-learning methods. Parameter adaptation by fine-tuning could greatly improve predictions,

especially when there was a mismatch. We also attempted parameter adaptation by MAML but it suffered from training instability, yielding large error bars (Table D.1). In general, CNPs performed better than LNPs, and NPs using molecular graphs (MGs) as representations performed significantly better than NPs using fingerprints (FPs). The MG-CNP ranked consistently as the best model in terms of the coefficient of determination R^2 by a large margin. However, ADKF-IFT [13], a GP-based model that meta-learns a deep kernel, showed the best calibration of uncertainty estimates according to the negative log predictive likelihood (NLPD, Table D.2).

Our results show that meta-learning MG-NPs can be highly beneficial for molecular property prediction in the low-data setting. We selected the MG-CNP as the best-performing model for more detailed analysis in the following sections.

		PARP1 (high correlation)							ESR2 (medium correlation)							PGR (low correlation)									
		Number of context points							Number of context points							Number of context points									
		20	50	100	200	500	1000	20	50	100	200	500	1000	20	50	100	200	500	1000	20	50	100	200	500	1000
Single task	Dummy regressor	-13.6 (3.9)	-18.9 (3.0)	-19.6 (2.9)	-21.4 (1.7)	-21.9 (1.1)	-21.4 (0.6)	-5.5 (1.9)	-3.5 (1.1)	-3.0 (1.1)	-3.5 (0.8)	-3.3 (0.4)	-3.1 (0.3)	-13.3 (4.6)	-5.3 (1.9)	-2.7 (0.8)	-1.6 (0.4)	-1.6 (0.3)	-1.5 (0.2)	-13.3 (4.6)	-5.3 (1.9)	-2.7 (0.8)	-1.6 (0.4)	-1.6 (0.3)	-1.5 (0.2)
	FP-RF	0.4 (2.1)	-1.2 (2.3)	3.0 (2.5)	15.1 (1.9)	23.7 (0.8)	30.9 (0.5)	-5.3 (3.0)	6.6 (2.0)	11.2 (1.4)	15.4 (1.4)	21.0 (0.5)	25.1 (0.2)	-13.2 (8.7)	-4.7 (2.2)	-0.3 (1.4)	2.0 (1.3)	8.9 (0.8)	11.9 (0.5)	-13.2 (8.7)	-4.7 (2.2)	-0.3 (1.4)	2.0 (1.3)	8.9 (0.8)	11.9 (0.5)
	FP-GP	6.4 (0.7)	9.5 (1.7)	15.6 (2.1)	29.6 (1.3)	47.5 (0.7)	58.1 (0.3)	2.0 (1.0)	8.3 (1.3)	14.0 (1.3)	19.9 (0.9)	26.7 (0.4)	31.8 (0.4)	-11.3 (4.9)	1.0 (1.8)	6.2 (0.9)	8.4 (0.7)	8.9 (0.6)	11.2 (0.8)	-11.3 (4.9)	1.0 (1.8)	6.2 (0.9)	8.4 (0.7)	8.9 (0.6)	11.2 (0.8)
	GNN	22.9 (11.9)	21.0 (13.9)	63.9 (4.2)	61.7 (9.0)	73.2 (0.6)	75.4 (0.5)	-10.7 (4.9)	-7.4 (3.6)	1.8 (3.9)	6.4 (3.9)	14.8 (3.5)	19.8 (4.2)	-56.6 (33.0)	-13.0 (6.6)	-4.4 (2.6)	-1.9 (4.8)	6.7 (1.6)	17.4 (3.0)	-56.6 (33.0)	-13.0 (6.6)	-4.4 (2.6)	-1.9 (4.8)	6.7 (1.6)	17.4 (3.0)
Transfer learning	GNN (random)	63.8 (1.4)	69.9 (1.3)	70.7 (2.5)	73.7 (1.3)	76.2 (0.7)	77.6 (1.3)	5.7 (3.2)	19.1 (2.1)	10.8 (4.9)	17.8 (2.5)	24.3 (2.0)	29.2 (1.6)	-21.2 (16.9)	14.5 (2.6)	9.9 (3.8)	13.5 (3.9)	23.3 (1.6)	29.9 (0.7)	-21.2 (16.9)	14.5 (2.6)	9.9 (3.8)	13.5 (3.9)	23.3 (1.6)	29.9 (0.7)
	GNN (fine-tuned)	75.0 (0.9)	77.3 (0.6)	77.9 (0.3)	76.2 (0.4)	75.4 (0.3)	77.1 (0.2)	34.9 (1.5)	37.8 (0.8)	35.2 (1.1)	30.3 (1.2)	30.3 (1.0)	34.2 (0.6)	21.1 (1.8)	26.4 (1.2)	23.7 (1.4)	16.3 (1.1)	19.4 (1.0)	24.7 (1.0)	21.1 (1.8)	26.4 (1.2)	23.7 (1.4)	16.3 (1.1)	19.4 (1.0)	24.7 (1.0)
Meta-learning	ADKF-IFT	10.8 (3.4)	28.2 (1.8)	38.8 (1.5)	47.5 (1.1)	55.2 (0.6)	61.9 (0.3)	-0.3 (1.4)	8.1 (1.3)	13.9 (0.9)	18.0 (1.1)	24.6 (0.7)	29.4 (0.4)	-6.6 (4.1)	1.7 (1.4)	4.5 (0.7)	5.2 (0.8)	8.7 (0.9)	14.6 (0.4)	-6.6 (4.1)	1.7 (1.4)	4.5 (0.7)	5.2 (0.8)	8.7 (0.9)	14.6 (0.4)
	FP-CNP	55.5 (0.7)	55.5 (0.5)	55.4 (0.5)	55.2 (0.4)	55.4 (0.3)	55.3 (0.4)	29.0 (0.8)	26.0 (2.0)	27.2 (1.3)	28.8 (0.7)	29.1 (0.6)	29.7 (0.5)	-36.7 (4.7)	-37.5 (4.5)	-39.1 (3.8)	-39.7 (2.3)	-42.4 (2.9)	-41.6 (2.5)	-36.7 (4.7)	-37.5 (4.5)	-39.1 (3.8)	-39.7 (2.3)	-42.4 (2.9)	-41.6 (2.5)
	MG-CNP	81.6 (0.6)	82.7 (0.3)	83.0 (0.1)	83.1 (0.1)	83.1 (0.1)	83.0 (0.2)	43.2 (1.0)	44.9 (0.7)	45.3 (0.6)	45.6 (0.5)	46.3 (0.5)	46.6 (0.6)	19.1 (2.5)	23.2 (1.7)	25.6 (1.4)	27.5 (1.5)	27.5 (1.2)	27.2 (1.3)	19.1 (2.5)	23.2 (1.7)	25.6 (1.4)	27.5 (1.5)	27.5 (1.2)	27.2 (1.3)
	MG-LNP	81.4 (0.3)	81.5 (0.3)	81.6 (0.3)	81.7 (0.3)	82.0 (0.3)	82.3 (0.3)	39.9 (1.3)	40.0 (1.3)	40.1 (1.2)	40.3 (1.2)	41.0 (1.0)	41.5 (0.9)	2.4 (3.7)	3.0 (3.7)	3.8 (3.7)	5.5 (3.8)	10.6 (3.7)	15.7 (3.4)	2.4 (3.7)	3.0 (3.7)	3.8 (3.7)	5.5 (3.8)	10.6 (3.7)	15.7 (3.4)
NPs with parameter adaptation	FP-CNP (fine-tuned)	55.3 (0.8)	55.2 (0.7)	55.7 (0.5)	55.7 (0.4)	56.1 (0.3)	56.2 (0.2)	30.4 (0.8)	29.5 (1.2)	29.9 (0.9)	31.4 (0.7)	33.3 (0.3)	34.0 (0.4)	-24.5 (5.2)	-13.7 (1.5)	-9.2 (1.3)	-5.7 (1.5)	1.6 (1.1)	7.7 (0.8)	-24.5 (5.2)	-13.7 (1.5)	-9.2 (1.3)	-5.7 (1.5)	1.6 (1.1)	7.7 (0.8)
	FP-LNP (fine-tuned)	38.8 (1.3)	36.0 (1.4)	37.6 (1.4)	40.3 (1.2)	41.6 (1.2)	41.2 (0.7)	29.6 (0.6)	30.6 (0.6)	29.8 (0.5)	29.0 (0.8)	30.4 (0.6)	29.8 (0.7)	-1.2 (3.4)	3.6 (0.6)	7.7 (0.8)	8.3 (0.6)	7.3 (0.9)	6.2 (0.6)	-1.2 (3.4)	3.6 (0.6)	7.7 (0.8)	8.3 (0.6)	7.3 (0.9)	6.2 (0.6)
	MG-CNP (fine-tuned)	82.1 (0.5)	83.4 (0.2)	83.7 (0.2)	83.9 (0.2)	84.4 (0.2)	84.9 (0.1)	45.0 (0.9)	45.3 (0.8)	46.9 (0.8)	49.7 (0.4)	51.3 (0.4)	52.5 (0.4)	20.3 (3.9)	31.9 (1.4)	36.6 (1.2)	40.7 (0.8)	43.7 (0.2)	45.1 (0.3)	20.3 (3.9)	31.9 (1.4)	36.6 (1.2)	40.7 (0.8)	43.7 (0.2)	45.1 (0.3)
	MG-LNP (fine-tuned)	82.5 (0.4)	82.3 (0.5)	83.0 (0.3)	83.5 (0.3)	84.2 (0.2)	84.7 (0.2)	43.3 (0.9)	42.8 (0.9)	46.2 (0.6)	47.3 (0.7)	49.5 (0.4)	50.4 (0.6)	10.8 (9.3)	7.7 (6.5)	18.3 (3.7)	31.6 (2.0)	40.1 (1.5)	42.2 (1.6)	10.8 (9.3)	7.7 (6.5)	18.3 (3.7)	31.6 (2.0)	40.1 (1.5)	42.2 (1.6)

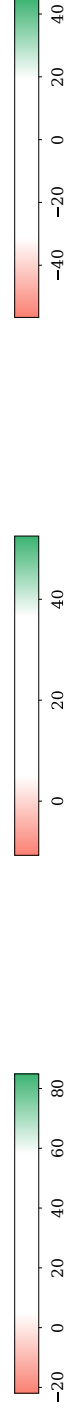


Table 1: Coefficient of determination (R^2) of molecular NP models and baselines in FSL prediction of PARP1, ESR2 and PGR. Each cell shows the mean and standard error (in parentheses) of 10 random repetitions. The MG-CNP ranked consistently as the best model in terms of R^2 . Fine-tuning led to substantial improvements, especially when the meta-test task was poorly correlated when meta-training tasks (as in ESR2 and PGR) and there were many observations available.

4.2 Applicability domain

In order to ascertain the applicability domain of our models in the low-data regime, we examined the quality of FSL predictions in the test set *dset* across a range of distances to the training set *dtrain*. We grouped test molecules by their Tanimoto similarity to the training set and computed prediction metrics for each model within each group (Figure 6). We compared predictions by the MG-CNP with predictions from two single task baselines (FP-GP and FP-RF), one transfer-learning baseline (fine-tuned GNN) and one meta-learning baseline (ADKF-IFT). All models used 50 molecules from *ftest*, *dtrain* as either training points (single-task and fine-tuning models) or context points (meta-learning models) and their predictions were evaluated on *fctest*, *dtest*. We observed that the MG-CNP generalized better than other models across the range of distances to the training set. Its performance was closely followed by that of the fine-tuned GNN, which stood out as a good candidate in the low-data regime which could be used in cases where uncertainty is not required, and which is perhaps simpler to implement than meta-learning alternatives. Finally, single-task models (FP-GP and FP-RF) and the ADKF-IFT displayed poor performance in test molecules distant to the training set, but achieved decent performance above 0.3 Tanimoto similarity.

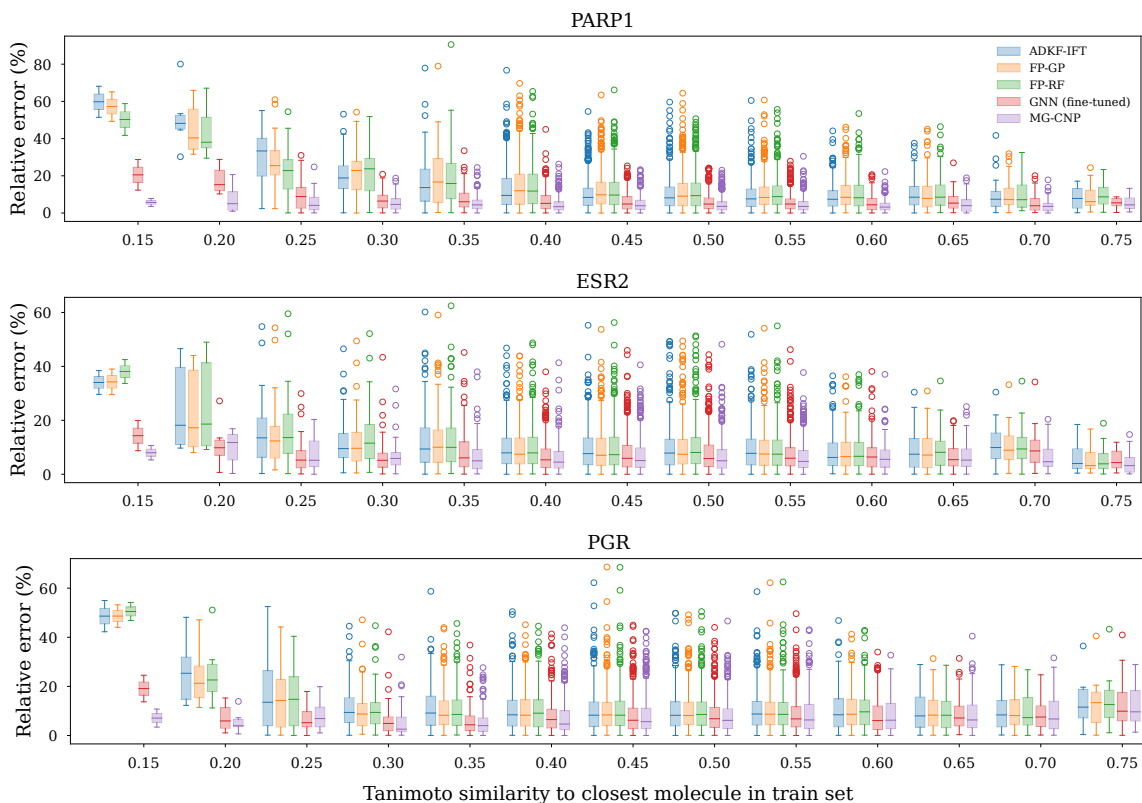


Fig. 6: Applicability domain of models within our FSL dataset. We aimed to examine the error of molecules in the test set across a range of distances to molecules in the training set. To this end, we ordered all test molecules by their Tanimoto distance to their closest neighbour in the training set, and assigned each molecule to one of 20 buckets, each of them covering a 0.05-long interval in the $[0, 1]$ range of Tanimoto distances. Then, we computed the relative error for each molecule’s prediction. Relative error RE_i of molecule i was defined as $RE_i = \left| \frac{y_i - \hat{y}_i}{y_i} \right| \cdot 100$, where y_i represents the true value and $\hat{y}_i$ the predicted one. For probabilistic models like GPs and NPs, $\hat{y}_i$ was taken to be the predicted mean μ_i . Finally, we plotted the distribution of relative errors of each model within each bucket. The MG-CNP displayed the best performance across all distances within our dataset, and it was closely followed by the fine-tuned GNN. Tanimoto similarities were computed on Morgan FPs (Section 3.7).

It is worth noting that our applicability domain results are specific to our dataset and to the size of the context and training set used in this experiment (50). Obviously, we would not expect the MG-CNP to generalize perfectly to every region of chemical space, and neither would we expect it to outperform single-task models in a high-data setting with hundreds of thousands or millions of training datapoints. Nonetheless, our results suggest that strategies that share knowledge across tasks such as meta-learning or transfer learning may prove beneficial to the applicability domain of models for molecular property prediction in the low-data setting.

4.3 Calibration of uncertainty estimates

NP predictions consist of a mean and variance value for each target point. Predictive variances can be viewed as an estimation of the confidence that the NP places on its own predictions. A model is well calibrated if, on average, lower predictive variances correlate with lower prediction errors. Figure 7 compares the calibration of MG-NPs to other models from Section 4.1. We selected 200 context datapoints at random (using the same data split as before, see Section 3.6) and ranked the 2.5k target datapoints by predicted variance from most confident (lowest predicted variance) to most uncertain (highest predicted variance). Then, we partitioned them into 100 groups of 25 datapoints, which we call confidence percentiles; lower percentiles represent higher confidence. We computed the average mean square error (MSE) and the average predicted variance within each percentile. Figure 7 shows that the highest correlation between the predicted variance and the prediction error was achieved by the MG-CNP, closely followed by the MG-LNP.

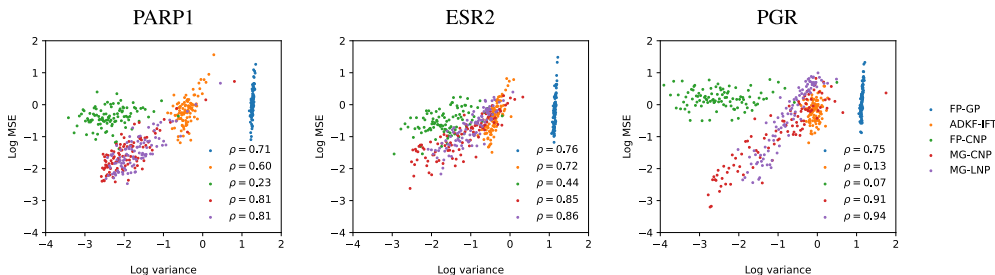


Fig. 7: Calibration of uncertainty estimates from models in Section 4.1. Each point in the scatterplot indicates the log mean MSE and log mean predicted variance in each confidence percentile (percentiles of target datapoints ranked by predicted variance). The legend in each plot shows the Pearson correlation between the log MSE and the log variance for each model. The MG-CNP achieved the best correlation on the meta-test functions PARP1, ESR2 and PGR.

In addition to comparing the calibration of MG-NPs and other models, we were also interested in the effect of fine-tuning on MG-NPs. Figure 8 examines the impact of fine-tuning on the uncertainty estimates, as well as calibration across different context

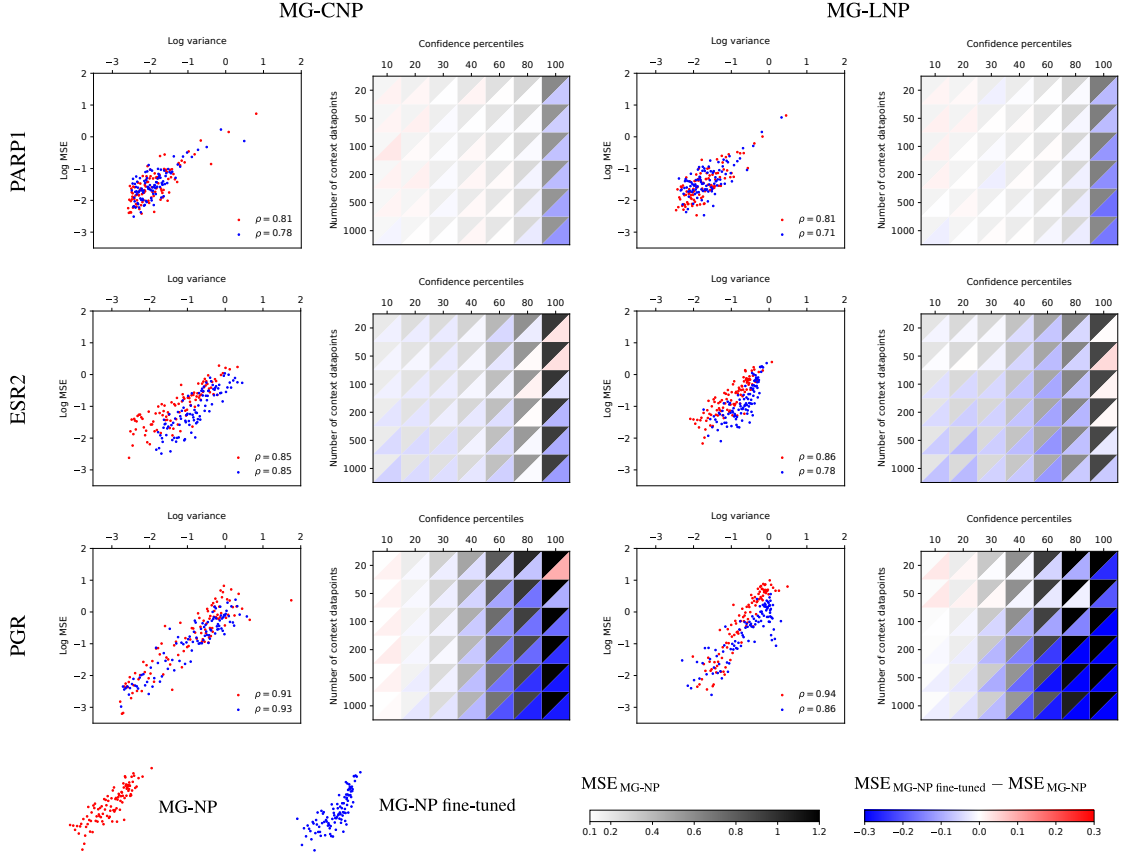


Fig. 8: Impact of fine-tuning on uncertainty estimates. Scatterplots on the left show the log mean MSE and log mean predicted variance in each confidence percentile (percentiles of target datapoints ranked by predicted variance). The right heatmaps depict the absolute MSE of MG-NPs (black and white scale; black means higher error) and the MSE difference between the fine-tuned and the unmodified models (blue and red scale; blue indicates that fine-tuned is better).

sizes. On the left columns, the scatter plots show the log mean MSE and log mean variance of an unmodified NP (i.e. without parameter adaptation, in red) and a fine-tuned NP (in blue), using 200 random context points. The Pearson correlation between predicted variance and prediction error remains high in the fine-tuned NP ($\rho > 0.7$ in all cases), suggesting good calibration. On the right, the heatmaps show the MSE of the unmodified and fine-tuned models, using context sets ranging from 20 to 1000 datapoints. The row dimension indicates context set size and the column dimension shows test datapoints grouped by confidence percentile (the first column includes the 1st to the 10th percentile, the second the 11th to the 20th, etc). In each cell, the black and white scale represents the MSE of the unmodified model. As before, we observe

that MG-NPs are well calibrated, with higher errors in the higher percentiles. Note that increasing the context set size (i.e. moving from top to bottom in the heatmaps of Figure 8) also improved performance and decreased MSE, as shown in Table 1 and in Supplementary Section D. However, the difference was small compared to the difference across confidence percentiles, so the latter dominated the black and white color scale. The red and blue scale indicates the difference between the MSE of the fine-tuned and the unmodified models. Blue cells signify that the fine-tuned model beat the unmodified one in that context size and confidence pair. In general, most fine-tuning gains came from the most erroneous unmodified predictions. This is to be expected: regions of the input space where predictions have higher loss will provide a larger signal for parameter adaptation.

Overall, our results suggest that MG-NPs are able to provide well-calibrated uncertainty estimates, and that fine-tuning for a small number of effective epochs can substantially improve mean predictions (Table 1) while maintaining good calibration (Figure 8) without overfitting to the meta-test function.

4.4 Randomization of context and target sets protects NPs from overfitting

The MG-CNP, a model with millions of parameters, exhibited good uncertainty calibration in meta-test functions (Figures 7 and 8) despite having been trained with a maximum likelihood objective (Equation 1) on as few as 2.5k molecules. Even though the CNP objective lacks a regularization term like that of the LNP, we did not find evidence that the model overfit, in the sense of underestimating posterior uncertainty. We hypothesized that this may be due to the randomization of context and target points during meta-training. The effect of this process may be two-fold: first, random sampling induces a combinatorially large number of unique views of each task or function, which may resemble an augmentation of the function set; and second, by using only a subset of all observations at each epoch, the number of effective epochs is kept low, hence reducing the risk of overfitting to any single datapoint.

In previous experiments, we trained MG-NPs using between 0.8% and 6% of observations as contexts and targets at each iteration. To investigate if random subsampling during meta-training protects NPs from overfitting, we trained a collection of MG-CNPs with the same architecture and on the same dataset as before, but using increasing fractions of observations as contexts and targets at each iteration. (Note that in this experiment we allowed overlapping of context and target sets during meta-training in order to be able to examine context and target sizes of up to 100% of the training datapoints; otherwise, if we kept context and target sets disjoint, the maximum simultaneous context and target size would be 50%). Then, we examined the performance of these models in terms of negative log predictive density (NLPD), a measure of calibration of uncertainty estimates (lower is better). We observed that as the fraction of points sampled at each training iteration grew, MG-CNPs overfit to the training molecules of the training tasks (Figure 9, left, solid lines), as evidenced by the low NLPD in these tasks and high NLPD in all other tasks (Figure 9, left, dashed lines and right, solid and dashed lines). This loss of calibration also manifested as poor correlation between the predicted uncertainty and the prediction error

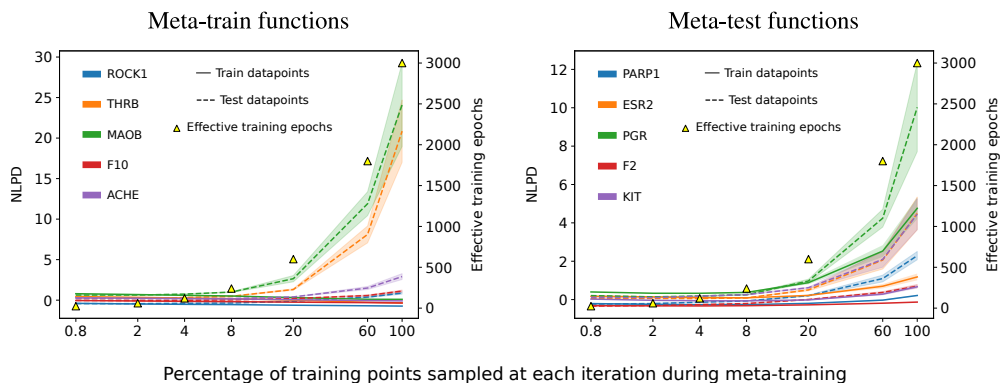


Fig. 9: Random sampling of observations during meta-training protects from overfitting. Increasing the percentage of points sampled as contexts or targets leads to less unique views of each task and to more effective epochs. This causes overfitting, with memorization of the labels from the train points of the train functions (left, solid lines) and a degradation of performance on the test points of the train functions (left, dashed line) and all points of the test functions (right, solid and dashed lines). Lower NLPD is better.

(Supplementary Figure G.2). These results suggest that, in order to achieve adequate meta-generalization and calibration, it is critical to tune the size of the context and target sets during meta-training.

4.5 Bayesian optimization (BO)

We benchmarked MG-CNP, ADKF-IFT and a GP on binary fingerprints in a sequential learning experiment using two objective functions from the DOCKSTRING optimization benchmark, druglike F2 and selective JAK2. The former encourages molecules that dock well against F2, while the latter favours molecules that dock well against JAK2 but dock poorly against LCK. Both objectives are penalized with a QED (quantitative estimate of drug-likeness) score which encourages compounds that have similar properties to orally-absorbed approved small-molecule drugs. A detailed description of the objectives is provided in Supplementary Section H. For this experiment, we created a library of 60k molecular candidates by sampling 30k from the DOCKSTRING training set and 30k from the test set. We selected molecules from the library in batches of 5 at a time, up to a total budget of 1000, using a lower confidence bound (LCB) or greedy acquisition function (the latter selects molecules according to the best predicted mean). For meta-training tasks, we created an augmented dataset of random combinations of transformed docking scores and QED (quantitative estimate of drug-likeness). To avoid data leakage across functions, we excluded from the augmentation procedure the proteins that participated in the objectives, i.e. F2, JAK2 and LCK. More details about the BO molecule library are provided in Section 3.6.

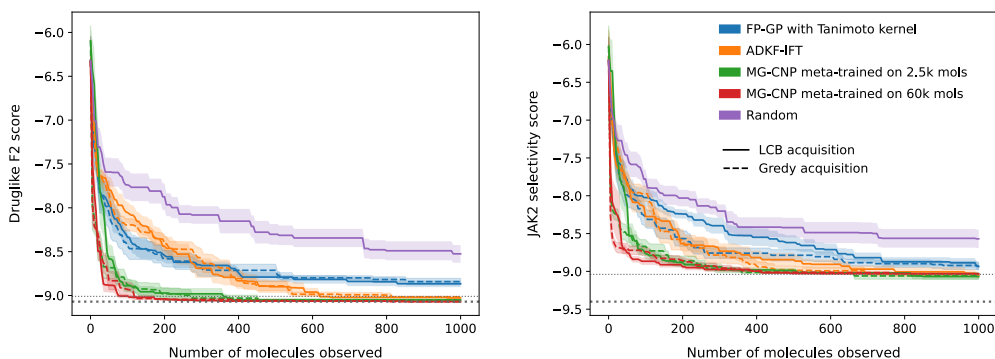


Fig. 10: Bayesian optimization of druglike F2 (left) and selective JAK2 (right). MG-CNPs always reaches either the best or a near-best molecule outperforming ADKF-IFT and GPs with a Tanimoto kernel on binary FPs. Trajectory lines indicate the mean of 10 trajectories with different random initializations, and shaded area indicates the standard error. Horizontal dotted lines indicate the best and second-best molecules in the library of 60k compounds.

We compared two MG-CNPs, one meta-trained on 2.5k molecules (Figure 10, orange) and another meta-trained on all 60k molecules from the library (green). The two MG-CNPs displayed similar BO trajectories, suggesting that NPs could be effective for molecular optimization even in the low-data setting. In druglike F2, they often reached the best molecule in the library or another with very similar score (Figure 10, left). In selective JAK2, they always reached the second-best molecule in the library (Figure 10, right). As baselines, we compared to a GP with a Tanimoto kernel on binary FPs (blue), and to random selection (red). The GP performed significantly better than random but the MG-CNP always found better molecules than the GP. Interestingly, LCB acquisition (solid line) and greedy acquisition (dashed line) yielded comparable performance, both in MG-CNPs and in the GP. While LCB with a well-calibrated model should perform better on expectation (i.e. given an infinite budget and choosing from a sufficiently large library of molecules), it may not always be beneficial in every data regime.

To ascertain whether any scaffolds or groups predominated or were favoured by the models, we visualized the median best molecule (i.e. the 5-th best within the 10 best molecules from the 10 experiment replicates) and the median 25th molecule (i.e. the 5th best molecule within the 10 25-th best molecules from the 10 experiment replicates) selected by each model for each objective function. We did not observe any obvious preference or bias, suggesting that MG-CNPs were able to adapt their predictions to the molecules in the library to find optimal candidates.

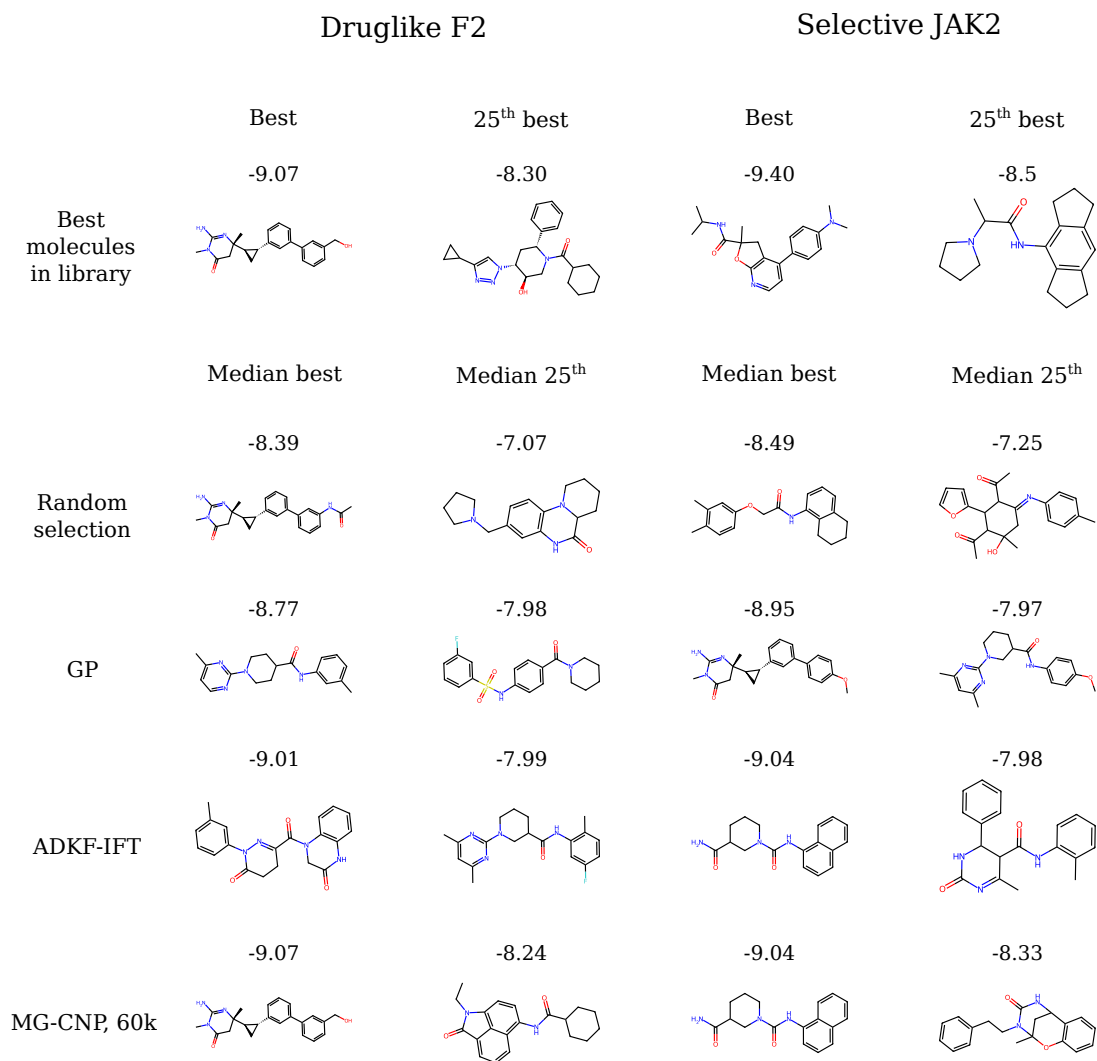


Fig. 11: Median best and 25-th best molecules selected by each model, compared to the best and 25-th molecules in the library. The models GP, ADKF-IFT and MG-CNP in this figure used a LCB acquisition.

5 Limitations and future work

In this paper, we have presented an evaluation of meta-learning with NPs for molecular property prediction. We have performed our benchmarking experiments on a subset

of the DOCKSTRING dataset of docking scores, which includes scores against 58 protein targets computed with AutoDock Vina. While the field of meta-learning for chemistry is gaining attention [4, 13–17, 39], many questions remain open about the application of meta-learning to real-world datasets, particularly with regards to data splitting (across tasks and datapoints) and to the minimum number of labels required for meta-learning to become effective. In addition, one of the advantages of NPs is that they provide uncertainty estimates; the calibration of these must be analyzed. A simulated property like docking scores was suitable to investigate these aspects because it allowed labelling arbitrary molecules and creating an optimal benchmarking dataset. In particular, the DOCKSTRING dataset enabled studying or controlling the following methodological features:

- Meta-generalization: full datapoint overlap across protein targets in DOCKSTRING enabled quantifying the distance between proteins as the Pearson correlation between their docking scores. This way, we could identify which test tasks were close or far away from the training tasks, which helped us develop parameter adaptation strategies for highly-divergent tasks.
- Molecule split: datapoint overlap allowed re-using the datapoint split in all tasks so that the same datapoints served as either contexts or targets in every task. This minimized the risk of data leakage from one task to another. In addition, we split molecules by cluster, which minimized the risk of data leakage within tasks.
- Context size: since we could choose the desired size of the DOCKSTRING subset in our experiments, we were able to compare the performance of single-task, transfer-learning and meta-learning methods over a range of context sizes (from 20 to 1,000, Section 4.1). We were also able to observe the increasing impact of fine-tuning as the number of contexts grew.
- Uncertainty calibration: predicted variance may be indicative of the true error on average but it cannot indicate the true error of each individual datapoint. Therefore, having many labelled target molecules available (2.5k in our DOCKSTRING subset) facilitated studying uncertainty calibration in Section 4.3. In particular, we grouped target points by percentiles of predicted variance and computed the average predicted variance and average error within each percentile.

These aspects, which are essential for methodological development and benchmarking, cannot be accurately controlled in experimental bioactivity datasets because they usually have few and non-overlapping labels. However, imputation of sparse datasets and bioactivity prediction are two areas where meta-learning could make a great impact. Therefore, although our work represents a significant step towards the practical deployment of NPs for molecules, restricting our experiments to simulated data is an important limitation of our study. We leave the application to real bioactivity measurements as future work. Real bioactivity datasets will require careful examination of the intra-task heterogeneity [17, 23], since in order for meta-learning to be effective in a certain task, the labels in that task should be self-consistent. For example, we do not expect meta-learning to be helpful in predicting a task which combines measurements from two different assays against the same protein, if we employ the labels from one assay as contexts and the labels from the other assay as targets. This

situation of combining different assays as a single task is a common source of noise in public bioactivity databases [23].

Another future area of research with high potential for molecular-property prediction in the low-data setting is the utilization of sequence models and large language models (LLMs) as foundation models that could be fine-tuned to novel tasks. Sequence models trained on SMILES in an unsupervised manner can produce highly informative molecular embeddings that outcompete molecular fingerprints [40]. Surprisingly, similar results have been reported for embeddings derived from general-purpose LLMs trained on natural language, even though they were not exposed to chemical tasks [41]. These large models could be treated as molecular featurizers, and their embeddings could be further improved by parameter adaptation with specialized techniques [42].

6 Discussion

Molecular property prediction poses a unique challenge to meta-learning algorithms such as NPs due to the inherent heterogeneity of molecular tasks. Using docking scores against a variety of protein targets from different families, we created a benchmarking dataset where we could control the amount of datapoints available for learning, and where we could objectively quantify the similarity or divergence between tasks in terms of the correlation of their docking scores. This controlled environment allowed us to evaluate predictions across a range of context sizes, and to investigate parameter adaptation strategies to improve meta-generalization to heterogeneous test functions. We used our benchmark to evaluate a novel NP architecture on molecular graphs, which we called the MG-NP.

Our large-scale benchmark demonstrates that MG-NPs outperform a variety of baselines in FSL, including single-task and traditional transfer-learning approaches for neural networks, probably due to the fact that they can efficiently transfer knowledge across many similar tasks simultaneously in a principled manner. In addition, we propose fine-tuning strategies which address one of the main shortcomings of NPs — the inflexible way in which training points are used in meta-testing — showing that they significantly improve meta-generalization. Fine-tuning proved highly beneficial when predicting docking scores of divergent test proteins such as PGR.

NPs are particularly attractive because they perform uncertainty quantification. We find that, despite the difference in their training loss function, both MG-CNPs and MG-LNPs provide well-calibrated uncertainty estimates, and these estimates do not deteriorate with fine-tuning. Furthermore, we show that randomization of context and target sets, and an adequate tuning of their size, are critical to calibration. Finally, we demonstrate the use of meta-learned MG-CNPs in a Bayesian optimization experiment, outperforming both more traditional models such as GPs on binary fingerprints, and other meta-learning models such as ADKF-IFT.

Overall, our work presents a novel GNN architecture for NPs, the MG-NP, that is competitive against a variety of baselines, and we suggest strategies for meta-training and meta-testing that increase the applicability of NPs for molecular property prediction.

Availability of data and materials. The source code is available from <https://github.com/mgarort/graph-nps-for-mols>.

Supplementary information. A PDF document with Supplementary Information is provided.

Declarations

The authors have nothing to declare.

References

- [1] Irwin, B.W.J., Levell, J.R., Whitehead, T.M., Segall, M.D., Conduit, G.J.: Practical Applications of Deep Learning To Impute Heterogeneous Drug Discovery Data. *J. Chem. Inf. Model.* **60**(6), 2848–2857 (2020)
- [2] ChEMBL: ChEMBL Web Services. Accessed on 9th August 2023. https://www.ebi.ac.uk/chembl/g/#search_results/all (2023)
- [3] Finn, C., Abbeel, P., Levine, S.: Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In: International Conference on Machine Learning, pp. 1126–1135 (2017). PMLR
- [4] Nguyen, C.Q., Kretsoulas, C., Branson, K.M.: Meta-learning GNN initializations for low-resource molecular property prediction. In: ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+) (2020)
- [5] Wieder, O., Kohlbacher, S., Kuenemann, M., Garon, A., Ducrot, P., Seidel, T., Langer, T.: A compact review of molecular property prediction with graph neural networks. *Drug Discovery Today: Technologies* **37**, 1–12 (2020) <https://doi.org/10.1016/j.ddtec.2020.11.009>
- [6] Liu, A., Seal, S., Yang, H., Bender, A.: Using chemical and biological data to predict drug toxicity. *SLAS Discovery* **28**(3), 53–64 (2023) <https://doi.org/10.1016/j.slasd.2022.12.003>
- [7] Bender, A., Cortes-Ciriano, I.: Artificial intelligence in drug discovery: what is realistic, what are illusions? Part 2: a discussion of chemical and biological data. *Drug Discovery Today* **26**(4), 1040–1052 (2021)
- [8] Thomas, M., Boardman, A., Garcia-Ortega, M., Hongbin, Y., Graaf, C., Bender, A.: Applications of Artificial Intelligence in Drug Design: Opportunities and Challenges. *Methods Mol. Biol.* **2390**(1-59.), (2022)
- [9] Valerie Jentzsch, M.: Costs and Causes of Oncology Drug Attrition With the Example of Insulin-Like Growth Factor-1 Receptor. *JAMA Netw. Open* **6**(7), 2324977 (2023)
- [10] Garnelo, M., Rosenbaum, D., Maddison, C., Ramalho, T., Saxton, D., Shananhan, M., Teh, Y.W., Rezende, D., Eslami, S.A.: Conditional neural processes. In: International Conference on Machine Learning, pp. 1704–1713 (2018). PMLR
- [11] Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D.J., Eslami, S.M.A., Teh, Y.W.: Neural Processes. In: ICML 2018 Workshop on Theoretical Foundations and Applications of Deep Generative Models (2018)

- [12] Antoniou, A., Edwards, H., Storkey, A.: How to train your MAML. In: International Conference on Learning Representations (2019)
- [13] Chen, W., Tripp, A., Hernández-Lobato, J.M.: Meta-learning adaptive deep kernel gaussian processes for molecular property prediction. In: International Conference on Learning Representations (2023)
- [14] Stanley, M., Bronskill, J.F., Maziarz, K., Misztela, H., Lanini, J., Segler, M., Schneider, N., Brockschmidt, M.: FS-mol: A few-shot learning dataset of molecules. In: Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2) (2021)
- [15] Lee, E., Yoo, J., Lee, H., Hong, S.: MetaDTA: Meta-learning-based drug-target binding affinity prediction. In: ICLR 2022 Machine Learning for Drug Discovery Workshop (2022)
- [16] Garcia-Ortegon, M., Bender, A., Bacallado, S.: Conditional neural processes for molecules. In: NeurIPS 2022 Workshop on Meta-Learning (2022)
- [17] Chan, L., Verdonk, M., Poelking, C.: Embracing assay heterogeneity with neural processes for markedly improved bioactivity predictions (2023)
- [18] Walter, M., Allen, L.N., León, A., Webb, S.J., Gillet, V.J.: Analysis of the benefits of imputation models over traditional QSAR models for toxicity prediction. *J. Cheminf.* **14**(1), 1–27 (2022)
- [19] Luukkonen, S., Meijer, E., Tricarico, G.A., Hofmans, J., Stouten, P.F.W., Westen, G.J.P., Lenselink, E.B.: Large-Scale Modeling of Sparse Protein Kinase Activity Data. *J. Chem. Inf. Model.* **63**(12), 3688–3696 (2023)
- [20] Wikberg, J.E.S., Mutulis, F., Mutule, I., Veiksina, S., Lapinsh, M., Petrovska, R., Prusis, P.: Melanocortin Receptors: Ligands and Proteochemometrics Modeling. *Ann. N.Y. Acad. Sci.* **994**(1), 21–26 (2003)
- [21] Bongers, B.J., IJzerman, Adriaan.P., Van Westen, G.J.P.: Proteochemometrics: recent developments in bioactivity and selectivity modeling. *Drug Discovery Today: Technologies* **32-33**, 89–98 (2019)
- [22] García-Ortegón, M., Simm, G.N.C., Tripp, A.J., Hernández-Lobato, J.M., Bender, A., Bacallado, S.: DOCKSTRING: Easy Molecular Docking Yields Better Benchmarks for Ligand Design. *J. Chem. Inf. Model.* **62**(15), 3486–3502 (2022)
- [23] Landrum, G.A., Riniker, S.: Combining IC50 or Ki Values from Different Sources Is a Source of Significant Noise. *J. Chem. Inf. Model.* **2024** (2024) <https://doi.org/10.1021/acs.jcim.4c00049>
- [24] Visser, U., Abeyruwan, S., Vempati, U., Smith, R.P., Lemmon, V., Schürer,

- S.C.: BioAssay Ontology (BAO): a semantic description of bioassays and high-throughput screening results. *BMC Bioinf.* **12**(1), 1–16 (2011) <https://doi.org/10.1186/1471-2105-12-257>
- [25] Vinod, R., Chen, P.-Y., Das, P.: Reprogramming Pretrained Language Models for Protein Sequence Representation Learning (2023)
 - [26] Formont, P., Jeannin, H., Piantanida, P., Ayed, I.B.: Is Meta-training Really Necessary for Molecular Few-Shot Learning ? *arXiv* (2024) <https://doi.org/10.48550/arXiv.2404.02314> 2404.02314
 - [27] Xiong, Z., Wang, D., Liu, X., Zhong, F., Wan, X., Li, X., Li, Z., Luo, X., Chen, K., Jiang, H., Zheng, M.: Pushing the Boundaries of Molecular Representation for Drug Discovery with the Graph Attention Mechanism. *J. Med. Chem.* **63**(16), 8749–8760 (2020)
 - [28] Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. In: *NeurIPS 2014 Deep Learning and Representation Learning Workshop* (2014)
 - [29] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: *Advances in Neural Information Processing Systems*, vol. 30 (2017)
 - [30] Sun, J., Jeliaskova, N., Chupakhin, V., Golib-Dzib, J.-F., Engkvist, O., Carlsson, L., Wegner, J., Ceulemans, H., Georgiev, I., Jeliaskov, V., Kochev, N., Ashby, T.J., Chen, H.: ExCAPE-DB: an integrated large scale dataset facilitating Big Data analysis in chemogenomics. *J. Cheminf.* **9**(1), 1–9 (2017) <https://doi.org/10.1186/s13321-017-0203-5>
 - [31] Mysinger, M.M., Carchia, M., Irwin, John.J., Shoichet, B.K.: Directory of Useful Decoys, Enhanced (DUD-E): Better Ligands and Decoys for Better Benchmarking. *J. Med. Chem.* **55**(14), 6582–6594 (2012) <https://doi.org/10.1021/jm300687e>
 - [32] Wu, Z., Ramsundar, B., Feinberg, E.N., Gomes, J., Geniesse, C., Pappu, A.S., Leswing, K., Pande, V.: MoleculeNet: a benchmark for molecular machine learning. *Chem. Sci.* **9**(2), 513–530 (2018) <https://doi.org/10.1039/C7SC02664A>
 - [33] Sheridan, R.P.: Time-Split Cross-Validation as a Method for Estimating the Goodness of Prospective Prediction. *J. Chem. Inf. Model.* **53**(4), 783–790 (2013)
 - [34] Martin, E.J., Polyakov, V.R., Tian, L., Perez, R.C.: Profile-QSAR 2.0: Kinase Virtual Screening Accuracy Comparable to Four-Concentration IC50s for Realistically Novel Compounds. *J. Chem. Inf. Model.* **57**(8), 2077–2088 (2017)
 - [35] Simm, J., Humbeck, L., Zalewski, A., Sturm, N., Heyndrickx, W., Moreau, Y.,

- Beck, B., Schuffenhauer, A.: Splitting chemical structure data sets for federated privacy-preserving machine learning. *J. Cheminf.* **13**(1), 1–14 (2021)
- [36] Morgan, H.L.: The Generation of a Unique Machine Description for Chemical Structures-A Technique Developed at Chemical Abstracts Service. *J. Chem. Doc.* **5**(2), 107–113 (1965)
- [37] RDKit: RDKit: Open-source cheminformatics. <https://www.rdkit.org>
- [38] Weininger, D.: SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *J. Chem. Inf. Comput. Sci.* **28**(1), 31–36 (1988)
- [39] Garcia-Ortegon, M., Seal, S., Singh, S., Bender, A., Bacallado, S.: Calibrated prediction of scarce adverse drug reaction labels with conditional neural processes. *bioRxiv*, 2024–0607598036 (2024) [2024.06.07.598036](https://doi.org/10.1101/2024.06.07.598036)
- [40] Ross, J., Belgodere, B., Chenthamarakshan, V., Padhi, I., Mroueh, Y., Das, P.: Large-scale chemical language representations capture molecular structure and properties. *Nat. Mach. Intell.* **4**, 1256–1264 (2022) <https://doi.org/10.1038/s42256-022-00580-7>
- [41] Sadeghi, S., Bui, A., Forooghi, A., Lu, J., Ngom, A.: Can large language models understand molecules? *BMC Bioinf.* **25**(1), 1–17 (2024) <https://doi.org/10.1186/s12859-024-05847-x>
- [42] Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: LoRA: Low-rank adaptation of large language models. In: International Conference on Learning Representations (2022). <https://openreview.net/forum?id=nZeVKeeFYf9>

Supplementary information for

“Graph neural processes for molecules: An evaluation on docking scores and strategies to improve meta-generalization”

A Neural processes (NPs)

A.1 Parameterization of CNP and LNP

NPs assume conditional independence between the targets and a Gaussian predictive distribution with mean $\mu_\theta(x)$ and variance $\sigma_\theta^2(x)$:

$$q(y_t \mid x_c, y_c; x_t) = \prod_{j=1}^T \mathcal{N}(y_{t,j} ; \mu_\theta(x_{t,j}, x_c, y_c), \sigma_\theta^2(x_{t,j}, x_c, y_c)).$$

The conditional NP (CNP) [1] and the latent NP (LNP) [2] parameterize the predictive mean and variance of an input x in three steps:

- *Datapoint encoder*: a neural network h_θ maps each context point $(x_{c,j}, y_{c,j})$ to a local datapoint representation r_j .

$$r_j = h_\theta(x_{c,j}, y_{c,j})$$

- *Function encoder*: all context encodings r_j are combined into a global context encoding r through a commutative operation $\oplus$, usually the sum or the mean. Commutativity of the function encoder guarantees invariance to permutations of the context set.

$$r = r_1 \oplus \dots \oplus r_C$$

- *Decoder*: a neural network g_θ maps the function encoding r and the input location x to the predictive mean and variance. In the CNP, the decoding process is deterministic:

$$(\mu_\theta(x, x_c, y_c), \sigma_\theta^2(x, x_c, y_c)) = g_\theta(r, x).$$

In the LNP, the decoding process involves sampling a latent variable. Its approximate posterior q_ϕ is parameterized as a Gaussian with mean and variance given by a neural network g_ϕ .

$$z \sim q_\phi(z \mid x_c, y_c) := \mathcal{N}(\mu_\phi, \sigma_\phi^2), \quad (\mu_\phi, \sigma_\phi^2) = g_\phi(r)$$

The final decoding stage continues like the CNP, but using z instead of r :

$$(\mu_\theta(x, x_c, y_c), \sigma_\theta^2(x, x_c, y_c)) = g_\theta(z, x).$$

A.2 Objective functions of neural processes (NPs)

The CNP objective $\mathcal{L}_\theta$ is the conditional log likelihood of the targets given the contexts [1].

$$\mathcal{L}_\theta(y_t \mid x_c, y_c; x_t) = \log q_\theta(y_t \mid x_c, y_c; x_t)$$

The LNP objective $\mathcal{L}_{\theta, \phi}$ consists of a reconstruction and a regularization term [2]:

$$\mathcal{L}_{\theta, \phi}(y_t \mid x_c, y_c; x_t) = \mathbb{E}_{q_\phi(z \mid x_d, y_d)} [\log p_\theta(y_t \mid z, x_t)] - D_{\text{KL}}(q_\phi(z \mid x_d, y_d) \parallel q_\phi(z \mid x_c, y_c)),$$

where $x_d = (x_c, x_t)$, $y_d = (x_c, y_t)$ and D_{KL} is the KL divergence. It is an approximation of an evidence lower bound (ELBO) to the conditional log marginal likelihood:

$$\log p_\theta(y_t \mid x_d, y_c) \geq \mathbb{E}_{q_\phi(z \mid x_d, y_d)} [\log p_\theta(y_t \mid z, x_t)] - D_{\text{KL}}(q_\phi(z \mid x_d, y_d) \parallel p_\theta(z \mid x_c, y_c)).$$

The ELBO is intractable due to the posterior $p_\theta(z \mid x_c, y_c)$ in the KL term. The LNP objective is derived by approximating the posterior with $q_\phi(z \mid x_c, y_c)$. This way the objective becomes tractable but $\mathcal{L}_{\theta, \phi}$ is no longer an analytical lower bound.

B Number of observations per task in bioactivity datasets

FSL experiments in other domains (e.g. image classification) often focus on an extremely low number of context points, rarely benchmarking more than 100 contexts. In the context of drug discovery, however, the amount of observations per function in bioactivity datasets can vary widely, with many functions of interest having more than 100 labels. For example, in Figure B.1 we show the distribution of observations registered in protein binding affinity assays in ChEMBL33 [3]. Whether meta-learning can provide benefits in this data regime is an open research question. For this reason, we decided to investigate a wide range of context points, from 20 to 1000 (Table 1 and Supplementary Section D).

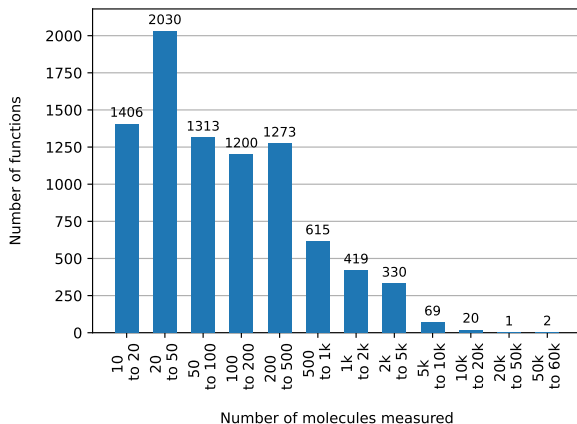


Fig. B.1: Distribution of observations per function in the protein binding subset of ChEMBL33

C Few-shot learning with MAML

C.1 Parameter adaptation with MAML

Model-agnostic meta-learning (MAML) [4] is a meta-training approach to find model parameters that can be rapidly adapted to test functions in one or more steps of gradient-descent. At each meta-training iteration t , MAML simulates an adaptation experiment in two stages: the inner update (the simulated adaptation) and the outer update (the actual update of the model’s parameters). First, during the inner update, it samples a set of support points from a test function, computes the loss of their predictions (inner loss), and takes a single step of gradient descent to adapt the model’s parameters from θ_t to θ'_t . Second, during the outer update, MAML samples another set of query points from the same function, computes the prediction loss on those points using θ'_t (outer loss), and backpropagates through the inner update to find new parameters θ_{t+1} . Since the inner update involves a step of gradient descent, the outer update involves computing a gradient through a gradient, i.e. computing the Hessian of the parameters.

Similar to the support and query sets in MAML, meta-training NPs entails sampling two sets of observations: the contexts and the targets. The support set in MAML informs the prediction of the query set, analogous to how the context set in NPs informs the prediction of targets. We can meta-train NPs with MAML by computing the inner loss on the context points and the outer loss on the target points. At each iteration t , during the inner update a prediction on the contexts of a train function f_i is made, with loss $\mathcal{L}_{\psi_t}(y_c^i | x_c^i, y_c^i; x_c^i)$. A single step of gradient descent is taken to adapt the parameters from ψ_t to ψ'_t . During the outer update, a prediction on the targets is made using the adapted parameters ψ'_t , with loss $\mathcal{L}_{\psi'_t}(y_t^i | x_c^i, y_c^i; x_t^i)$. Finally, we backpropagate through the inner update to find new parameters ψ_{t+1} . Later, at meta-testing, we adapt NPs by taking a single iteration of gradient descent on the loss of the contexts of f , $\mathcal{L}_{\psi}(y_c | x_c, y_c; x_c)$.

Computing the Hessian of all MG-NP parameters was memory-prohibitive, so we only applied MAML to some layers, similar to Raghu et al. [5]. In particular, we applied it to the same layers adapted in the fine-tuning experiments. During meta-training, these layers underwent the inner and outer update cycle, while other layers experienced a single update per iteration, as usual. Later, during meta-testing, the MAML layers were adapted and the rest were frozen. In addition, backpropagating through gradients can lead to training instability, so we implemented modifications from Antoniou et al. [6] to increase robustness.

C.2 MAML implementation

In all experiments involving MAML (whether single-task GNN or NPs), we used a single step of gradient descent in the inner update with a learning rate of 10^{-3} . We attempted other rates to try to increase training stability and decrease MAML’s error bars but found this to be the most stable. In addition, our MAML training regime borrowed tricks from Antoniou et al. [6] to improve robustness. In particular, we implement derivative-order annealing such that the first 50 epochs of training do

first-order MAML and the rest do second-order MAML. We also implement cosine annealing of the outer learning rate. The tricks of multi-step loss optimization and per-step batch normalization are not applicable to our model since we performed a single gradient descent iteration in the inner loop. Due to memory constraints, we follow Raghu et al. [5] and only apply MAML the last two layers of the encoder or the decoder of NPs, as explained in Section 3.4. Similarly, we only train the last two layers of single-task GNNs meta-trained with MAML.

C.3 MAML few-shot learning results

Few-shot learning (FSL) results for models trained with a MAML regime are shown in the extended FSL results (Supplementary Section D).

D Extended few-shot learning (FSL) results

	PARP1 (high correlation)						ESR2 (medium correlation)						PGR (low correlation)					
	Number of context points						Number of context points						Number of context points					
	20	50	100	200	500	1000	20	50	100	200	500	1000	20	50	100	200	500	1000
Single task	Dummy regressor	-13.6 (3.9)	-18.9 (3.0)	-19.6 (2.9)	-21.4 (1.7)	-21.9 (1.1)	-21.4 (0.6)	-5.5 (1.9)	-3.5 (1.1)	-3.0 (1.1)	-3.5 (0.8)	-3.3 (0.4)	-3.1 (0.3)	-13.3 (4.6)	-5.3 (1.9)	-2.7 (0.8)	-1.6 (0.4)	-1.5 (0.2)
	FP-RF	0.4 (2.1)	-1.2 (2.3)	3.0 (2.5)	15.1 (1.9)	23.7 (0.8)	30.9 (0.5)	-5.3 (3.0)	6.6 (2.0)	11.2 (1.4)	15.4 (1.4)	21.0 (0.5)	25.1 (0.2)	-13.2 (8.7)	-4.7 (2.2)	-0.3 (1.4)	2.0 (1.3)	8.9 (0.8)
	FP-GP	6.4 (0.7)	9.5 (1.7)	15.6 (2.1)	29.6 (1.3)	47.5 (0.7)	58.1 (0.3)	2.0 (1.0)	8.3 (1.3)	14.0 (1.3)	19.9 (0.9)	26.7 (0.4)	31.8 (0.4)	-11.3 (4.9)	1.0 (1.8)	6.2 (0.9)	8.4 (0.7)	8.9 (0.6)
	GNN	22.9 (11.9)	21.0 (13.9)	63.9 (4.2)	61.7 (9.0)	73.2 (0.6)	75.4 (0.5)	-10.7 (4.9)	-7.4 (3.6)	1.8 (3.9)	6.4 (3.9)	14.8 (3.5)	19.8 (4.2)	-56.6 (33.0)	-13.0 (6.6)	4.4 (2.6)	-1.9 (4.8)	17.4 (3.0)
Transfer learning	GNN (random)	63.8 (1.4)	69.9 (1.3)	70.7 (2.5)	73.7 (1.3)	76.2 (0.7)	77.6 (1.3)	5.7 (3.2)	19.1 (2.1)	10.8 (4.9)	17.8 (2.5)	24.3 (2.0)	29.2 (1.6)	-21.2 (16.9)	14.5 (2.6)	9.9 (3.8)	13.5 (3.9)	23.3 (1.6)
	GNN (fine-tuned)	75.0 (0.9)	77.3 (0.6)	77.9 (0.3)	76.2 (0.4)	75.4 (0.3)	77.1 (0.2)	34.9 (1.5)	37.8 (0.8)	35.2 (1.1)	30.3 (1.2)	30.3 (1.0)	34.2 (0.6)	21.1 (1.8)	26.4 (1.2)	23.7 (1.4)	16.3 (1.1)	19.4 (1.0)
Meta-learning	GNN (MAML)	51.6 (8.9)	52.2 (9.0)	52.0 (9.0)	51.0 (9.8)	50.9 (9.8)	51.3 (9.8)	11.5 (11.1)	12.7 (11.0)	11.3 (10.5)	10.8 (10.3)	9.8 (10.4)	11.1 (10.6)	-31.2 (7.7)	-26.6 (6.2)	-28.4 (6.3)	-30.5 (6.4)	-32.4 (6.8)
	ADKF-IFT	10.8 (3.4)	28.2 (1.8)	38.8 (1.5)	47.5 (1.1)	55.2 (0.6)	61.9 (0.3)	-0.3 (1.4)	8.1 (1.3)	13.9 (0.9)	18.0 (1.1)	24.6 (0.7)	29.4 (0.4)	-6.6 (4.1)	1.7 (1.4)	4.5 (0.7)	5.2 (0.8)	8.7 (0.9)
	FP-CNP	55.5 (0.7)	55.5 (0.5)	55.4 (0.5)	55.2 (0.4)	55.4 (0.3)	55.3 (0.4)	29.0 (0.8)	26.0 (2.0)	27.2 (1.3)	28.8 (0.7)	29.1 (0.6)	29.7 (0.5)	-36.7 (4.7)	-37.5 (4.5)	-39.1 (3.8)	-39.7 (2.3)	-42.4 (2.9)
	MG-CNP	81.6 (0.6)	82.7 (0.3)	83.0 (0.1)	83.1 (0.1)	83.1 (0.1)	83.0 (0.2)	43.2 (1.0)	44.9 (0.7)	45.3 (0.6)	45.6 (0.5)	46.3 (0.5)	46.6 (0.6)	19.1 (2.5)	23.2 (1.7)	25.6 (1.4)	27.5 (1.5)	27.2 (1.3)
	MG-LNP	81.4 (0.3)	81.5 (0.3)	81.6 (0.3)	81.7 (0.3)	82.0 (0.3)	82.3 (0.3)	39.9 (1.3)	40.0 (1.3)	40.1 (1.2)	40.3 (1.2)	41.0 (1.0)	41.5 (0.9)	2.4 (3.7)	3.0 (3.7)	3.8 (3.7)	5.5 (3.8)	10.6 (3.7)
NPs with parameter adaptation	FP-CNP (MAML)	57.8 (0.5)	58.0 (0.6)	58.3 (0.4)	58.2 (0.3)	58.2 (0.3)	58.3 (0.3)	25.8 (2.6)	24.5 (2.0)	24.7 (1.4)	26.7 (1.3)	26.8 (1.0)	27.9 (0.7)	-26.4 (6.0)	-19.0 (2.3)	-19.0 (1.8)	-20.7 (1.6)	-19.9 (1.6)
	FP-CNP (fine-tuned)	55.3 (0.8)	55.2 (0.7)	55.7 (0.5)	55.7 (0.4)	56.1 (0.3)	56.2 (0.2)	30.4 (0.8)	29.5 (1.2)	29.9 (0.9)	31.4 (0.7)	33.3 (0.3)	34.0 (0.4)	-24.5 (5.2)	-13.7 (1.5)	-9.2 (1.3)	-5.7 (1.5)	1.6 (1.1)
	FP-LNP (MAML)	42.1 (0.6)	42.3 (0.8)	42.5 (0.7)	42.4 (0.7)	42.5 (0.8)	42.4 (0.8)	6.8 (5.4)	8.0 (4.6)	8.2 (4.2)	6.9 (4.3)	7.4 (4.2)	7.3 (4.2)	-24.1 (7.9)	-21.2 (6.7)	-21.0 (6.4)	-21.6 (6.5)	-21.3 (6.3)
	FP-LNP (fine-tuned)	38.8 (1.3)	36.0 (1.4)	37.6 (1.4)	40.3 (1.2)	41.6 (1.2)	41.2 (0.7)	29.6 (0.6)	30.6 (0.6)	29.8 (0.5)	29.0 (0.8)	30.4 (0.6)	29.8 (0.7)	-1.2 (3.4)	3.6 (0.6)	7.7 (0.8)	8.3 (0.6)	7.3 (0.9)
	MG-CNP (MAML)	82.1 (0.5)	83.3 (0.3)	83.5 (0.2)	83.7 (0.2)	83.7 (0.2)	83.7 (0.2)	44.0 (0.9)	44.9 (0.9)	44.8 (1.2)	45.9 (0.5)	46.7 (0.5)	47.0 (0.4)	22.2 (2.2)	29.7 (1.5)	31.3 (1.6)	32.6 (1.4)	33.3 (1.1)
	MG-CNP (fine-tuned)	82.1 (0.5)	83.4 (0.2)	83.7 (0.2)	83.9 (0.2)	84.4 (0.2)	84.9 (0.1)	45.0 (0.9)	45.3 (0.8)	46.9 (0.8)	49.7 (0.4)	51.3 (0.4)	52.5 (0.4)	20.3 (3.9)	31.9 (1.4)	36.6 (1.2)	40.7 (0.8)	43.7 (0.2)
	MG-LNP (MAML)	80.8 (0.6)	80.7 (0.6)	80.8 (0.6)	80.9 (0.6)	81.2 (0.5)	81.5 (0.5)	38.4 (1.3)	39.5 (0.9)	39.9 (0.9)	39.9 (0.9)	40.5 (0.8)	41.1 (0.7)	10.2 (3.1)	11.5 (2.8)	12.2 (2.7)	13.2 (2.7)	16.0 (2.4)
	MG-LNP (fine-tuned)	82.5 (0.4)	82.3 (0.5)	83.0 (0.3)	83.5 (0.3)	84.2 (0.2)	84.7 (0.2)	43.3 (0.9)	42.8 (0.9)	46.2 (0.6)	47.3 (0.7)	49.5 (0.4)	50.4 (0.6)	10.8 (9.3)	7.7 (6.5)	18.3 (3.7)	31.6 (2.0)	40.1 (1.5)

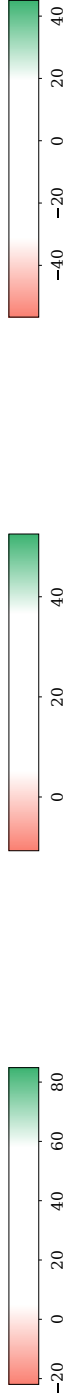


Table D.1: Coefficient of determination (R^2) few-shot learning results for all models, including those trained with MAML (Section C). Each cell shows the mean and standard error (in parentheses) of 10 random repetitions.

D Extended few-shot learning (FSL) results

Single task	PARP1 (high correlation)						ESR2 (medium correlation)						PGR (low correlation)					
	Number of context points						Number of context points						Number of context points					
	20	50	100	200	500	1000	20	50	100	200	500	1000	20	50	100	200	500	1000
Meta-learning	FP-GP	0.85 (0.01)	0.71 (0.01)	0.64 (0.01)	0.55 (0.01)	0.40 (0.01)	0.28 (0.00)	0.52 (0.01)	0.40 (0.01)	0.35 (0.01)	0.31 (0.00)	0.26 (0.00)	0.22 (0.00)	0.64 (0.04)	0.49 (0.02)	0.43 (0.01)	0.41 (0.01)	0.38 (0.00)
	ADKF-IFT	0.73 (0.04)	0.57 (0.02)	0.48 (0.02)	0.39 (0.01)	0.31 (0.01)	0.22 (0.00)	0.41 (0.01)	0.36 (0.01)	0.32 (0.00)	0.29 (0.01)	0.25 (0.00)	0.21 (0.00)	0.52 (0.03)	0.44 (0.01)	0.42 (0.01)	0.41 (0.01)	0.36 (0.00)
	FP-CNP	3.61 (0.11)	3.50 (0.11)	3.52 (0.07)	3.49 (0.06)	3.49 (0.05)	3.50 (0.06)	1.69 (0.06)	1.86 (0.10)	1.76 (0.07)	1.65 (0.04)	1.63 (0.04)	1.60 (0.04)	7.29 (1.16)	8.03 (1.17)	7.83 (0.79)	7.44 (0.45)	8.16 (0.58)
	MG-CNP	0.83 (0.06)	0.78 (0.02)	0.75 (0.01)	0.74 (0.01)	0.74 (0.01)	0.74 (0.01)	1.04 (0.01)	1.00 (0.01)	0.98 (0.01)	0.98 (0.01)	0.97 (0.01)	0.97 (0.01)	1.18 (0.02)	1.13 (0.02)	1.09 (0.01)	1.07 (0.01)	1.08 (0.01)
	MG-LNP	0.75 (0.01)	0.75 (0.01)	0.75 (0.01)	0.75 (0.01)	0.74 (0.01)	0.74 (0.01)	1.08 (0.01)	1.07 (0.01)	1.07 (0.01)	1.07 (0.01)	1.06 (0.01)	1.05 (0.01)	1.28 (0.03)	1.27 (0.03)	1.27 (0.03)	1.25 (0.03)	1.21 (0.03)
NPs with parameter adaptation	FP-CNP (MAML)	3.42 (0.17)	3.35 (0.12)	3.31 (0.08)	3.27 (0.06)	3.18 (0.05)	3.19 (0.03)	1.69 (0.10)	1.77 (0.13)	1.70 (0.10)	1.60 (0.05)	1.57 (0.03)	1.53 (0.02)	4.60 (0.52)	5.18 (0.49)	5.13 (0.40)	5.09 (0.29)	5.23 (0.26)
	FP-CNP (fine-tuned)	3.76 (0.84)	3.20 (0.25)	3.46 (0.27)	3.49 (0.22)	3.46 (0.10)	3.48 (0.11)	1.18 (0.01)	1.20 (0.02)	1.16 (0.01)	1.16 (0.01)	1.17 (0.01)	1.21 (0.01)	2.36 (0.28)	2.23 (0.15)	1.69 (0.04)	1.49 (0.01)	1.45 (0.01)
	FP-LNP (MAML)	1.44 (0.02)	1.43 (0.02)	1.43 (0.02)	1.43 (0.02)	1.43 (0.02)	1.43 (0.02)	1.32 (0.04)	1.30 (0.03)	1.30 (0.03)	1.31 (0.03)	1.31 (0.03)	1.31 (0.03)	1.49 (0.04)	1.46 (0.03)	1.46 (0.03)	1.46 (0.03)	1.46 (0.03)
	FP-LNP (fine-tuned)	2.14 (0.08)	1.87 (0.02)	2.68 (0.05)	3.24 (0.09)	3.52 (0.09)	3.66 (0.11)	1.21 (0.02)	1.15 (0.00)	1.20 (0.01)	1.22 (0.01)	1.27 (0.01)	1.34 (0.01)	1.37 (0.02)	1.34 (0.01)	1.32 (0.01)	1.32 (0.00)	1.46 (0.02)
	MG-CNP (MAML)	0.80 (0.03)	0.75 (0.02)	0.73 (0.01)	0.72 (0.00)	0.71 (0.00)	0.71 (0.00)	1.03 (0.01)	1.00 (0.01)	0.99 (0.01)	0.98 (0.01)	0.97 (0.01)	0.96 (0.01)	1.14 (0.02)	1.06 (0.01)	1.04 (0.01)	1.03 (0.01)	1.02 (0.01)
	MG-CNP (fine-tuned)	0.95 (0.08)	0.84 (0.05)	0.79 (0.03)	0.76 (0.01)	0.73 (0.01)	0.70 (0.01)	0.98 (0.01)	0.95 (0.01)	0.94 (0.01)	0.91 (0.01)	0.89 (0.01)	0.87 (0.01)	1.15 (0.02)	1.06 (0.01)	1.03 (0.01)	1.00 (0.01)	0.96 (0.01)
	MG-LNP (MAML)	0.78 (0.02)	0.78 (0.02)	0.78 (0.02)	0.78 (0.02)	0.77 (0.02)	0.76 (0.02)	1.05 (0.01)	1.04 (0.01)	1.04 (0.01)	1.04 (0.01)	1.03 (0.00)	1.02 (0.01)	1.20 (0.02)	1.19 (0.02)	1.19 (0.02)	1.18 (0.02)	1.14 (0.02)
	MG-LNP (fine-tuned)	0.91 (0.07)	0.81 (0.03)	0.77 (0.02)	0.74 (0.01)	0.72 (0.01)	0.69 (0.01)	1.04 (0.02)	1.01 (0.01)	0.96 (0.01)	0.95 (0.01)	0.92 (0.01)	0.91 (0.01)	1.30 (0.11)	1.29 (0.07)	1.22 (0.05)	1.06 (0.01)	1.01 (0.01)



Table D.2: Negative log predictive likelihood (NLPD) few-shot learning results for all probabilistic models, including those trained with MAML (Section C). Each cell shows the mean and standard error (in parentheses) of 10 random repetitions.

E Model implementation

In every experiment we trained every model 10 times with different random initializations. We report the mean and standard error of these repetitions. The number of contexts in FSL experiments, as well as the initial points in the BO experiment, were also sampled randomly in each repetition.

The subset of points used for training each type of model (single task, transfer learning and meta-learning) is detailed in Section 3.6.

All neural models were implemented in Pytorch [7] and trained with the Adam optimizer using a learning rate of 10^{-3} , weight decay of 10^{-5} and a cosine annealing scheduler.

E.1 Single-task models

- *Random forest (RF)*. Random forest regressors were implemented in scikit-learn [8] using the default parameters.
- *Gaussian process (GP)*. We implemented exact GPs in GPytorch [9], using binary fingerprints and a Tanimoto kernel. The noise parameter was adjusted automatically in GPytorch by evidence maximization for 2000 epochs. Automatic relevance determination (ARD) to find an optimal lengthscale for each feature was attempted but eventually rejected, since the large number of fingerprint features led to gross overfitting.
- *Graph neural networks (GNNs)*. Our GNN is a 6-layer fully-connected neural network (FNN) on top of the the MGAE, with maximum layer width of 1000 hidden neurons and layer normalization between each layer. We chose 6 layers to promote a fair comparison to our MG-NP models, which had an encoder and decoder of 3 layers each. GNNs were trained for 1000 epochs with a batch size of 50.

E.2 Transfer-learning models

- *GNN with random initialization (GNN random)*. This model had the same architecture as the GNN in the previous section, but instead of training all layers, the GNN was initialized with random weights and only the last two layers were trained for 1000 epochs, while the rest were kept frozen, in a way reminiscent of a transfer learning experiment. In this way, the randomly-initialized layers worked as feature extractors.
- *GNN pre-trained and fine-tuned (GNN fine-tuned)*. This model had the same architecture as the GNN but was pre-trained as a multi-task model on the 53 meta-train proteins. Then, the last layer was changed (to make it single-task) and the last two layers were trained for 1000 epochs while the rest were kept frozen.

E.3 Meta-learning models

E.3.1 Neural processes

- *Conditional neural processes (CNPs)*. The FP-CNP had an encoder and decoder FNN of 3 layers each, with a maximum layer width of 1000, layer normalization between every layer, and an encoding size of 250. Layer normalization was chosen instead of batch normalization because, due to memory constraints, our batch size was just 2 functions. We choose the mean as commutative operation. The MG-CNP had the same architecture but built its encoder on top of the MGAE.
- *Latent neural processes (LNPs)*. The FP-LNP had the same encoder and decoder as the CNP, and an additional network to parameterize the mean and variance of the latent variable. This network was also a FNN of 3 layers with layer normalization. The size of the latent variable was 250. Again, we chose the mean as the commutative operation for aggregation. The MG-LNP had the same architecture but it built its encoder on top of the MGAE.
- *Meta-training*. We trained using between 20 and 150 contexts and targets at each iteration. The amount of contexts and targets was itself sampled uniformly between 20 and 150. The contexts and targets were always disjoint, with the only exception of the experiment in Section 4.4 where they were allowed to overlap in order to be able to increase the size of both contexts and targets to 100% of the training set, thus increasing the number of effective epochs. In the FSL experiments, we trained for 3000 epochs on the 53 train proteins with a batch size of 2 functions per batch. In the BO experiments we trained for 1000 epochs on the augmented train set of combinations of docking scores with a batch size of 8.
- *Adaptation during meta-testing*. The adaptation procedures with fine-tuning and MAML and fine-tuning are described in Sections 3.4 and C respectively.
- *Loss functions*. As explained in Sections 3.1 and A.2, the parameters of the CNP $\psi = \{\theta\}$ were trained by backpropagation from the predictive log-likelihood

$$\mathcal{L}_\psi(y_t \mid x_c, y_c; x_t) = \log q_\theta(y_t \mid x_c, y_c; x_t).$$

The parameters of the LNP $\psi = \{\theta, \phi\}$ were trained by backpropagation from a loss function consisting of the predictive log-likelihood and a KL regularization term,

$$\mathcal{L}_\psi(y_t \mid x_c, y_c; x_t) = \log q_\theta(y_t \mid x_c, y_c; x_t) + D_{\text{KL}}(q_\phi(z \mid x_d, y_d) \parallel q_\phi(z \mid x_c, y_c)).$$

Due to the extra regularization term, achieving convergence with LNPs is more challenging than with CNPs. In order to facilitate convergence, we tried training LNPs either with an exact KL divergence term or with an approximate one, and retained the best-performing model out of the two. We approximated the KL term

$$D_{\text{KL}}(q_\phi(z \mid x_d, y_d) \parallel q_\phi(z \mid x_c, y_c)) = \mathbb{E}_{q_\phi(z \mid x_d, y_d)} \left[\log \frac{q_\phi(z \mid x_d, y_d)}{q_\phi(z \mid x_c, y_c)} \right]$$

with Monte Carlo, taking a single sample from $q_\phi(z \mid x_d, y_d)$. We found that the FP-LNP achieved better performance when trained with an approximate KL regularization term, whereas the MG-LNP achieved better performance when trained with an exact one.

E.3.2 ADKF-IFT

We used the ADKF-IFT model with regression weights, trained and shared by the ADKF-IFT authors at https://figshare.com/articles/online_resource/adkf-ift-weights.zip/22070105 .

F The challenge of meta-generalization in sinusoids

In this experiment, we investigated the ability of meta-learning techniques to meta-generalize to 1D sinusoids that were slightly different from the ones seen during meta-training. We trained two models, a 6-layer fully-connected NN with MAML, and a fully-connected CNP with 3 encoder layers and 3 decoder layers (for implementation details, see Supplementary Section E). Sinusoids had functional form

$$y = A \sin(\rho(x - B)),$$

where A is the amplitude, ρ the frequency and B the shift. As meta-training set, we used 10^4 functions with $A \in [0.1, 5)$, $\rho = 1.0$ and $B \in [0, \pi)$. All parameters were sampled uniformly from these ranges. For each function, we sampled points uniformly in $x \in [-5, 5]$ as contexts and targets. The context and target sets' sizes ranged between 5 and 25 and was also sampled too. The two models were trained for 10 epochs on the meta-train set. For meta-testing we used two different sets: in the first one, we created 10^4 functions with the same parameters as the meta-train functions, and in the second we created 10^4 functions with frequency $\rho = 1.5$. We sampled 20 points from each meta-test function in the same x interval as contexts, and 1000 points as targets. We also evaluated the prediction error on the meta-train set in the same way. The models performed well on the meta-train set, as expected, and meta-generalized well to the test functions. However, they failed to meta-generalize to the test functions with the slightly different frequency F.1.

Table F.1: Mean squared error (MSE) of two meta-learning models on sinusoids. Values show mean and standard deviation.

	Train functions ($\rho = 1.0$)	Test functions ($\rho = 1.0$)	Test functions ($\rho = 1.5$)
MAML	0.39 (0.10)	0.394 (0.10)	3.53 (0.28)
CNP	0.01 (0.00)	0.005 (0.00)	6.69 (0.17)

G Random sampling of observations protects NPs from overfitting

Increasing the percentage of points sampled as contexts or targets leads to less unique function views and to more effective epochs. This causes overfitting, both in terms of R^2 (Figure G.1) and NLPD (see Figure 9).

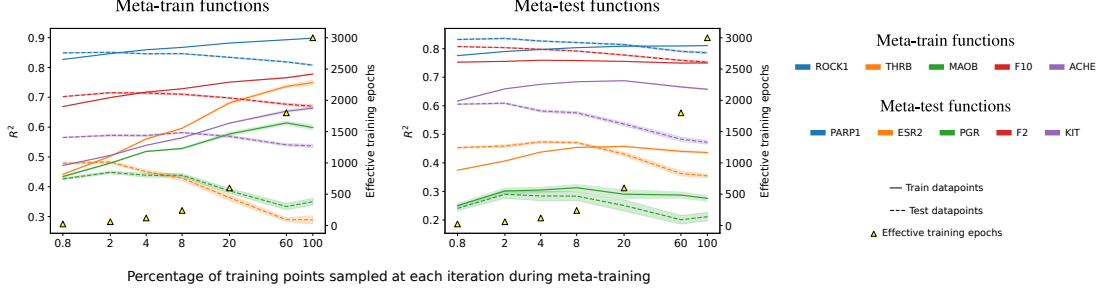


Fig. G.1: Increasing the percentage of points sampled as contexts or targets at each iteration leads to memorization of the labels from the train points of the train functions (left, solid lines) and a degradation of performance on the test points of the train functions (left, dashed line) and all points of the test functions (right, solid and dashed lines).

As the percentage of points sampled as contexts or targets increased, the calibration of the uncertainty estimates deteriorated substantially (Figure G.2).

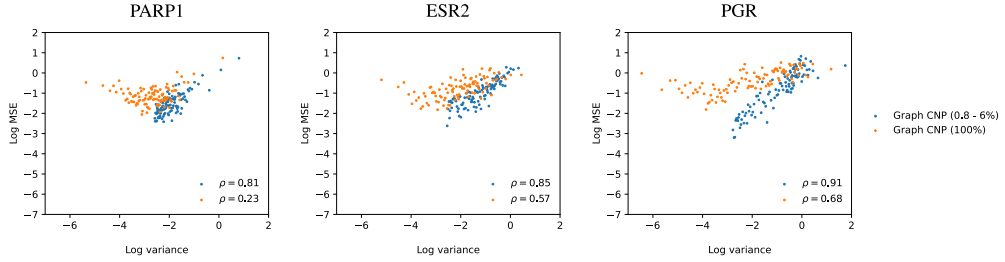


Fig. G.2: A MG-CNP meta-trained using a small fraction of the training points as contexts or targets (blue) maintains a good calibration, as shown by the high Pearson correlation between the predicted log variance and the prediction error. In contrast, a MG-CNP meta-trained using all training points as context or targets (orange) displays poor calibration.

H Bayesian optimization objectives

Our BO experiments (Section 4.5) used the objective functions druglike F2 and selective JAK2 from the DOCKSTRING optimization benchmark [10].

- *Druglike F2* is a comparatively easy task that requires docking well to a single target and satisfying orally-bioavailable druglike properties according to QED:

$$f_{F2}(\ell) = s(\ell, F2) + 10(1 - \text{QED}(\ell)),$$

where ℓ is a ligand molecule, and $s(\ell, p)$ is the docking score of ligand ℓ against protein p .

- *Selective JAK2* is a comparatively difficult task that requires docking well against JAK2 and not against LCK. This is a hard task since the docking scores of these two kinases are very highly correlated. This objective reflects a real interest in the drug discovery community to design selective kinase inhibitors.

$$f_{JAK2}(\ell) = s(\ell, JAK2) - \min(s(\ell, LCK) - 8.1, 0) + 10(1 - \text{QED}(\ell))$$

(Note that -8.1 is the median of LCK docking scores.)

References

- [1] Garnelo, M., Rosenbaum, D., Maddison, C., Ramalho, T., Saxton, D., Shananhan, M., Teh, Y.W., Rezende, D., Eslami, S.A.: Conditional neural processes. In: International Conference on Machine Learning, pp. 1704–1713 (2018). PMLR
- [2] Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D.J., Eslami, S.M.A., Teh, Y.W.: Neural Processes. In: ICML 2018 Workshop on Theoretical Foundations and Applications of Deep Generative Models (2018)
- [3] Mendez, D., Gaulton, A., Bento, A.P., Chambers, J., De Veij, M., Félix, E., Magariños, M.P., Mosquera, J.F., Mutowo, P., Nowotka, M., Gordillo-Marañón, M., Hunter, F., Junco, L., Mugumbate, G., Rodriguez-Lopez, M., Atkinson, F., Bosc, N., Radoux, C.J., Segura-Cabrera, A., Hersey, A., Leach, A.R.: ChEMBL: towards direct deposition of bioassay data. *Nucleic Acids Res.* **47**(D1), 930–940 (2019)
- [4] Finn, C., Abbeel, P., Levine, S.: Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In: International Conference on Machine Learning, pp. 1126–1135 (2017). PMLR
- [5] Raghu, A., Raghu, M., Bengio, S., Vinyals, O.: Rapid learning or feature reuse? towards understanding the effectiveness of MAML. In: International Conference on Learning Representations (2020)
- [6] Antoniou, A., Edwards, H., Storkey, A.: How to train your MAML. In: International Conference on Learning Representations (2019)
- [7] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
- [8] Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., Varoquaux, G.: API design for machine learning software: experiences from the scikit-learn project. In: ECML PKDD Workshop: Languages for Data Mining and Machine Learning, pp. 108–122 (2013)
- [9] Gardner, J., Pleiss, G., Weinberger, K.Q., Bindel, D., Wilson, A.G.: Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In: Advances in Neural Information Processing Systems, vol. 31 (2018)
- [10] García-Ortegón, M., Simm, G.N.C., Tripp, A.J., Hernández-Lobato, J.M., Bender, A., Bacallado, S.: DOCKSTRING: Easy Molecular Docking Yields Better Benchmarks for Ligand Design. *J. Chem. Inf. Model.* **62**(15), 3486–3502 (2022)