META-LEARNING INITIALIZATIONS FOR LOW-RESOURCE DRUG DISCOVERY

Anonymous authors
Paper under double-blind review

ABSTRACT

Building *in silico* models to predict chemical properties and activities is a crucial step in drug discovery. However, drug discovery projects are often characterized by limited labeled data, hindering the applications of deep learning in this setting. Meanwhile advances in meta-learning have enabled state-of-the-art performances in few-shot learning benchmarks, naturally prompting the question: *Can meta-learning improve deep learning performance in low-resource drug discovery projects?* In this work, we assess the efficiency of the Model-Agnostic Meta-Learning (MAML) algorithm – along with its variants FO-MAML and ANIL – at learning to predict chemical properties and activities. Using the ChEMBL20 dataset to emulate low-resource settings, our benchmark shows that meta-initializations perform comparably to or outperform multi-task pre-training baselines on 16 out of 20 in-distribution tasks and on all out-of-distribution tasks, providing an average improvement in AUPRC of 7.2% and 14.9% respectively. Finally, we observe that meta-initializations consistently result in the best performing models across fine-tuning sets with $k \in \{16, 32, 64, 128, 256\}$ instances.

1 INTRODUCTION

Drug discovery is a multi-parameter optimization process requiring efficient exploration of chemical space for compounds with desired properties. In a typical project, medicinal chemists propose structural changes to compounds in an effort to improve their therapeutic effects without compromising other properties. Validating these changes are costly – e.g. compounds need to be purchased or synthesized, assays need to be developed and validated – and thus *in silico* models are often used to prioritize experiments. Following the Merck Molecular Activity Challenge, there has been significant interest in applying deep learning to property prediction. More recently, by directly learning molecular features from chemical graphs, novel architectures in the graph neural networks family have demonstrated improved predictions in quantum chemistry and various property prediction benchmarks (Lusci et al., 2013; Duvenaud et al., 2015; Kearnes et al., 2016; Gilmer et al., 2017; Feinberg et al., 2018; Yang et al., 2019).

The successes of deep learning, however, hinge on an abundance of data: For instance, ImageNet (Deng et al., 2009) contains over 14M images and the English Wikipedia database commonly used to pre-train language models has over 2,500M words. On the contrary labeled scientific data in drug discovery projects often consists of many small, sparse, and heavily biased datasets, consequently limiting the applications of deep learning in this setting. Recent works approach this problem by using pre-training and multitask learning to leverage data from multiple sources (Ramsundar et al., 2015; Wenzel et al., 2019; Hu et al., 2019).

In parallel, the problem of learning in low-data domain has been tackled vehemently by the few-shot learning community. A prominent solution is the meta-learning paradigm, which aims to learn a learner that is efficient at adapting to new task (Thrun & Pratt, 1998; Vilalta & Drissi, 2002; Vanschoren, 2018). Matching Networks (Vinyals et al., 2016), a member of this family, have been previously applied to property prediction in one-shot learning settings by Altae-Tran et al. (2017). A related approach is the Model-Agnostic Meta-Learning (MAML) algorithm (Finn et al., 2017), which has particularly been successful at producing state-of-the-arts results on few-shots classification, regression, and reinforcement learning benchmarks, resulting in numerous follow-ups that expand on this elegant framework. In this work we evaluate the MAML algorithm, its first-order approximation (FO-MAML), and the Almost-No-Inner-Loop (ANIL) variant (Raghu et al., 2020) for learning to predict molecular properties and activities from chemical graphs in low-data domains. Specifically we aim to answer the following questions:

1. Does meta-initializations offer improvements over multitask pre-training in this setting?
2. How little data can meta-initializations learn efficiently from?

Using ChEMBL20 (Bento et al., 2014), performances of meta-initializations on in- and out-of-distribution tasks are benchmarked with multitask pre-training baselines, showing that they perform favorably across fine-tuning set sizes of $k \in \{16, 32, 64, 128, 256\}$ instances.

2 Background

MAML, FO-MAML, and ANIL MAML's approach is to directly optimize for a set of initial parameters that is efficient at learning from new data. The algorithm consists of an outer loop that learns an initialization θ_0, and an inner loop that adapts to new tasks starting from θ_0. In this setting we have a set of tasks $\{T_1, T_2, ..., T_K\}$ – denoted as $\mathcal{T}^{tr}$ – that is available to obtain θ_0, from which we would like to learn the set of tasks $\mathcal{T}^{test}$. Following the nomenclature in Finn et al. (2017), we call the process of obtaining the initialization θ_0 *meta-training*, and the process of adapting to $\mathcal{T}^{test}$ *meta-testing*. More formally, we define a task T_j with K instances as $T = \{(x_i, y_i) \,|\, i \in \{1, ..., K\}\}$ which is divided in to a training set $D^{tr}_{T_j}$ and a test set $D^{test}_{T_j}$, also referred to as the support and query set, respectively. The inner loop adaptation to T_j for a function f parameterized by θ using N gradient descent updates is expressed as

$$\theta^j_n = \theta^j_{n-1} - \alpha \nabla_\theta \mathcal{L}_{D^{tr}_{T_j}}(f_{\theta^j_{n-1}})$$

where θ^j_n are the network's weights after n steps toward task T_j, α is the inner loop learning rate, and $\mathcal{L}_{D^{tr}_{T_j}}$ is the loss on the training set of task T_j.The loss is calculated using the function f after $n-1$ updates. The inner loop repeated for a batch of B tasks sampled from $\mathcal{T}^{tr}$.

For the outer loop, the meta-loss is defined as the sum of task-specific losses after inner loop updates:

$$\mathcal{L}_{meta}(\theta_0) = \sum_{j=1}^{B} \mathcal{L}_{D^{test}_{T_j}}(f_{\theta^j_N})$$

The task-specific loss $\mathcal{L}_{D^{test}_{T_j}}$ is calculated on the test set of task T_j. We then minimize the meta-loss using stochastic gradient descent to optimize the initialization θ_0, with updates expressed by

$$\theta_0 \leftarrow \theta_0 - \eta \nabla_\theta \mathcal{L}_{meta}(\theta_0)$$

where η is the outer loop learning rate. Intuitively, the meta-loss $\mathcal{L}_{meta}(\theta_0)$ measures how well θ_0 adapts to new tasks, and minimizing this loss enables the algorithm to learn good initial parameters.

Updating θ_0 is computationally expensive since it requires the use of second-order derivates to compute $\nabla_\theta \mathcal{L}_{meta}(\theta_0)$. FO-MAML sidesteps this problem by omitting the second-order terms, effectively ignoring the inner loop gradients. On the other hand, Raghu et al. (2020) proposes the ANIL algorithm, which reduces the number of second-order gradients required by limiting inner loop adaptation to only the output layer of the network. ANIL and FO-MAML have both demonstrated significant speedup over MAML.

Graph Neural Networks The graph neural networks framework enables representation learning on graph structured data by learning node-level representations which are aggregated to form graph-level representations. Throughout our experiments, we use a variant of the Gated Graph Neural Network (GGNN) architecture (Li et al., 2017), a member of the message passing neural network (MPNN) family (Gilmer et al., 2017). Similar to other MPNNs, the GGNN architecture operates in two phases: a message passing phase and a readout phase. For an undirected graph $\mathcal{G}$ with V nodes

Table 1: Summary of dataset split

	A	T	P	B	F
$\mathcal{T}^{tr}$	0	0	0	126	737
$\mathcal{T}^{val}$	0	0	0	10	10
$\mathcal{T}^{test}$	1	1	1	10	10

where each node has F features, the message passing phase updates the hidden representation of node v at layer t according to

$$m_v^{t+1} = A_{e_{vv}} h_v^t + \sum_{w \in N(v)} A_{e_{vw}} h_w^t$$

$$h_v^{t+1} = \text{GRU}(h_v^t, m_v^{t+1})$$

where $A_{e_{vw}} \in \mathbb{R}^{F \times F}$ is an edge-specific learnable weight matrix, $N(v)$ denotes neighbors of v, GRU is the Gated Recurrent Unit (Cho et al., 2014), and $m_v \in \mathbb{R}^F$ is a message used to update the hidden representation of node v denoted by $h_v \in \mathbb{R}^F$. The message m_v is often interpreted as aggregating information across central and neighboring nodes. A deviation from Li et al. (2017) comes in our choice to remove weight sharing between GRUs in different layers. Following T updates, the readout phase pools node representations according to

$$\hat{y} = \text{MLP}\left(\sum_{v \in G} h_v^T \right)$$

to calculate the neural network output $\hat{y}$. Using sum as the readout operation is the second deviation from Li et al. (2017), and has been shown to have maximal expressive power over mean and max aggregators (Xu et al., 2018).

3 Experimental Settings

ChEMBL20 Dataset We evaluate the effectiveness of meta-initializations for low-resource tasks using a subset of ChEMBL20. More specifically, the dataset processed by Mayr et al. (2018) is filtered so that only tasks with at least 128 instances remain. The resulting dataset contains 902 binary classification tasks from 5 distinct task types denoted in the *assay_type* field: ADME (A), Toxicity (T), Physicochemical (P), Binding (B), and Functional (F).

The dataset is further divided into $\mathcal{T}^{tr}$, $\mathcal{T}^{val}$, and $\mathcal{T}^{test}$. To construct $\mathcal{T}^{val}$, we randomly select 10 B and F tasks. We randomly select another set of 10 B and F tasks and all A, T, and P tasks to construct $\mathcal{T}^{test}$. The rest of B and F tasks are included in $\mathcal{T}^{tr}$ for meta-training. A summary of task types in each set is shown in Table 3. From $\mathcal{T}^{test}$, we use B and T tasks to assess *in-distribution* performance, and A, T, and P tasks for *out-of-distribution* performance. For the baselines, we combine $\mathcal{T}^{tr}$ and $\mathcal{T}^{val}$, which is randomly split into $D^{tr}_{baseline}$ for training and $D^{val}_{baseline}$ for early stopping. This setup gives the baselines access to more tasks in during training.

Each molecule is represented as an undirected graph where nodes and edges are atoms and bonds. We use 75 atomic features for each node, as provided by DeepChem (Ramsundar et al., 2019).

Baselines We include the Finetune-All, Finetune-Top, and k-NN baselines as proposed by Triantafillou et al. (2019). To ensure the baselines are competitive, we perform hyperparameter tuning using the Tree-of-Parzen Estimator implementation of Hyperopt (Bergstra et al., 2015) to optimize performance on $D^{val}_{baseline}$ (see Appendix A for details). The resulting architecture has 7 GGNN layers, 1 fully-connected layer with 1024 units, and Dropout applied with a probability of 0.2 at every layer except for the output layer. We use the Adam optimizer (Kingma & Ba, 2017) with a learning rate of $10^{-3.75}$, batch size of 512, and patience of 20 epochs for early stopping during pre-training.

The k-NN baseline uses the activations from the penultimate of the pre-trained neural network to perform classification from 3 nearest neighbors. The Finetune-Top baseline reinitializes and trains

Table 2: Performance on in-distribution tasks measured in AUPRC. The top and bottom halves of the table are tasks with type B and F, respectively. Standard deviation are obtained from repeating experiments with five random seeds. The best and second best values are in bold and regular text, respectively. Statistically significant difference from the next best is denoted by (*).

ChEMBL ID	k-NN	Finetune-All	Finetune-Top	FO-MAML	ANIL	MAML
2363236	0.317 ± 0.007	0.324 ± 0.026	0.324 ± 0.022	$\mathbf{0.335 \pm 0.021}$	0.323 ± 0.008	0.333 ± 0.010
1614469	0.458 ± 0.009	0.491 ± 0.017	0.508 ± 0.021	0.491 ± 0.018	0.446 ± 0.044	$\mathbf{0.512 \pm 0.029}$
2363146	0.564 ± 0.014	0.649 ± 0.032	$\mathbf{0.663 \pm 0.022}$	0.547 ± 0.012	0.506 ± 0.034	0.582 ± 0.041
2363366	0.483 ± 0.015	0.557 ± 0.024	0.535 ± 0.021	0.539 ± 0.027	0.571 ± 0.030	$\mathbf{0.587 \pm 0.042}$
2363553	0.741 ± 0.006	0.720 ± 0.008	$\mathbf{0.745 \pm 0.021}$	0.697 ± 0.014	0.685 ± 0.022	0.690 ± 0.011
1963818	0.628 ± 0.037	0.744 ± 0.024	0.615 ± 0.156	0.679 ± 0.032	0.692 ± 0.081	$\mathbf{0.755 \pm 0.059}$
1963945	0.815 ± 0.017	$\mathbf{0.835 \pm 0.023}$	0.817 ± 0.022	0.773 ± 0.032	0.756 ± 0.035	0.827 ± 0.021
1614423	0.481 ± 0.024	0.615 ± 0.041	0.650 ± 0.049	0.767 ± 0.020	0.728 ± 0.079	$\mathbf{0.833 \pm 0.040}^*$
2114825	0.682 ± 0.012	0.751 ± 0.035	0.758 ± 0.033	0.844 ± 0.032	0.761 ± 0.076	$\mathbf{0.894 \pm 0.007}^*$
1964116	0.685 ± 0.012	0.758 ± 0.043	0.762 ± 0.038	0.893 ± 0.019	0.903 ± 0.019	$\mathbf{0.912 \pm 0.011}$
2155446	0.464 ± 0.003	0.476 ± 0.012	0.477 ± 0.009	0.503 ± 0.026	0.482 ± 0.021	$\mathbf{0.519 \pm 0.011}$
1909204	0.544 ± 0.023	0.592 ± 0.024	$\mathbf{0.602 \pm 0.034}$	0.589 ± 0.055	0.548 ± 0.026	0.599 ± 0.030
1909213	0.687 ± 0.006	0.740 ± 0.014	$\mathbf{0.755 \pm 0.010}$	0.692 ± 0.024	0.696 ± 0.027	0.725 ± 0.014
3111197	0.631 ± 0.028	0.653 ± 0.063	0.673 ± 0.084	0.644 ± 0.037	0.738 ± 0.030	$\mathbf{0.756 \pm 0.042}$
3215171	0.470 ± 0.029	0.557 ± 0.018	0.530 ± 0.052	0.733 ± 0.022	0.710 ± 0.050	$\mathbf{0.780 \pm 0.020}^*$
3215034	0.401 ± 0.063	0.458 ± 0.122	0.449 ± 0.196	$\mathbf{0.831 \pm 0.053}$	0.679 ± 0.020	0.811 ± 0.051
3215092	0.670 ± 0.021	0.780 ± 0.019	0.761 ± 0.025	0.875 ± 0.030	0.835 ± 0.022	$\mathbf{0.901 \pm 0.017}$
1738253	0.691 ± 0.028	0.844 ± 0.014	0.853 ± 0.023	0.859 ± 0.043	0.753 ± 0.107	$\mathbf{0.902 \pm 0.006}$
1909103	0.905 ± 0.010	$\mathbf{0.939 \pm 0.005}$	0.930 ± 0.009	0.884 ± 0.025	0.726 ± 0.054	0.911 ± 0.018
1614549	0.736 ± 0.026	0.868 ± 0.029	0.865 ± 0.050	0.932 ± 0.021	0.857 ± 0.031	$\mathbf{0.954 \pm 0.012}$
Avg. Rank	5.4	3.3	3.4	3.2	4.1	**1.7**

Table 3: Performance on out-of-distribution tasks measured in AUPRC. Standard deviations are obtained from repeating with five random seeds. Notations are the same as Table 2.

ChEMBL ID	k-NN	Finetune-All	Finetune-Top	FO-MAML	ANIL	MAML
1804798	0.319 ± 0.015	0.335 ± 0.023	0.329 ± 0.010	0.367 ± 0.015	0.361 ± 0.029	$\mathbf{0.369 \pm 0.021}$
2095143	0.291 ± 0.015	0.167 ± 0.042	0.302 ± 0.076	$\mathbf{0.579 \pm 0.024}$	0.561 ± 0.035	0.518 ± 0.044
918058	0.306 ± 0.048	0.538 ± 0.065	0.556 ± 0.050	0.494 ± 0.087	0.447 ± 0.184	$\mathbf{0.747 \pm 0.076}^*$
Avg. Rank	5.7	4.3	3.7	2.3	3.3	**1.7**

a new output layer while the Finetune-All baseline updates weights at every layer in the neural network. In our experiments each baseline is repeated with five random seeds.

Meta-Learning The same GGNN architecture as the baselines is used for all three meta-learning algorithms. Other hyperparameters are hand-tuned for performance on $\mathcal{T}^{val}$. For MAML and ANIL we use an inner loop learning rate of 0.05, 2 inner gradient steps, and inner batch size of 32, while the outer loop has a learning rate of 0.003 and a batch size of 32. FO-MAML uses an outer loop learning rate of 0.0015. We use the Learn2Learn (Arnold et al., 2019) and PyTorch (Paszke et al., 2019) libraries for our implementation and collect performances from five random seeds.

Evaluation For task T_j in $\mathcal{T}^{test}$, we fine-tune each method on k randomly selected instances from $D^{tr}_{T_j}$ using the Adam optimizer with learning rate of 10^{-4} and batch size of $b = min(64, k)$. We use $D^{val}_{T_j}$ for early stopping with patience of 10 epochs and report the final performances on $D^{test}_{T_j}$.

4 Results & Discussions

Performances on $\mathcal{T}^{test}$ The performance of each method on 23 test tasks is reported in Table 2 and 3. Since random splits have been shown to be overly optimistic in scientific applications (Kearnes et al., 2017; Wu et al., 2018), we emphasize relative ranking over absolute performance throughout our benchmark. We observe that meta-initializations generally exhibit similar or better performances over baselines despite having been trained on fewer tasks. For in-distribution tasks, MAML performs comparably to or outperforms other methods on 16 out of 20 tasks, 3 of which shows

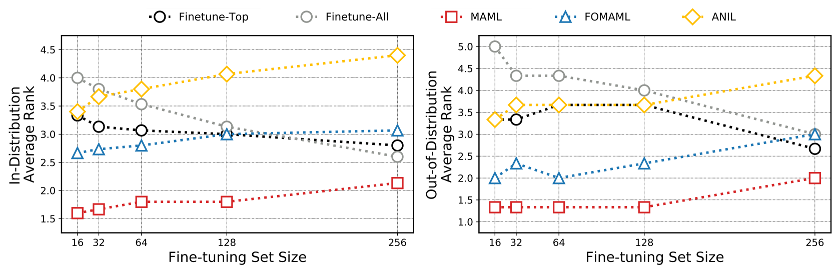

Figure 1: Average ranks of each method performance after fine-tuning with $k \in \{16, 32, 64, 128, 256\}$ instances for in- (left) and out-of-distribution tasks (right). Rankings are based on mean AUPRC measured from five random seeds. MAML and ANIL are consistently ranked as the best method across all k for in-distribution and out-of-distribution tasks, respectively.

significant improvement over the next best method, making it the top performer with an average rank of 1.7. ANIL and FO-MAML, while benefitting from a shorter training time (Appendix B), rank 3.2 and 4.1 on average, respectively. Similar to observations by Triantafillou et al. (2019), Finetune-All and Finetune-Top baselines prove to be strong competitors, both ranking above ANIL in our benchmark. Given their significantly shorter training time, we suspect both baselines to remain crucial in compute-limited settings. In out-of-distribution settings, meta-initializations outperform baselines on all 3 tasks. Again, MAML is ranked as the best method, followed by FO-MAML and ANIL. Overall, compared to the best baselines, meta-initializations learned by MAML provide an average increase in AUPRC of 7.2% for in-distribution tasks and 14.9% out-of-distributions tasks.

Effect of Fine-tuning Set Size From $\mathcal{T}^{test}$, we select all tasks with at least 256 instances in $D^{tr}_{T_j}$, resulting in 18 tasks available for evaluation (as opposed to 9 when a threshold of 512 instances is used). The average ranking of each method after fine-tuning on $k \in \{16, 32, 64, 128, 256\}$ instances is reported in Figure 1 (see Appendix C for performance on each task). As the best performing baselines from the previous experiment, Finetune-Top and Finetune-All are selected for comparison. We observe that the baselines benefit greatly from having more data, with Finetune-All rising from fifth to second in in-distribution tasks and Finetune-Top rising from fourth to second in out-of-distribution tasks. Nonetheless, MAML remains the best method, consistently ranked first across fine-tuning set sizes for both sets of tasks.

5 Conclusion & Future Directions

In this work, we explore meta-learning as a tool for learning to predict chemical properties and activities in low-resource settings. Emulating this setting using the ChEMBL20 dataset, we demonstrate that GGNN's initializations learned by MAML perform comparably to or outperform multitask pre-training baselines on 16 out of 20 in-distribution tasks and on all 3 out-of-distribution tasks. Improved performances of meta-initializations are further shown to remain consistent across fine-tuning sets of size $k \in \{16, 32, 64, 128, 256\}$.

While the ChEMBL20 dataset enables differentiating between in- and out-of-distribution tasks, we recognize that its chemical space is biased towards compounds that have been reviewed and selected for publications. Moreover, our analysis does not include initializations obtained using self- and unsupervised approaches such as those described in Veličković et al. (2018), Hu et al. (2019), and Sun et al. (2020). We leave experiments with additional datasets and methods to future work. Overall, we believe our contributions open opportunities in applying deep learning to ongoing drug discovery projects where limited data is available.

6 Acknowledgment

We would like to thank Jiajie Zhang, Robert Woodruff, and Darren Green for helpful discussions during the preparation of this manuscript.

REFERENCES

Han Altae-Tran, Bharath Ramsundar, Aneesh S. Pappu, and Vijay Pande. Low Data Drug Discovery with One-Shot Learning. *ACS Central Science*, 3(4):283–293, April 2017. ISSN 2374-7943. doi: 10.1021/acscentsci.6b00367.

Sebastien M. R. Arnold, Praateek Mahajan, Debajyoti Datta, and Bunner Ian. learn2learn, September 2019. URL `https://github.com/learnables/learn2learn`.

A. Patrícia Bento, Anna Gaulton, Anne Hersey, Louisa J. Bellis, Jon Chambers, Mark Davies, Felix A. Krüger, Yvonne Light, Lora Mak, Shaun McGlinchey, Michal Nowotka, George Papadatos, Rita Santos, and John P. Overington. The ChEMBL bioactivity database: an update. *Nucleic Acids Research*, 42(D1):D1083–D1090, January 2014. ISSN 0305-1048. doi: 10.1093/nar/gkt1031.

James Bergstra, Brent Komer, Chris Eliasmith, Dan Yamins, and David D. Cox. Hyperopt: a Python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, 8(1):014008, July 2015. ISSN 1749-4699. doi: 10.1088/1749-4699/8/1/014008.

Kyunghyun Cho, B. van Merrienboer, Caglar Gulcehre, F. Bougares, H. Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, 2014.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. Convolutional Networks on Graphs for Learning Molecular Fingerprints. *arXiv:1509.09292 [cs, stat]*, November 2015. arXiv: 1509.09292.

Evan N. Feinberg, Debnil Sur, Zhenqin Wu, Brooke E. Husic, Huanghao Mai, Yang Li, Saisai Sun, Jianyi Yang, Bharath Ramsundar, and Vijay S. Pande. PotentialNet for Molecular Property Prediction. *ACS Central Science*, 4(11):1520–1530, November 2018. ISSN 2374-7943. doi: 10.1021/acscentsci.8b00507.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135, International Convention Centre, Sydney, Australia, August 2017. PMLR.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for Quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 1263–1272, Sydney, NSW, Australia, August 2017. JMLR.org.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Pre-training Graph Neural Networks. *arXiv:1905.12265 [cs, stat]*, May 2019. arXiv: 1905.12265.

Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of Computer-Aided Molecular Design*, 30(8): 595–608, August 2016. ISSN 1573-4951. doi: 10.1007/s10822-016-9938-8.

Steven Kearnes, Brian Goldman, and Vijay Pande. Modeling Industrial ADMET Data with Multitask Networks. *arXiv:1606.08793 [stat]*, January 2017. arXiv: 1606.08793.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, January 2017. arXiv: 1412.6980.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated Graph Sequence Neural Networks. *arXiv:1511.05493 [cs, stat]*, September 2017. arXiv: 1511.05493.

Alessandro Lusci, Gianluca Pollastri, and Pierre Baldi. Deep Architectures and Deep Learning in Chemoinformatics: The Prediction of Aqueous Solubility for Drug-Like Molecules. *Journal of Chemical Information and Modeling*, 53(7):1563–1575, July 2013. ISSN 1549-9596. doi: 10.1021/ci400187y.

Andreas Mayr, Günter Klambauer, Thomas Unterthiner, Marvin Steijaert, Jörg K. Wegner, Hugo Ceulemans, Djork-Arné Clevert, and Sepp Hochreiter. Large-scale comparison of machine learning methods for drug target prediction on ChEMBL. *Chemical Science*, 9(24):5441–5451, June 2018. ISSN 2041-6539. doi: 10.1039/C8SC00148K.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid Learning or Feature Reuse? Towards Understanding the Effectiveness of MAML. In *International Conference on Learning Representations*, 2020.

Bharath Ramsundar, Steven Kearnes, Patrick Riley, Dale Webster, David Konerding, and Vijay Pande. Massively Multitask Networks for Drug Discovery. *arXiv:1502.02072 [cs, stat]*, February 2015. arXiv: 1502.02072.

Bharath Ramsundar, Peter Eastman, Patrick Walters, Vijay Pande, Karl Leswing, and Zhenqin Wu. *Deep Learning for the Life Sciences*. O'Reilly Media, 2019.

Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. *arXiv:1908.01000 [cs, stat]*, January 2020. arXiv: 1908.01000.

Sebastian Thrun and Lorien Pratt (eds.). *Learning to Learn*. Springer US, 1998. ISBN 978-0-7923-8047-4. doi: 10.1007/978-1-4615-5529-2.

Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. Meta-Dataset: A Dataset of Datasets for Learning to Learn from Few Examples. *arXiv:1903.03096 [cs, stat]*, October 2019. arXiv: 1903.03096.

Joaquin Vanschoren. Meta-Learning: A Survey. *arXiv:1810.03548 [cs, stat]*, October 2018. arXiv: 1810.03548.

Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. Deep Graph Infomax. *arXiv:1809.10341 [cs, math, stat]*, December 2018. arXiv: 1809.10341.

Ricardo Vilalta and Youssef Drissi. A Perspective View and Survey of Meta-Learning. *Artificial Intelligence Review*, 18(2):77–95, June 2002. ISSN 1573-7462. doi: 10.1023/A:1019956318069.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. Matching Networks for One Shot Learning. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 29*, pp. 3630–3638. Curran Associates, Inc., 2016.

Jan Wenzel, Hans Matter, and Friedemann Schmidt. Predictive Multitask Deep Neural Network Models for ADME-Tox Properties: Learning from Large Data Sets. *Journal of Chemical Information and Modeling*, 59(3):1253–1268, March 2019. ISSN 1549-9596. doi: 10.1021/acs.jcim.8b00785.

Table 4: Wall clock time to train each method

	TIME (HOURS)	SPEEDUP
MAML	57.9 ± 0.8	$1\times$
ANIL	48.0 ± 0.6	$1.2\times$
FO-MAML	27.0 ± 0.9	$2.1\times$
PRE-TRAINING	1.4 ± 0.1	$41.4\times$

Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. MoleculeNet: a benchmark for molecular machine learning. *Chemical Science*, 9(2):513–530, January 2018. ISSN 2041-6539. doi: 10.1039/C7SC02664A.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? *arXiv:1810.00826 [cs, stat]*, October 2018. arXiv: 1810.00826.

Kevin Yang, Kyle Swanson, Wengong Jin, Connor Coley, Philipp Eiden, Hua Gao, Angel Guzman-Perez, Timothy Hopper, Brian Kelley, Miriam Mathea, Andrew Palmer, Volker Settels, Tommi Jaakkola, Klavs Jensen, and Regina Barzilay. Analyzing Learned Molecular Representations for Property Prediction. *Journal of Chemical Information and Modeling*, 59(8):3370–3388, August 2019. ISSN 1549-9596. doi: 10.1021/acs.jcim.9b00237.

A BASELINES HYPERPARAMETER TUNING

Using Hyperopt, we allow a maximum of 50 evaluations and provide the following search space:

– Number of GGNN layers: $\{3, 7, 9\}$
– Fully connected layer dimension: $\{1024, 2048\}$
– Batch size: $\{128, 256, 512\}$
– Learning rate: $10^{\{-4.0,-3.75,-3.5,-3.25\}}$

B TRAINING TIME

Training time was measured as the total time required to reach best performance on the $D_{baseline}^{val}$ for baselines and $\mathcal{T}^{val}$ for MAML, FO-MAML, and ANIL on 1 NVIDIA Tesla V100 GPU. We report the recorded times in Table 4. The mean and standard deviation are calculated by repeating the training process with five random seeds.

C EFFECT OF FINE-TUNING SET SIZES ON PERFORMANCES

We report the performances used to create Figure 1 below. Figure 2 and 3 show in-distribution tasks, while Figure 4 shows out-of-distribution tasks.

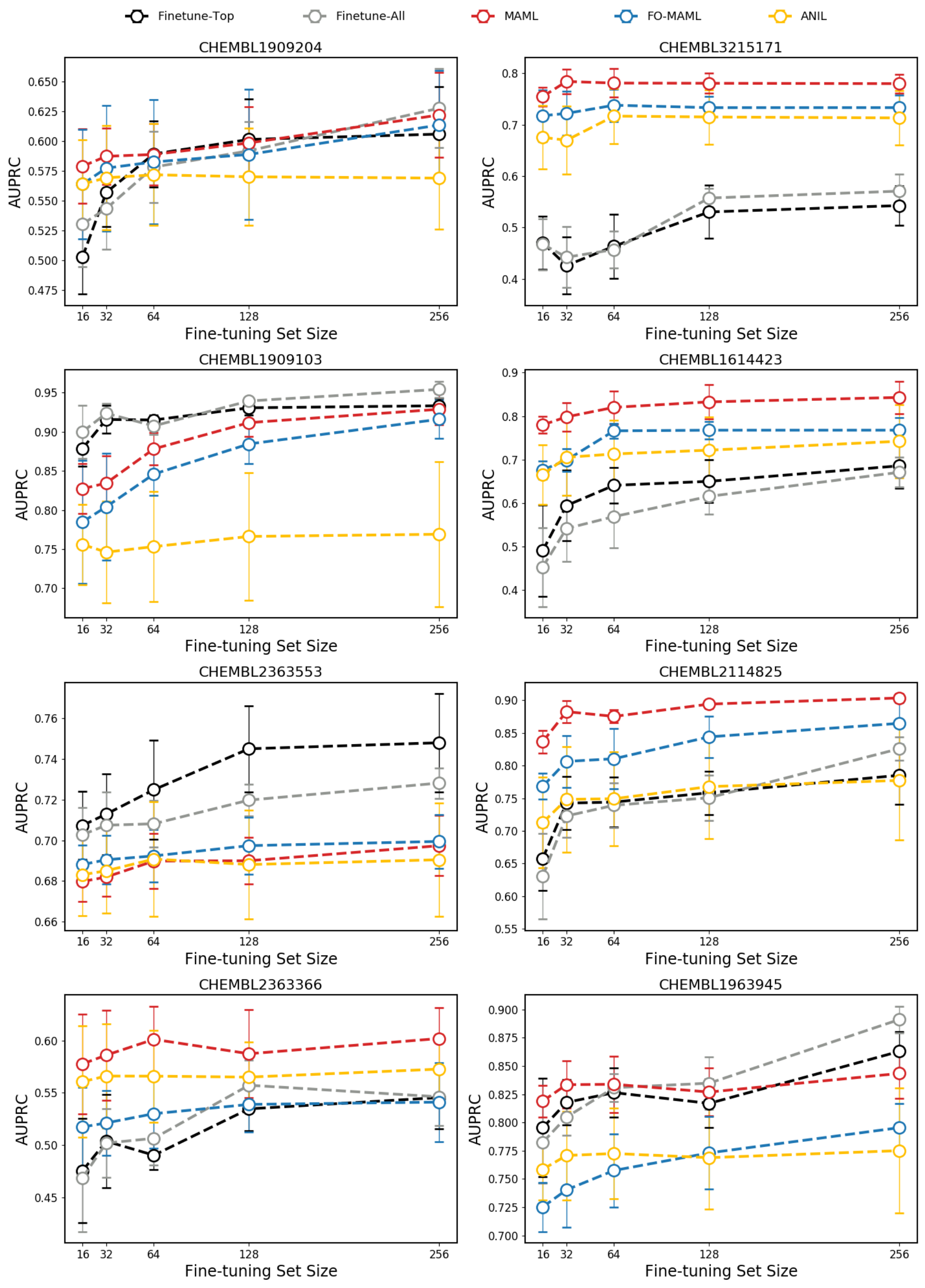

Figure 2: Performances on in-distribution tasks

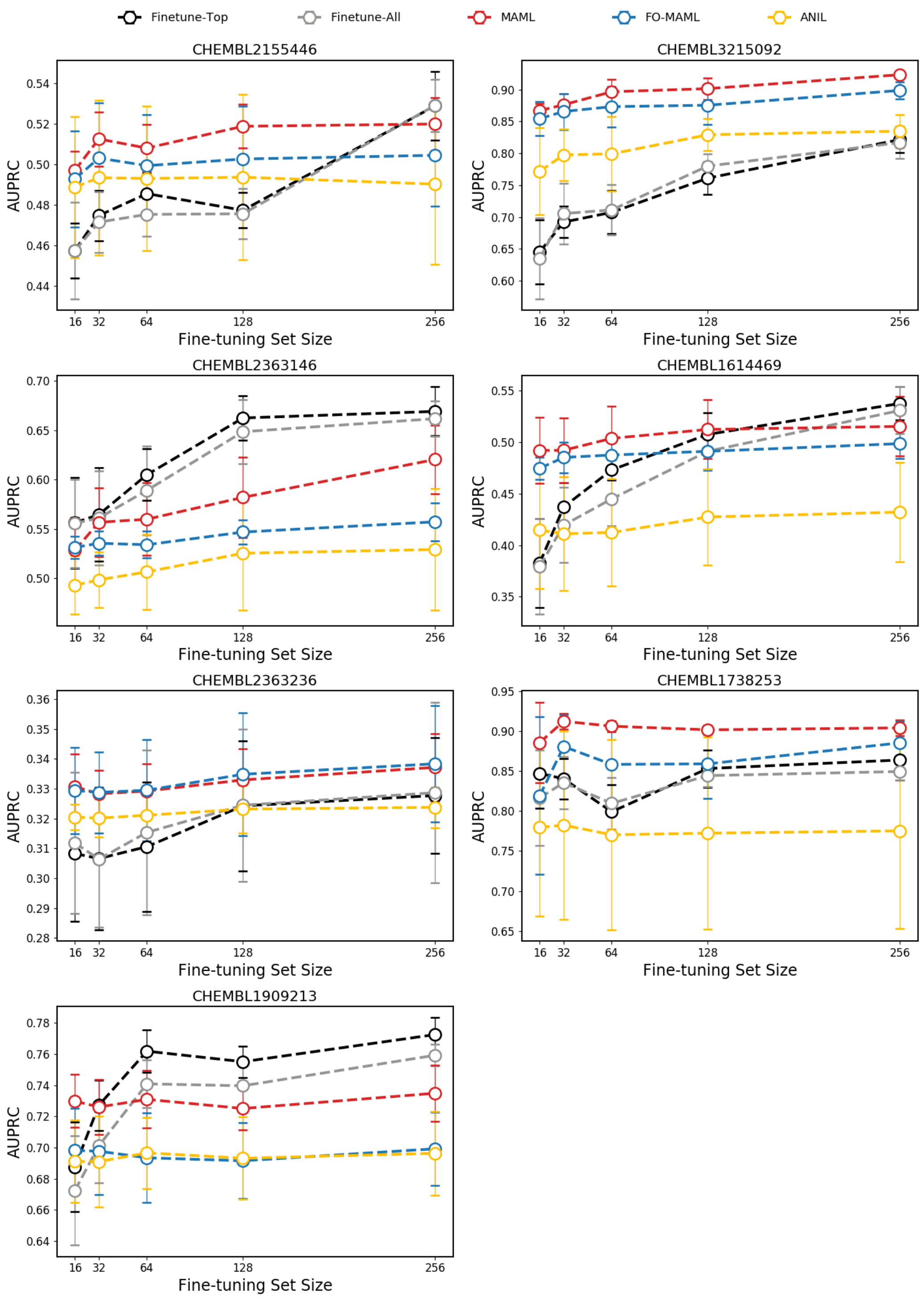

Figure 3: Performances on in-distribution tasks (continued)

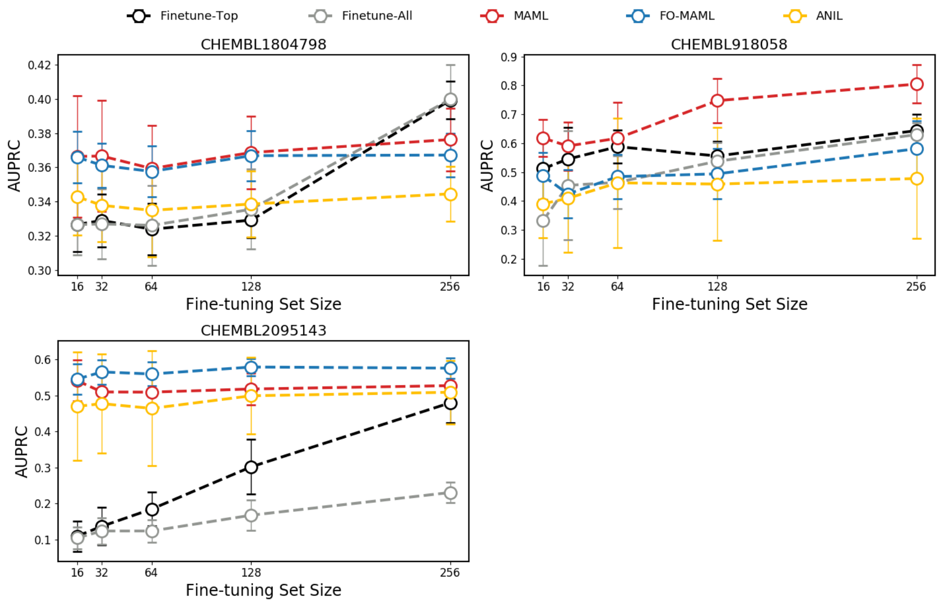

Figure 4: Performances on out-of-distribution tasks