Few-Shot Learning for Low Data Drug Discovery

Daniel Vella*,† and Jean-Paul Ebejer‡,¶

†*Department of Artificial Intelligence*

‡*Centre for Molecular Medicine and Biobanking*

¶*University of Malta, Msida, Malta*

E-mail: daniel.vella.12@um.edu.mt

Abstract

The discovery of new leads through ligand-based virtual screening in drug discovery is essentially a low-data problem, as data acquisition is both difficult and expensive to acquire. The application of conventional machine learning techniques to this problem domain is hindered by the requirement for large amounts of training data. In this work, we explore few-shot machine learning for LBVS, in which we build on the state-of-the-art, and introduce two new metric-based meta-learning techniques, Prototypical and Relation Networks, to this problem domain. We also explore the use of different embeddings and find that learned graph embeddings consistently perform better than extended-connectivity fingerprints. We conclude that the effectiveness of few-shot learning is highly dependent on the nature of the data. Few-shot learning models struggle to perform consistently on MUV and DUD-E data, in which the active compounds are structurally distinct. However, on Tox21 data, the few-shot models perform well, and we find that Prototypical Networks outperform the state of the art. Additionally, training these networks is substantially faster (up to 190%) and therefore take a fraction of the time to train for comparable, or better, results.

Introduction

We humans exhibit a remarkable ability to learn new concepts fast and efficiently. A child seeing a cat for the first time is able to discriminate future encounters with cats from other animals. This ability is in stark contrast with conventional supervised end-to-end machine learning, which is data hungry and requires a plethora of data points to develop an effective model. Meta-learning reframes the traditional machine learning problem, allowing machine learning models to learn utilising only a few examples. Humans have an innate capability to *learn how to learn*, and bridging this gap between human and machine learning is beneficial, particularly in domains where data availability or acquisition is difficult, such as the drug-discovery domain. The main goal in the drug-discovery process is the development of active compounds, that exhibit therapeutic effects against biological targets, which classifies the tested compound as a lead. The drug-discovery process comes with exorbitant costs and resource expenditure, which can exceed one billion dollars and take up to 15 years to complete.[1] Moreover, data is also expensive and difficult to acquire, as this requires testing of numerous compounds both *in-vitro* and *in-vivo*. Even upon identification of leads, attrition rates are high as the compound usually fails for other reasons such as poor absorption, distribution, metabolism, excretion, or toxicology (ADMET) characteristics.[2] It is difficult to predict such characteristics about the candidate molecule when only a small amount of related biological data is available. Therefore, the lead identification and optimisation step in drug discovery is essentially a low-data problem.[3] The requirement for the plethora of training data required to train a model hinders the applicability of machine learning to lead identification and optimisation in the drug-discovery process. In the past years, the computer vision saw successful applications and advancements for low-data machine learning.[4–7] Few-shot learning relieves the burden of collecting large-scale labelled data and makes the learning of rare cases possible.[8]

Building on this notion, we aim to explore few-shot learning for virtual screening to address the low-data problem in this domain. The ability for a machine learning model to

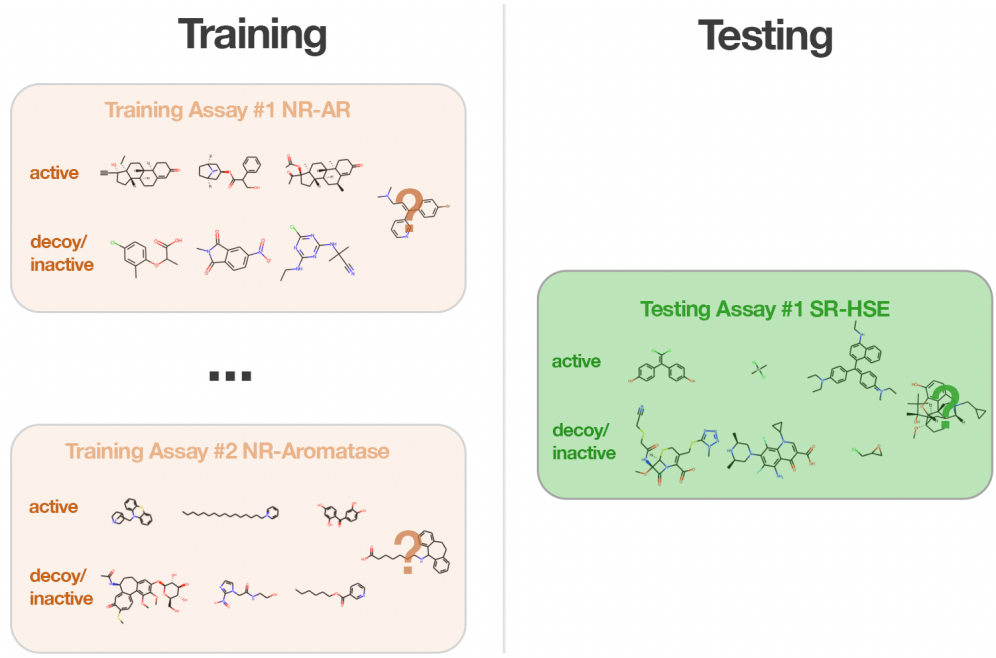

Figure 1: 2-way 3-shot few-shot classification. Training a meta-learner on a set of experimental assays, and generalising for an unseen assay in the Tox-21 dataset.

learn new concepts fast with just a few training examples is invaluable for virtual screening, where data on active compounds is scarce. Meta-learning aims to achieve generalising capabilities for environments that were previously unseen during training time. In meta-learning, such as few-shot classification, we train using a variety of training tasks and optimise for performance over a distribution of tasks, including unseen ones. Learning consists of a series of episodes, each consisting of an N-way K-shot classification task, effectively simulating the conditions at testing time. The *way* refers to the number of classes we have per task and the number of samples we have is the *shot* component. These samples make up the support set.[6] During test time, a small support set is sampled from new, previously unseen targets, and these few data points are used by the model to generalise for the activity of query molecules against this new target.[5] Figure 1 shows an example of a typical meta-learning scenario on the Tox21 dataset, where data from a set of training assays are used to train a model, which is in turn used to generalise for a new, unseen assay using a support set from this new assay. We highlight that few-shot learning in the domain under study is

in contrast to other domains such as computer vision, where the trained model recognises new classes. For example, given a few images of a lion as the support set, the model must generalise for new unseen images of a lion. In this domain, the challenge is to generalise to the behaviour of molecules in experimental assays which are related but not identical to the assays in the training collection. Given a few molecules from new experimental assays, can the model predict the activity of molecules in this new assay?

Related Work

Inspired by human learning,[9] few-shot learning makes use of data from similar tasks to compensate for the lack of data for the task at hand. Several successful research undertakings have exploited this paradigm,[4–7] mainly for the computer vision domain. Meta-learning refers to this ability to learn how to learn.[10,11] Being able to learn from only a few examples is important as certain domains do not have access to the plethora of data that we see in other domains such as computer vision. This inaccessibility could be due to privacy, safety, or ethical issues. For instance, data acquisition can be problematic in the drug-discovery domain due to possible toxicity, low activity or solubility in clinical candidates. Learning with low data leads to less expensive data gathering and reduced computational cost for learning.[8]

Altae-Tran et al.[3] introduce a deep-learning architecture for few-shot learning in drug discovery, building on past work in metric-based meta-learning,[5] in which they propose the iterative refinement long short-term memory (IterRefLSTM). IterRefLSTM builds on the Matching Networks[5] by introducing iterative refinement of embeddings using long-short term memory (LSTM) networks. We build on their work by applying other successful few-shot learning approaches to this problem domain. In this study, we explore the application of a number of few-shot learning architectures including, in chronological order, Siamese networks[4], Matching Networks[5], Prototypical Networks[6] and Relation Networks[7]. This

group of architectures fall under the umbrella of metric-based meta-learning. In our study, we embed molecule representations using graph convolution networks, and then use or learn a distance function over these embeddings. Effectively, metric-based learners seek to learn a relationship between the input embeddings in the task space. For the purposes of this study, few-shot learning refers to training with as little as one example per class, to a maximum of 10 examples per class. Training with only one example per class is referred to as one-shot learning[4,5].

Graph Neural Networks

Before processing the embeddings using few-shot machine learning architectures, the SMILES molecular representations need to be represented in computational space. Wu et al.[12] report that graph-based models outperforms conventional machine learning models on the majority of datasets, suggesting that a learned embedding is advantageous over other molecular representations. Thus, we opt for graph learned molecular representations to embed the input molecules. Graphs are natural representations of molecules, where nodes and edges represent atoms and bonds, respectively. When representing molecules, the set of vertices V intuitively refers to atoms within a molecule, while the set of edges E refers to the bonds that connects two atoms together (see Equation 1). Selected properties such as atomic number, atom type, charge, and valences, among others, can be encoded in the node feature vector.

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}) \tag{1}$$

Graph neural networks can be used to learn molecular representations by applying convolutional operators and pooling layers on the molecular graphs.[13] Embeddings learned through neural networks afford the construction of automated features, rather than fixed fingerprints. Graph neural networks are effective in transforming small molecules into real-valued vector representations, which has been found to be a productive way of processing

small molecules within deep neural networks.[14] Duvenaud et al.[15] report that using a differentiable method reduces collisions of substructures, and the fingerprint can be optimised to contain only relevant features. The fingerprint's interpretability is enhanced as this method captures activity and similarity of substructures.

Metric-based few-shot learning

The success of a few-shot learning model for metric-based meta-learning is dependent on the effectiveness of the kernel k_θ, which measures the similarity between data samples (see Equation 2) using a metric or distance function. The models discussed in this section, excluding the benchmark model, use the support and query embeddings generated from the graph neural network to learn the kernel function.

$$P_\theta(y|\mathbf{x}, S) = \sum_{(\mathbf{x}_i, y_i) \in S} k_\theta(\mathbf{x}, \mathbf{x}_i) y_i \tag{2}$$

Siamese Networks

Siamese networks[4,16] are composed of two identical networks, with shared weights and parameters, taking in a pair of data samples as inputs. As the neural networks share weights, the feature extraction is maintained to the same feature space for both inputs. These identical subnetworks are finally connected in a final layer that acts as a distance function for the two outputs.

Matching Networks

Building on Siamese Networks, but instead of learning a metric function over pairs of data, the classifier learns how to define a probability distribution of output labels from query/test examples using a support set S.[5] The classifier outputs a sum of attention weighted labels from the support set to predict the similarity between the test example and the samples from the support set. We use the same embedding function for the support and query sets

to compute the molecular embeddings. Subsequently, the cosine similarity between pairs of data points between the support and query sets is computed, which is then normalised by a softmax function. The attention mechanism a in $\hat{y} = \sum_{i=1}^{n} a(\hat{x}, x_i) y_i$ specifies how similar $\hat{x}$ is to each example x in S.

Prototypical Networks

Proposed by Snell et al.[6], these networks are similar to Matching Networks, but instead of comparing the query support to every support data point, a *prototype* is calculated, which takes all the support data points per class and creates an embedding by averaging over the embeddings. The euclidean distance between the query data points and the prototypes is calculated for classification. In a one-shot learning scenario, Prototypical Networks are equivalent to Matching Networks, however, the Euclidean distance is used instead of the cosine distance used in Matching Networks.

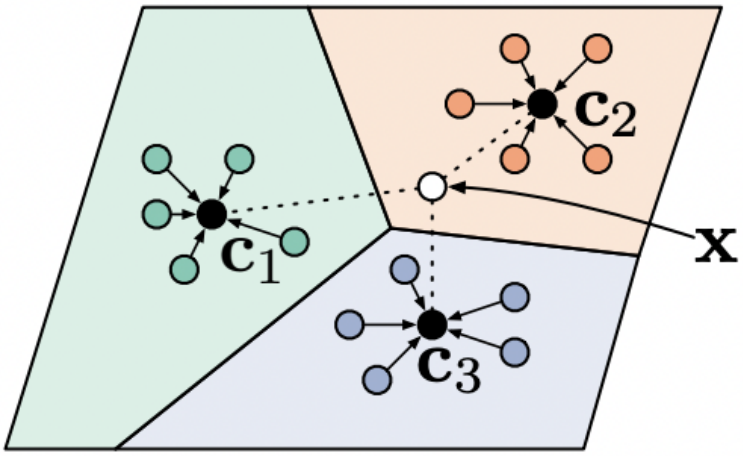

Figure 2: Few-shot learning in Prototypical Networks, where prototypes c_k are taken as the mean of embedded support examples for each class. Reproduced from Snell et al.[6].

Relation Networks

Sung et al.[7] present the Relation Network, a framework for few-shot learning, which could also be extended to zero-shot learning. The Relation Network learns a non-linear distance metric to compare support and query examples. As opposed to the aforementioned networks, this network uses a feed-forward neural network to learn a distance function in feature space. After embedding the support and query examples through an embedding function, each query example is concatenated with each of the feature maps. The resulting feature map concatenations are processed using a convolutional neural network to output a relation score vector, from which the class can be inferred.

Iterative Refinement LSTM

Altae-Tran et al.[3] propose the iterative refinement long-short term memory (IterRefLSTM) to further process the resulting embeddings in a few-shot machine learning pipeline. In IterRefLSTMs two embedding functions $f(\dot{|}S)$ and $g(\dot{|}S)$ are developed simultaneously. Therefore, the embedding of the query is built iteratively with that of the support set, using information from both sets to enhance both the support and query embeddings (see Figure 3).

Methodology

In this work, we implement two benchmark models, and four few-shot machine learning architectures. Molecules are represented using graph objects, which are processed using graph convolutional networks (GCNs). IterRefLSTMs are used to enrich the resulting embeddings in latent space. The following sections go into the implementations in more detail.

Datasets

In this work, we make use of the following three datasets;

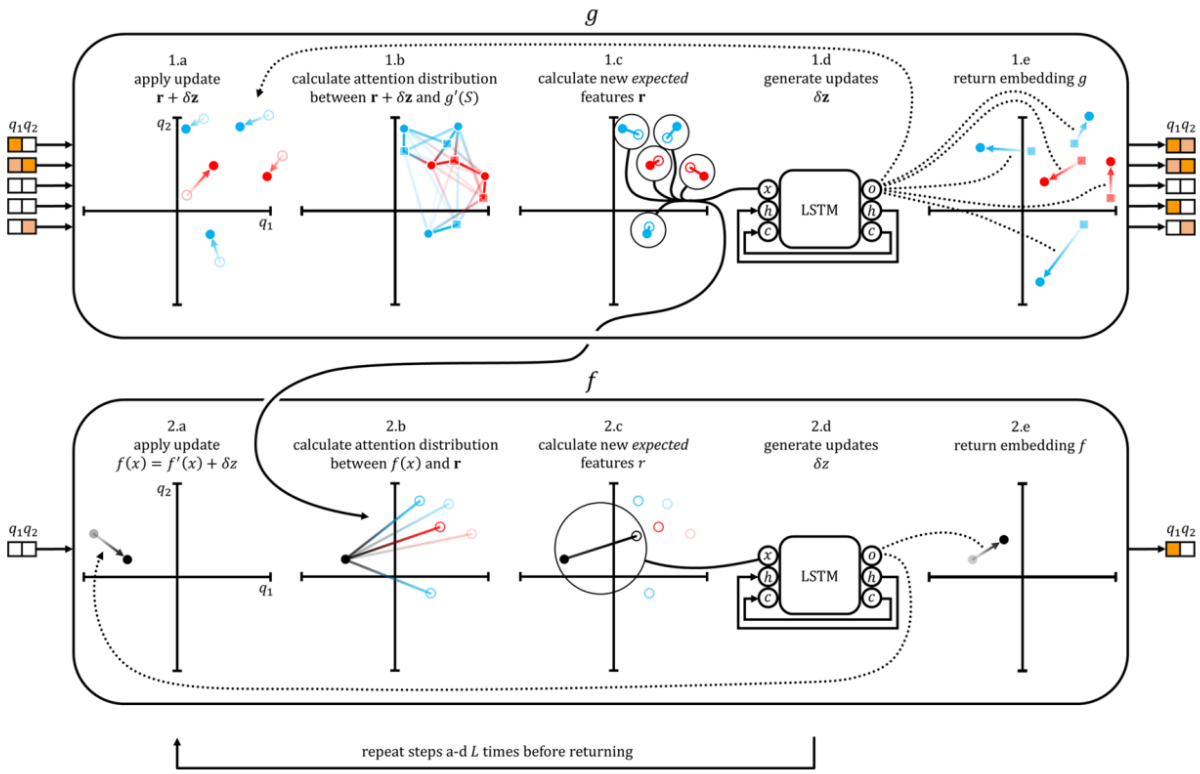

Figure 3: Iterative refinement of embeddings using an LSTM network. The red and blue points depict the active and inactive/decoy class, respectively. The squares depict the original embedding from g'. Reproduced from Altae-Tran et al.[3].

Tox21.[17] Mainly used for lead optimisation, containing toxicity data for 12 targets[18]. The dataset was obtained from the DeepChem AWS bucket[1] in CSV format. The NR-AR, NR-AR-LBD, NR-AhR, NR-Aromatase, NR-ER, NR-ER-LBD, NR-PPAR-gamma, SR-ARE, SR-ATAD5 targets are reserved for training, and the remaining SR-HSE, SR-MMP, SR-p53 targets for testing. *Maximum Unbiased Validation (MUV)*[19]. Based on PubChem BioAssays, used for validating virtual screening techniques against 17 different targets[19]. The dataset was obtained from the DeepChem AWS bucket[2] in CSV format. A total of 12 targets (MUV-466 - MUV-810) are reserved for training, while MUV-832, MUV-846, MUV-852, MUV-858, and MUV-859 are reserved for testing. *Directory of Useful Decoys (Enhanced)*

[1]Accessed from: deepchemdata.s3-us-west-1.amazonaws.com/datasets/tox21.csv.gz. Last Accessed: 08/11/2021

[2]Accessed from: deepchemdata.s3-us-west-1.amazonaws.com/datasets/muv.csv.gz. Last Accessed: 08/11/2021

(DUD-E).[20] Used for benchmarking virtual screening techniques by introducing a number of active compounds against specific targets. For each active, a number of *decoys* with similar physical properties, but different topologies, are made available. For this research study, we made use of the GPCR subset of the DUD-E dataset[20]. The data was obtained directly from the DUD-E website.[3] The AA2AR, DRD3, and ADRB1 are used for training. Two targets are reserved for testing, in which ADRB2 contains decoys that are auto-generated against a set of known active ligands, while for the CXCR4 target these are hand-picked.

Table 1: Number of actives and inactives/decoys across all targets in the datasets used. Figures in parentheses show the percentage of the total compounds in the dataset.

Dataset	Actives	Inactives/Decoys
Tox21	4,149 (7.04%)	54,746 (92.96%)
MUV	347 (0.20%)	175,990 (99.80%)
DUD-E (GPCR)	1,249 (1.45%)	84,856 (98.55%)

Molecular Representations

We first create a molecular graph from the SMILES string using RDKit, an open-source toolkit for cheminformatics. Standardisation of compounds according to a set of well-defined and consistent rules and conventions is of utmost importance to maintain uniformity and integrity across the data being used. Bento et al.[21] present an open source chemical structure curation pipeline based on RDKit for validating and standardising chemical structures, which follow FDA/IUPAC guidelines[22,23]. Their work is available in the ChEMBL Structure Pipeline package[21] and is used to standardise the molecules in our pipeline. We then create one-hot encoded features for the atoms in each molecule, namely, atom type, atomic number, atom degree, explicit valence, hybridisation, formal charge, number of radical electrons, and aromaticity. Self loops are added to every node in the generated graph, so aggregation functions during message passing consider the features of the node itself. The order of the atoms follows the canonical order of the atoms assigned through RDKit. We make use of

[3]Accessed from: http://dude.docking.org/subsets. Last Accessed: 08/11/2021

the DGL LifeSci[24] library to create the graph objects and subsequently process them using the DGL library.[25]

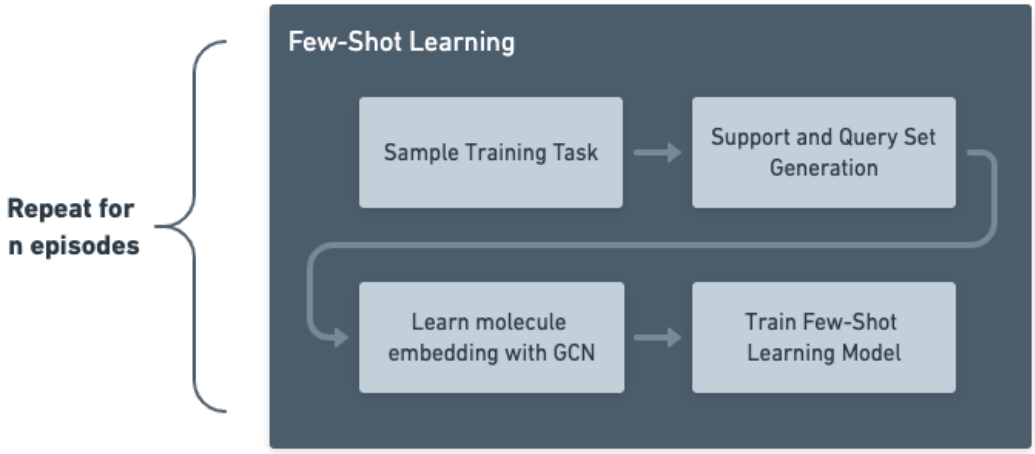

Figure 4: Episodic learning schematic

Machine Learning Models

Before processing the molecular graph, we first learn an embedding using graph neural networks. Four different architectures, including Siamese, Matching, Prototypical and Relation Networks, process the learned graph embeddings to train our meta-learner. IterRefLSTMs are utilised to refine the latent space embeddings.

Graph Convolutional Networks

Graph convolutional networks (GCNs) are used to learn embeddings for the support and query molecules in latent space. Figure 5 illustrates the GCN pipeline to learn a molecular embedding. In our study, we make use of the convolutional operator from Kipf and Welling[26] to process graphs and learn the molecular embeddings. The convolutional layer can be mathematically defined through Equation 3. h_j is the feature set of the node, N_i is the set of neighbouring nodes i, b is the learnable bias, and $c_j i$ is the product of the square

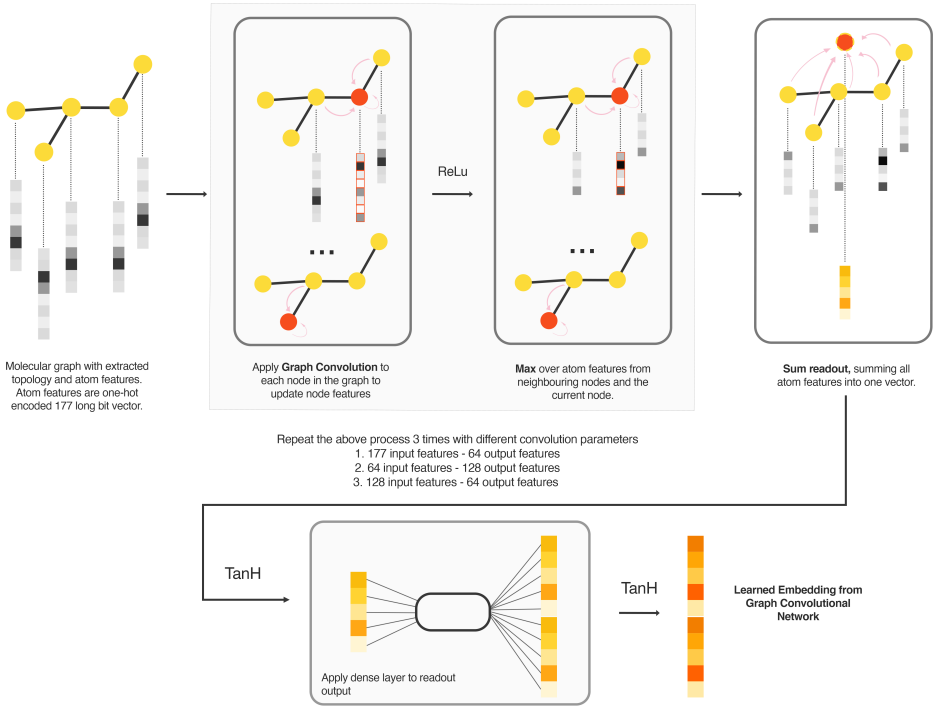

Figure 5: Learning an embedding through a Graph Convolutional Network (GCN). The molecule, represented as a graph object with nodes, edges, and atom features, is processed using graph convolutions. A max message-passing function over the current and neighbouring nodes follows each convolution layer. After this process, a sum readout aggregates all atom features into one vector. A TanH function activates this vector, and a dense linear layer processes the output vector. A non-linear TanH function activates this vector to yield the final learned molecular embedding.

root of node degrees. From a message-passing perspective, this can be summarised into the following steps for every node feature space u;

$$h_i^{(l+1)} = \sigma(b^{(l)} + \sum_{j \in \mathcal{N}(i)} \frac{1}{c_{ji}} h_j^{(l)} W^{(l)}) \tag{3}$$

1. Aggregating the neighbouring representations h_v, producing an intermediate representation $\hat{h}_u$.

2. Transforming $\hat{h}_u$ through a linear projection and a non-linearity function such that $h_u = f(W_u \hat{h}_u)$[26].

Three convolutional layers are present in our architecture, after which a maximum function aggregating the node features with the maximum value of the neighbours and the node itself is applied. We highlight that this is not a coarsening operation, as the number of nodes remain the same. Finally, we apply a global pooling layer (readout), in which we sum over the node features of the graph (see Equation 4). A linear transformation is applied to the output from the read-out layer, followed by a non-linear activation function, for which we use a hyperbolic tangent function (TanH), outputting the final molecule embedding.

$$r^{(i)} = \sum_{k=1}^{N_i} x_k^{(i)} \tag{4}$$

Benchmark

We make use of a Random Forest model with 100 decision trees and a Graph Convolutional Network (GCN) to build a baseline to benchmark the purpose-built few-shot learning models. For the random forest model, ECFP representations of the molecules of size 2048 bits are used for the classification task. Meanwhile, the same GCN architecture used for the few-shot learning models is used for the benchmark. The only addition to the architecture is a final linear layer that takes as input 128 features, which is the size of the embedding used for

the experiments to follow, and outputs a feature of size one, onto which we apply a non-linear function, in this case a Sigmoid function, to output the probabilities for a Boolean target $(0, 1)$. These two models are trained on a small support set, sampled from the targets assigned for testing. The remaining data for the designated target is used for testing.

Few-Shot Learning Models

Episodic learning is used to train a few-shot machine learning model. Vinyals et al.[5] suggest that conditions during training must match those during testing. Training consists of a sequence of learning problems where the model is supplied with a *support* set and a corresponding *query* set. The support set consists of a few molecules sampled from each class, in our case representing the active molecules and the inactives/decoys. We consider *N-way K-shot* classification tasks, where the support set contains N classes and K labelled molecules. N is always assigned a value of two as we are attempting to solve a binary classification problem, whereby the model tries to classify the query molecules as active or inactive in a specific experimental assay. We experimented with a varying number of molecules for the support sets, however the minimum limit was set to one compound per class, while the maximum was set to 10 compounds per class. The *2-way N-shot* formulation is what the model is presented with at test time. We sample a total of 128 query molecules for each episode, which is composed of a balanced combination of molecules from each class. If the active class for a specific target contains less than 64 molecules, the active molecules are over-sampled such that each query set contains 64 actives. We reproduce the work of Altae-Tran et al.[3] from scratch and also apply the IterRefLSTM to the embeddings from all other networks to effectively compare our contribution to past work. Additionally, we also provide implementations for the Prototypical Networks and Relation Networks. All the experiments are run on Google Colaboratory and all implementations are open-sourced on GitHub.[4]

[4]Accessed From: https://github.com/danielvlla/Few-Shot-Learning-for-Low-Data-Drug-Discovery

Results

Performance evaluation is carried out using Receiver Operator Characteristic AUC (ROC-AUC) and Precision-Recall Area Under the Curve (PR-AUC) metrics. The state-of-the-art[3] only reports ROC results, however, this metric alone does not fully encompass the nature of the performance of the machine learning models due to the hihgly imbalanced nature of the data at hand. In virtual screening, the detection of rare events (equivalent to our minority active class) holds significant importance, as active compounds against a specific target should be identified from the compound database. However, we do not disregard the importance of correct classification of the majority inactive/decoy class, as this is also important for filtering out thousands of screened compounds. As the active class is the minority class, PRC are used to evaluate how well the model can classify the active class. A high area under the PRC indicates high recall and high precision. The former is related to a low false negative rate, meaning active compounds incorrectly classified as inactive/decoys, while high precision is attributed by a low false positive rate, meaning compounds classified as active when in fact they are inactive/decoys. The ideal scenario for predicting the minority active class is thus one where we achieve high recall and high precision. As our data contains a lot of negative examples, there is a higher chance of these being predicted as false positives. On the other hand, we have much fewer active examples which could be predicted as false negatives. Given that the active class is in such a minority, even a small false positive rate could result in high numbers of false positives, due to the high number of the negative class examples. In this scenario, the precision will be low as we are predicting a lot of false positives when compared to true positives. We can also have a scenario of high recall with low precision. In this scenario, we have a high number of incorrect predictions as the model returns a lot of false positives, but it correctly predicts most of the active class as it has high recall. On the other hand, if we achieve high precision with low recall, most of the predictions are correct as we have a high number of true positives when compared to false positives.

The notation K+/K- used in the tables to follow signify the composition of the support set consisting of a number of actives and inactives/decoys respectively. Each model is evaluated 20 times per target, with a randomly sampled support set for each round. The tabulated results are the mean values from these 20 rounds, encompassing all the test targets, along with the standard deviation for each mean.

Tox21

Table 2: Mean ROC-AUC and PRC-AUC Scores with standard deviation for ML Models for the Tox21 Test Targets over 20 rounds of testing. Bold text illustrates the best obtained value. The first column shows the composition of the support set.

Tox21	**Metric**	**RF**	**Graph Conv**	**SiameseNet**	**MatchingNet**	**ProtoNet**
10+/10-	ROC	0.617 ± 0.060	0.620 ± 0.065	0.825 ± 0.043	0.824 ± 0.022	**0.826 ± 0.034**
	PRC	0.158 ± 0.102	0.150 ± 0.095	0.226 ± 0.107	0.367 ± 0.105	**0.384 ± 0.105**
5+/10-	ROC	0.602 ± 0.059	0.610 ± 0.062	0.828 ± 0.069	**0.824 ± 0.033**	0.823 ± 0.038
	PRC	0.148 ± 0.090	0.152 ± 0.094	0.190 ± 0.094	0.369 ± 0.110	**0.388 ± 0.111**
1+/10-	ROC	0.563 ± 0.068	0.558 ± 0.076	0.836 ± 0.138	0.822 ± 0.025	**0.826 ± 0.032**
	PRC	0.128 ± 0.084	0.126 ± 0.075	0.099 ± 0.093	0.301 ± 0.103	**0.384 ± 0.096**
1+/5-	ROC	0.534 ± 0.066	0.559 ± 0.090	0.807 ± 0.159	**0.820 ± 0.033**	**0.820 ± 0.033**
	PRC	0.112 ± 0.059	0.128 ± 0.080	0.106 ± 0.086	0.339 ± 0.115	**0.362 ± 0.106**
1+/1-	ROC	0.550 ± 0.061	0.548 ± 0.102	0.818 ± 0.075	0.819 ± 0.036	**0.820 ± 0.030**
	PRC	0.118 ± 0.068	0.123 ± 0.082	0.198 ± 0.102	0.352 ± 0.121	**0.373 ± 0.102**

When testing the machine learning models on the Tox21 dataset, the few-shot learning models outperform the baseline models. This performance is in line with that reported by Altae-Tran et al.[3]. In line with the established state-of-the-art, the MN with IterRefLSTM performs well and obtain the best ROC results in a number of experiments. The fact that the same implementation for the Matching Networks (MNs) obtained slightly better results (1-14% across the five support set experiments) than the state-of-the-art work, can be attributed to the set of atom descriptors used for the initial graph representations. Our few-shot learning architecture implementation is identical to their work, but different hyperparameters in the implementation which were not clear in the original work could also contribute to variations in results. Hence, we focus mainly on the performance of how our implementations performed

against each other. The results from the Prototypical Networks (PNs) overall outperform the results from the MNs. Meanwhile, MNs and PNs, overall outperform Relation Networks (RNs) in both ROC and PRC performance. Results for one shot-learning do not provide a clear-cut choice between our implementations for MNs and PNs with the IterRefLSTM. They both achieve comparable performance on Tox21 targets for one-shot learning. The performance of MNs for this scenario is consistent with the state-of-the-art work. Based on the underlying theories of both architectures, the results are consistent with our expectations. In a one-shot learning scenario, MNs and PNs are conceptually similar. The *prototypes* in PNs are a mean of all embeddings for each class in the support set. The euclidean distance between the *prototypes* and each embedding from the query set is calculated to predict the query's activity. As in one-shot learning we only have one example per class, the *prototypes* are equivalent to the embedding for each class, making this identical to MNs. The difference lies in the distance function used as for MNs we use the cosine distance, while for PNs, we make use of the euclidean distance, as proposed in the original literature which introduced these two techniques.

MUV

Each active in the MUV dataset is structurally distinct from the other, making each data sample maximally informative. Therefore, structural similarities cannot be exploited on unseen active molecules. The baseline benchmark tests consistently outperformed few-shot learning techniques. Altae-Tran et al.[3] report that the results obtained through the GCNs baseline also struggle in performance, however, from our tests and statistical analysis we find that this is not the case for all MUV targets. For most targets, there is no significant difference between the scores obtained through the RFs and GCNs baselines. RNs obtain the best ROC scores in one instance on the MUV-832 target when trained with a 1+/10- support set, obtaining a mean ROC-AUC score of 0.683 ± 0.010. However, this result is not consistent and the performance is only observed in this single instance. Other than this

single instance, our results are consistent with the conclusion from the state-of-the-art that baseline machine learning outperforms few-shot machine learning techniques on the MUV dataset.

Table 3: Mean ROC-AUC and PRC-AUC Scores with standard deviation for ML Models for MUV Test Targets over 20 rounds of testing. Bold text illustrates the best obtained value. The first column shows the composition of the support set.

MUV	Metric	RF	Graph Conv	SiameseNet	MatchingNet	ProtoNet
10+/10-	ROC	**0.728 ± 0.145**	0.713 ± 0.133	0.562 ± 0.046	0.628 ± 0.096	0.599 ± 0.085
	PRC	**0.066 ± 0.073**	0.009 ± 0.012	0.001 ± 0.000	0.007 ± 0.010	0.003 ± 0.002
5+/10-	ROC	**0.696 ± 0.132**	0.666 ± 0.115	0.550 ± 0.054	0.516 ± 0.085	0.576 ± 0.055
	PRC	**0.071 ± 0.076**	0.015 ± 0.022	0.001 ± 0.001	0.003 ± 0.002	0.003 ± 0.002
1+/10-	ROC	**0.599 ± 0.104**	0.585 ± 0.116	0.648 ± 0.158	0.492 ± 0.082	0.540 ± 0.053
	PRC	**0.021 ± 0.032**	0.006 ± 0.008	0.001 ± 0.002	0.002 ± 0.001	0.003 ± 0.002
1+/5-	ROC	**0.587 ± 0.106**	0.585 ± 0.126	0.613 ± 0.179	0.461 ± 0.046	0.494 ± 0.050
	PRC	**0.027 ± 0.040**	0.006 ± 0.008	0.001 ± 0.002	0.002 ± 0.001	0.002 ± 0.001
1+/1-	ROC	0.573 ± 0.103	**0.577 ± 0.147**	0.620 ± 0.138	0.507 ± 0.037	0.505 ± 0.030
	PRC	**0.022 ± 0.036**	0.006 ± 0.007	0.004 ± 0.011	0.002 ± 0.000	0.003 ± 0.001

GPCR subset of the DUD-E

For the ADRB2 target, the few-shot learning models achieve stellar performance based on ROC and PRC scores. The results are close to a perfect classifier, which raises concerns about the underlying data. Our hypothesis is that the underlying data contains an inherent bias, which is confirmed by further research on the matter. Some studies indicate that the DUD-E dataset has limited chemical space and bias from the decoy compound selection process.[27,28] Chen et al.[29] investigate this further to establish the effect these characteristics have on CNN models. The authors conclude that there is analogue bias within the set of actives within the targets (intra-target analogue bias), and also between the actives of different targets (inter-target analogue bias). They also provide evidence that there is also bias in decoy selection through the selection criteria for decoys. Results obtained from the DUD-E dataset are not conclusive. On the other hand, for the hand-picked decoys for the CXCR4 target, the RF model excels and outperforms the few-shot learning models. Seeing that the

GCN benchmark model also performed significantly better than few-shot learning models, this implies that there is a clear benefit of training on the same data from the target, as opposed to the few-shot learning models which are trained on other targets instead. Having such mixed results on two different targets within the same subset of the dataset does not give us a conclusive picture of whether few-shot learning is effective on this kind of data.

Table 4: Mean ROC-AUC and PRC-AUC Scores with standard deviation for ML Models for DUD-E GPCR Test Targets over 20 rounds of testing. Bold text illustrates the best obtained value. The first column shows the composition of the support set.

DUDE-GPCR	Metric	RF	Graph Conv	SiameseNet	MatchingNet	Proto
10+/10-	ROC	**0.982 ± 0.018**	0.940 ± 0.039	0.784 ± 0.215	0.900 ± 0.102	0.816 ±
	PRC	**0.872 ± 0.102**	0.504 ± 0.225	0.489 ± 0.475	0.535 ± 0.451	0.552 ±
5+/10-	ROC	**0.958 ± 0.023**	0.901 ± 0.058	0.761 ± 0.238	0.845 ± 0.153	0.843 ±
	PRC	**0.762 ± 0.119**	0.428 ± 0.247	0.495 ± 0.465	0.506 ± 0.477	0.559 ±
1+/10-	ROC	0.854 ± 0.071	0.788 ± 0.098	0.759 ± 0.247	**0.881 ± 0.119**	0.841 ±
	PRC	0.360 ± 0.136	0.230 ± 0.176	0.474 ± 0.445	0.521 ± 0.455	0.504 ±
1+/5-	ROC	**0.858 ± 0.084**	0.763 ± 0.087	0.759 ± 0.246	0.851 ± 0.155	0.793 ±
	PRC	0.378 ± 0.123	0.221 ± 0.153	0.482 ± 0.444	0.516 ± 0.438	**0.519 ±**
1+/1-	ROC	0.804 ± 0.108	0.710 ± 0.121	0.771 ± 0.228	0.795 ± 0.203	**0.865 ±**
	PRC	0.301 ± 0.168	0.116 ± 0.121	0.500 ± 0.417	0.511 ± 0.470	**0.543 ±**

ECFP vs Graph Learned Embeddings

We also ran an experiment to test whether the molecular representation affects the performance in few-shot learning. These experiments are run on the Tox21 dataset, using Prototypical Networks, as these performed consistently well in our other experiments. ECFPs are based on the topology and a number of atom descriptors, in which the molecule is fragmented into local neighbourhoods and hashed into a vector. On the other hand, graph-learned embeddings are guided by gradient descent during training to produce a more relevant latent space embedding for the molecule. A neural network was used to learn a differentiable molecular embedding, from the ECFP, of the same size (a vector of size 128) as the one produced by the GCN. The results obtained using a learned embedding from GCNs consistently outperform the ones in which an ECFP was used.

Training Times

From the results on the Tox21 dataset, MNs, PNs and RNs obtain good predictive performance, however, it is evident from the presented result that the two latter networks are much faster to train on the same hardware. From our experiments on the three Tox21 targets, MNs and PNs were the most consistent in results. As the decrease in training times is substantial, by over 150% between MNs and both PNs and RNs, we believe that this puts the latter two networks at an advantage. Faster training times allow faster turnaround of results from datasets, while requiring less intense use of computer hardware. This increase in efficiency also allows scientists to perform a more rigorous hyperparameter search on various datasets in a shorter time.

Conclusion

In this study, we explore how a machine learning model can *learn how to learn* (hence meta-learning) and generalise using only a few examples for virtual screening. This research project builds on the work from Altae-Tran et al.[3], who have set important foundations for this problem domain. First and foremost, we reproduce their work effectively and provide deeper insight into the study by introducing PRC reporting, over and above the ROC scores. Secondly, we also introduce two new few-shot machine learning models to this problem domain and explore their performance against the state-of-the-art,[3] which we reproduce. Our reproduction provides results consistent with the original work. The introduced Prototypical Networks perform better on the Tox21 dataset based on ROC performance, while outperforming all other machine learning models in PRC performance. We believe that this is a valuable contribution as, in addition to obtaining better results than the state of the art, given the nature of the data used, the PRC provides more reliable insight into the performance of the models. We also find that making use of learned embeddings through GCNs, as opposed to ECFPs, consistently results in better ROC and

PRC performance. For datasets in which the ligands provided are structurally distinct, holding no relationship whatsoever between them, the conventional machine learning techniques, used as a baseline in our experiments, perform better.

Acknowledgement

Please use "The authors thank ..." rather than "The authors would like to thank ...".

The author thanks Mats Dahlgren for version one of achemso, and Donald Arseneau for the code taken from cite to move citations after punctuation. Many users have provided feedback on the class, which is reflected in all of the different demonstrations shown in this document.

References

(1) Hughes, J. P.; Rees, S.; Kalindjian, S. B.; Philpott, K. L. Principles of early drug discovery. *British journal of pharmacology* **2011**, *162*, 1239–1249.

(2) Waring, M. J.; Arrowsmith, J.; Leach, A. R.; Leeson, P. D.; Mandrell, S.; Owen, R. M.; Pairaudeau, G.; Pennie, W. D.; Pickett, S. D.; Wang, J., et al. An analysis of the attrition of drug candidates from four major pharmaceutical companies. *Nature reviews Drug discovery* **2015**, *14*, 475–486.

(3) Altae-Tran, H.; Ramsundar, B.; Pappu, A. S.; Pande, V. Low data drug discovery with one-shot learning. *ACS central science* **2017**, *3*, 283–293.

(4) Koch, G.; Zemel, R.; Salakhutdinov, R., et al. Siamese neural networks for one-shot image recognition. ICML deep learning workshop. 2015.

(5) Vinyals, O.; Blundell, C.; Lillicrap, T.; Wierstra, D., et al. Matching networks for one shot learning. *Advances in neural information processing systems* **2016**, *29*, 3630–3638.

(6) Snell, J.; Swersky, K.; Zemel, R. S. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175* **2017**,

(7) Sung, F.; Yang, Y.; Zhang, L.; Xiang, T.; Torr, P. H.; Hospedales, T. M. Learning to compare: Relation network for few-shot learning. Proceedings of the IEEE conference on computer vision and pattern recognition. 2018; pp 1199–1208.

(8) Wang, Y.; Yao, Q.; Kwok, J. T.; Ni, L. M. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR)* **2020**, *53*, 1–34.

(9) Lake, B. M.; Salakhutdinov, R.; Tenenbaum, J. B. Human-level concept learning through probabilistic program induction. *Science* **2015**, *350*, 1332–1338.

(10) Thrun, S.; Pratt, L. *Learning to learn*; Springer Science & Business Media, 2012.

(11) Finn, C.; Abbeel, P.; Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. International Conference on Machine Learning. 2017; pp 1126–1135.

(12) 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. *Chemical science* **2018**, *9*, 513–530.

(13) Jiang, D.; Wu, Z.; Hsieh, C.-Y.; Chen, G.; Liao, B.; Wang, Z.; Shen, C.; Cao, D.; Wu, J.; Hou, T. Could graph neural networks learn better molecular representation for drug discovery? A comparison study of descriptor-based and graph-based models. *Journal of cheminformatics* **2021**, *13*, 1–23.

(14) Gómez-Bombarelli, R.; Wei, J. N.; Duvenaud, D.; Hernández-Lobato, J. M.; Sánchez-Lengeling, B.; Sheberla, D.; Aguilera-Iparraguirre, J.; Hirzel, T. D.; Adams, R. P.; Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science* **2018**, *4*, 268–276.

(15) Duvenaud, D.; Maclaurin, D.; Aguilera-Iparraguirre, J.; Gómez-Bombarelli, R.; Hirzel, T.; Aspuru-Guzik, A.; Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints. *arXiv preprint arXiv:1509.09292* **2015**,

(16) Bromley, J.; Bentz, J. W.; Bottou, L.; Guyon, I.; LeCun, Y.; Moore, C.; Säckinger, E.; Shah, R. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence* **1993**, *7*, 669–688.

(17) Huang, R.; Xia, M.; Nguyen, D.-T.; Zhao, T.; Sakamuru, S.; Zhao, J.; Shahane, S. A.; Rossoshek, A.; Simeonov, A. Tox21Challenge to build predictive models of nuclear receptor and stress response pathways as mediated by exposure to environmental chemicals and drugs. *Frontiers in Environmental Science* **2016**, *3*, 85.

(18) NIH, Tox21 Data Challenge 2014. 2014; Accessed on 20.08.2021.

(19) Rohrer, S. G.; Baumann, K. Maximum unbiased validation (MUV) data sets for virtual screening based on PubChem bioactivity data. *Journal of chemical information and modeling* **2009**, *49*, 169–184.

(20) Mysinger, M. M.; Carchia, M.; Irwin, J. J.; Shoichet, B. K. Directory of useful decoys, enhanced (DUD-E): better ligands and decoys for better benchmarking. *Journal of medicinal chemistry* **2012**, *55*, 6582–6594.

(21) Bento, A. P.; Hersey, A.; Félix, E.; Landrum, G.; Gaulton, A.; Atkinson, F.; Bellis, L. J.; De Veij, M.; Leach, A. R. An open source chemical structure curation pipeline using RDKit. *Journal of Cheminformatics* **2020**, *12*, 1–16.

(22) Brecher, J. Graphical representation of stereochemical configuration (IUPAC Recommendations 2006). *Pure and applied chemistry* **2006**, *78*, 1897–1970.

(23) Food, F.; Administration, D. Substance Definition Manual. *Standard Operating Procedure, "Substance Definition Manual," Version 5c* **2007**, *94*.

(24) Mufei, L.; Jinjing, Z.; Jiajing, H.; Wenxuan, F.; Yangkang, Z.; Yaxin, G.; George, K. DGL-LifeSci: An Open-Source Toolkit for Deep Learning on Graphs in Life Science. *arXiv preprint arXiv:2106.14232* **2021**,

(25) Wang, M.; Zheng, D.; Ye, Z.; Gan, Q.; Li, M.; Song, X.; Zhou, J.; Ma, C.; Yu, L.; Gai, Y.; Xiao, T.; He, T.; Karypis, G.; Li, J.; Zhang, Z. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. *arXiv preprint arXiv:1909.01315* **2019**,

(26) Kipf, T. N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv preprint arXiv:1609.02907* **2016**,

(27) Smusz, S.; Kurczab, R.; Bojarski, A. J. The influence of the inactives subset generation on the performance of machine learning methods. *Journal of cheminformatics* **2013**, *5*, 1–8.

(28) Wallach, I.; Heifets, A. Most ligand-based classification benchmarks reward memorization rather than generalization. *Journal of chemical information and modeling* **2018**, *58*, 916–932.

(29) Chen, L.; Cruz, A.; Ramsey, S.; Dickson, C. J.; Duca, J. S.; Hornak, V.; Koes, D. R.; Kurtzman, T. Hidden bias in the DUD-E dataset leads to misleading performance of deep learning in structure-based virtual screening. *PloS one* **2019**, *14*, e0220113.