Few-Shot Learning for Low-Data Drug Discovery

Daniel Vella[†] and Jean-Paul Ebejer[*,†,‡]

[†]*Department of Artificial Intelligence, University of Malta, Msida, MSD 2080, Malta.*

[‡]*Centre for Molecular Medicine and Biobanking, University of Malta, Msida, MSD 2080, Malta.*

E-mail: jean.p.ebejer@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. The requirement for large amounts of training data hinders the application of conventional machine learning techniques to this problem domain. This work explores few-shot machine learning for lead optimisation and hit identification. 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 using different embeddings, namely extended-connectivity fingerprints (ECFP) and embeddings generated through graph convolutional networks (GCN), as inputs to neural networks for classification. This study shows that learned embeddings through GCNs consistently perform better than extended-connectivity fingerprints for toxicity and LBVS experiments. 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 based on the Matching Networks architecture. Additionally, training these networks is substantially faster (up to 190%) and therefore takes 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. This ability is in stark contrast with conventional supervised 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 new problems by 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 particularly beneficial in domains where data availability or acquisition is difficult, such as the drug-discovery domain. The drug-discovery process's primary goal is identifying and developing active compounds that exhibit therapeutic effects against biological targets. 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 later *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 of 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] in contrast to conventional machine learning, which is data-hungry. In recent years, the computer vision domain 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 learning rare cases possible.[8]

Building on this notion, we explore few-shot learning to address the low-data problem for

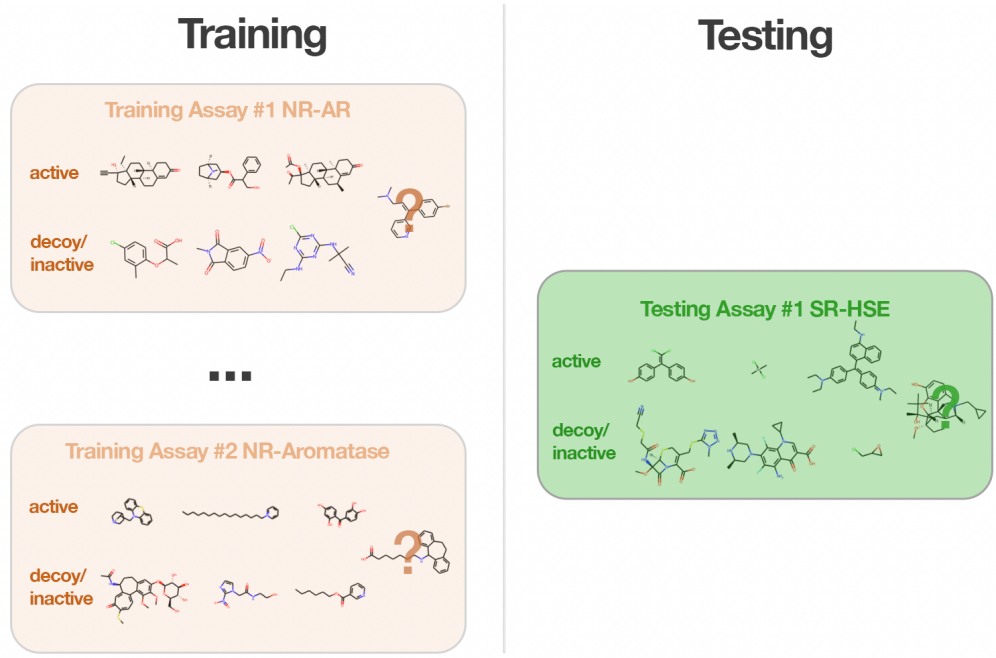

Figure 1: 2-way 3-shot few-shot classification. Training a meta-learner on experimental assays and generalising for an unseen assay in the Tox-21 dataset.

hit identification and lead optimisation. The ability of a machine learning model to learn new concepts fast with just a few training examples is invaluable for this domain, where data on active compounds is scarce. Meta-learning aims to achieve generalising capabilities for previously unseen environments during training time. In few-shot classification, a meta-learning paradigm, we train models using a variety of training tasks and optimise classification 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. N refers to the number of classes we include per task, while K refers to the number of molecules we sample for each class to make up the support set.[6] For this study, few-shot learning refers to training with as little as one example per class, referred to as one-shot learning,[4,5] to a maximum of ten examples per class. During test time, a small support set is sampled from new, previously unseen targets, and the model uses these few data points to generalise query molecules' activity against this new target.[5] Figure 1 shows an example of a typical meta-learning scenario on the Tox21 dataset[9], where

data from a set of assays reserved for training are used to train a model. This model is subsequently used to generalise for a previously unseen assay using only a small support set from this new assay. We highlight that few-shot learning in this problem domain differs from other domains, such as computer vision, where a model is trained to recognise new classes. For example, given a few images of a lion, a class unseen during training, as the support set, the model must generalise for new images of a lion. In the domain under study, the challenge is to train a model that can generalise for the behaviour of molecules in experimental assays which are related but not identical to the assays in the training collection, using only a small support set from these new experimental assays. The molecules used during testing can thus be previously seen during training, but only in the context of their activity for different, but related experimental assays. Given a few molecules from new experimental assays, can the model predict the activity of molecules in this new assay using molecular data for different but related targets as training data?

Molecules are complex structures consisting of atoms and bonds which must be somehow represented in computational space. The classical notation of compounds is the empirical formula such as $C_3H_7NO_2$. However, this can refer to alanine, sarcosine, or lactamide as empirical formulae hold no information on a molecule's topology. Molecular representations such as Extended-Connectivity Fingerprints (ECFP),[10] and graph convolution learned embeddings[11] embed more information than the empirical formula on the properties of the molecule. This study mainly explores using graphs as embeddings for the low-data machine learning networks. Formally, a *graph* is a set of nodes and a set of edges, where each edge connects a pair of nodes. This notion intuitively translates to molecular representations where atoms form the set of nodes, and the bonds form the set of edges. Graphs are 2D objects, so spatial properties of a molecule, such as bond angles and chirality, are not inherent to the data object but are instead encoded as node or edge attributes.[12] Embeddings of molecular graphs, augmented with atom feature information, can be learned using graph convolutional networks (GCNs). Wu et al.[13] report that learned embeddings could be of

benefit over topological molecular representations such as ECFP.

In this study, we explore the application of several 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 all fall under the umbrella of metric-based meta-learning. In our study, we embed molecule representations using GCNs, 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.

Related Work

Several successful research undertakings have exploited the low-data learning paradigm, especially in the computer-vision domain.[4–7] Learning from only a few examples is especially important in domains with a paucity of data. This inaccessibility could be attributed to privacy, safety, or ethical issues and other issues such as the time, resources and exorbitant costs associated with data acquisition. Learning with low-data can lead to less expensive data gathering, and reduced computational cost for learning.[8]

Building on past work in the metric-based meta-learning sphere,[5] Altae-Tran et al.[3] introduce a deep-learning architecture for few-shot learning in drug discovery, 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. In our research, we build on the work by Altae-Tran et al.[3] and extend it by applying other successful few-shot learning approaches explored for other domains, such as the computer-vision domain. The authors employ Graph Convolutional Networks (GCN) to learn molecular embeddings, which are then fed into the low-data architectures for classification.

Graph Convolutional Networks

Molecules must be represented in computational space before processing them using few-shot machine learning techniques. Wu et al.[13] report that graph-based models outperform conventional machine learning models on most datasets, suggesting that a learned embedding is advantageous over other molecular representations. Building on this rationale, we opt for graph learned molecular representations to embed the input molecules in this study. 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 connect two atoms (see Equation 1). In addition to bond information, 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 Convolutional Networks (GCNs) may be used to learn molecular representations.[14] Embeddings learned through neural networks afford the construction of automated features rather than fixed fingerprints. GCNs transform small molecules into real-valued vector representations, which are an effective way of processing small molecules via deep neural networks.[15] Duvenaud et al.[11] report that using a differentiable method reduces collisions of substructures, and the learned embedding can be optimised to contain relevant features such as biological activity and substructure information.

If the graph object is our input signal, we can apply a set of operators to approximate the function we are attempting to learn. Bronstein et al.[16] propose four key building blocks for deep learning on graphs, which include linear set equivariant layers, non-linear functions, local pooling layers, and set invariant layers. For graphs, the nodes v are found on a domain Ω such that $v \in \Omega$. The nodes in Ω are stored in a feature space C, such that $C = \mathbb{R}^k$. Using a set of feature functions $X(\Omega, C)$, we can transform the feature space of the nodes in our

domain.

In the equivariant layer B, we can take the nodes in our domain and apply a function that transforms the features of the nodes such that $X(\Omega, C) \rightarrow X(\Omega', C')$. Equivariance allows for a function g to be applied before or after this layer, such that $B(g.x) = g.B(x)$. The non-linear activation functions can be applied element-wise on the features of the nodes in a graph, such that $(\sigma(x))(v) = \sigma(x(v))$. Local pooling layers can be used to apply coarsening to the graph such that $X(\Omega, C) \rightarrow X(\Omega', C)$, in which we can reduce the number of nodes in our domain such that $\Omega' \subseteq \Omega$. Finally, we have the invariant layer Z, which can also be referred to as a global pooling layer, in which $X(\Omega, C) \rightarrow y$, which satisfies the invariant condition such that $Z(g.x) = Z(x)$[16]. Figure 2 illustrates an example of a GNN to learn a molecular embedding.

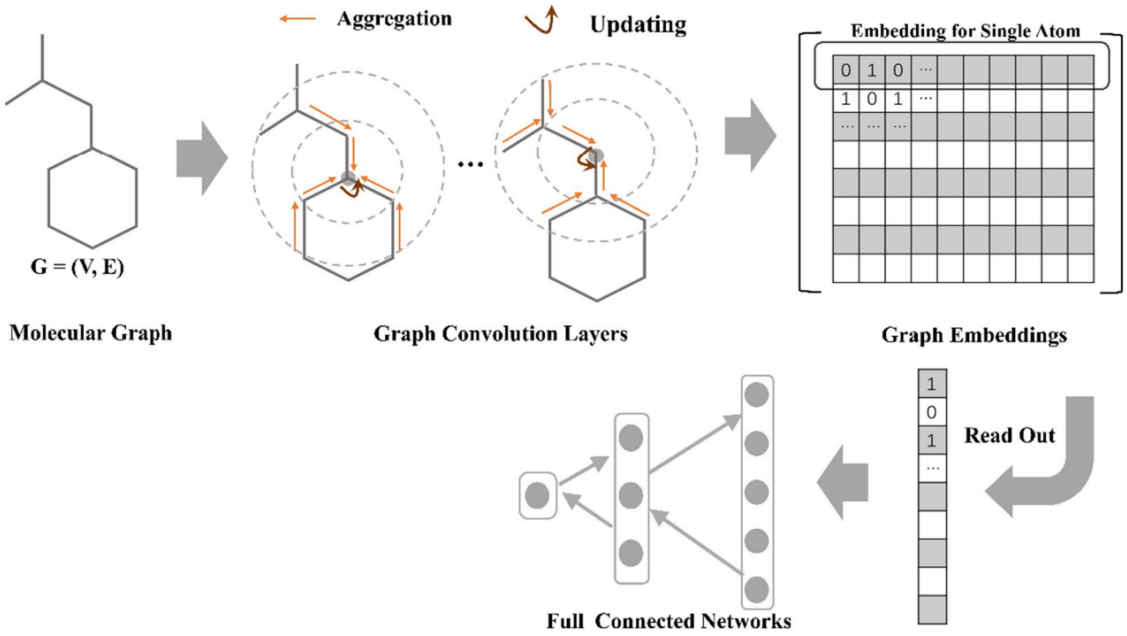

Figure 2: A typical pipeline for representing molecules using a learned embedding function, which can be processed further using feedforward neural networks as shown. Reproduced from Jiang et al. [14].

Metric-based Few-Shot Learning

The success of a few-shot learning model for metric-based meta-learning is dependent on the effectiveness of a kernel k_θ, which measures the similarity between data samples $x..x_i$ from a support set S (see Equation 2) using a metric or distance function. The techniques employed in this study, excluding the benchmark model, use the support and query embeddings generated from the GCNs to learn the kernel function. We explore four few-shot learning models in this study, which are presented in the Methodology section. These include Siamese Networks, Matching Networks (upon which the state-of-the-art is developed), Prototypical Networks and Relation Networks. The two latter networks are new approaches for this problem domain and are mostly inspired by the computer vision domain.

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

Siamese Networks

Siamese networks[4] are composed of two identical networks with shared weights and parameters, taking in a pair of data samples (molecules) as inputs. The distance between outputs from each component in the pair of networks is calculated to learn their relationship. The following is the process, repeated for all training tasks, employed for learning a classifier using Siamese Networks.

1. Generate a list of all possible pairs between training data. If both data samples in the pair have the same target, the pair's label is set to one and zero if otherwise.

2. Create a twin network using the GCN architecture to embed two molecular graph inputs into latent space.

3. Calculate the L1 distance between the molecule embeddings.

4. The distance between the two embeddings is passed through a linear feedforward layer,

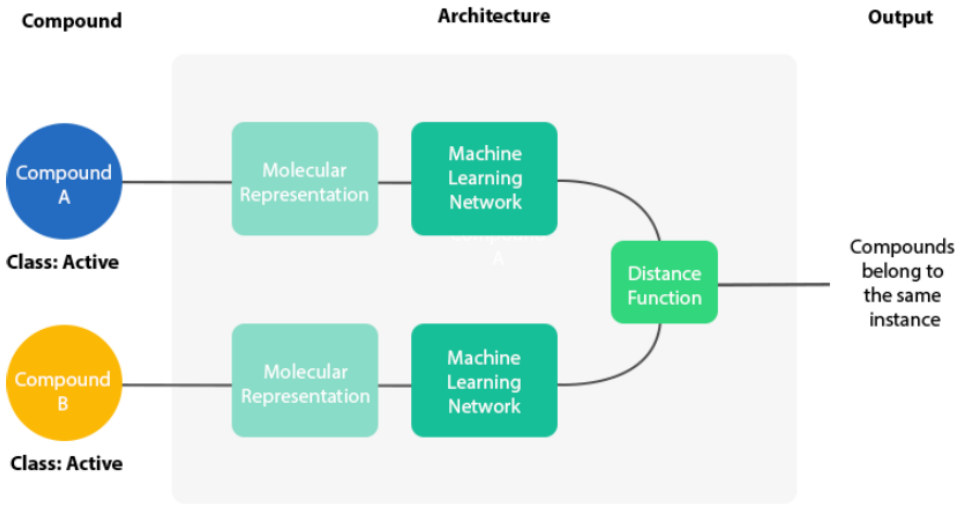

Figure 3: High-level schematic of a Siamese network for the molecular network.

followed by a sigmoid function to output the probability that the two molecules belong to the same class.

5. The binary cross-entropy loss is calculated and backpropagated through the network.

Matching Networks

The Matching Networks architecture builds on Siamese Networks. However, instead of learning a metric function over data pairs, the classifier learns how to define a probability distribution of output labels from query examples using a support set S. The classifier outputs a sum of attention-weighted labels from the support set to predict the similarity between the test example and the support set samples. We use the same embedding function for the support and query sets to compute the molecular embeddings. Subsequently, the cosine similarity of 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.

Figure 4 illustrates the Matching Nets architecture. Embedding functions f and g are Convolutional Neural Networks (CNNs)[17], potentially being identical to each other, which

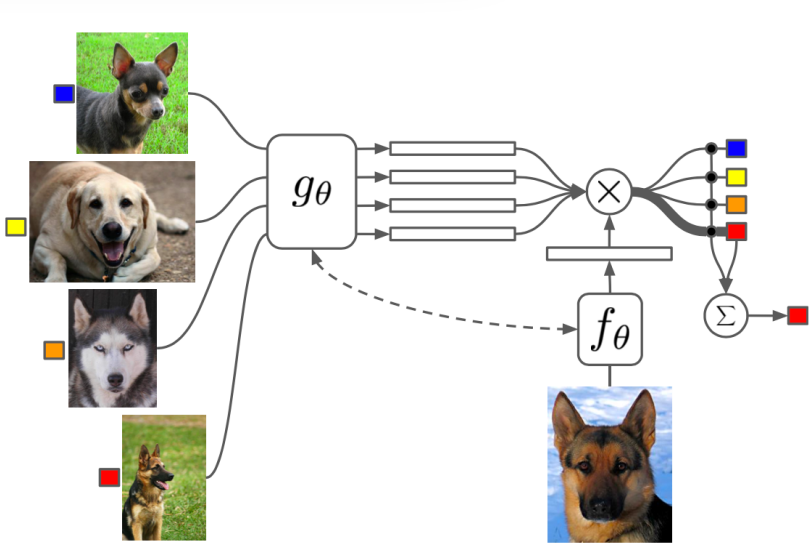

Figure 4: Matching Networks Architecture. Reproduced from Vinyals et al. [5].

project the inputs to the feature space. Vinyals et al. [5] also propose full context embedding functions, which take as input the whole support set with the element x_i, thus resulting in $g(x_i, S)$. Full context embeddings effectively modify how the element is embedded with respect to the whole support set S. A bidirectional LSTM is used to encode x_i in the context of the support set. Finally, the attention mechanism a, is the classifier which takes a softmax over the cosine distance of the embeddings. As proposed in Vinyals et al. [5], fully contextual embeddings (FCE) are used in our implementation. Taking single data points to learn an embedding function limits the ability to embed the molecules effectively into latent space. Therefore, a bidirectional long-short term memory (LSTM) is used, which takes the whole support set as input to adjust the embedding based on the other support samples. $g_\theta(x_i, S)$ encodes x_i, a data sample from the support set, in the context of the whole support set S, using a bidirectional LSTM. The LSTM transforms our support set embeddings by adding the forward and backward activations to the original support image embeddings. Subsequently, $f_\theta(x, S)$, encodes the query sample x and trains the LSTM with read attention over the support set. The hidden state is updated over ten processing "read" steps until, eventually, the hidden state is equivalent to the aforementioned $f_\theta(x, S)$. The

hidden state and the output from the attention function are added in each iteration. The cross-entropy loss is computed for each query prediction to train the model using stochastic gradient descent.

Prototypical Networks

Prototypical Networks[6] have similarities to Matching Networks, but instead of considering the individual support set embeddings, the mean vector of the embeddings (referred to as the *prototype*) for each class within the support set is taken. Another improvement the authors make over Matching Networks is using Euclidean distance rather than the cosine distance to calculate the distances to classify the query (refer to Figure 5). In order to classify query data samples, the softmax of the Euclidean distance's inverse between each query and each prototype is taken. The negative log-likelihood is used to train the network through stochastic gradient descent

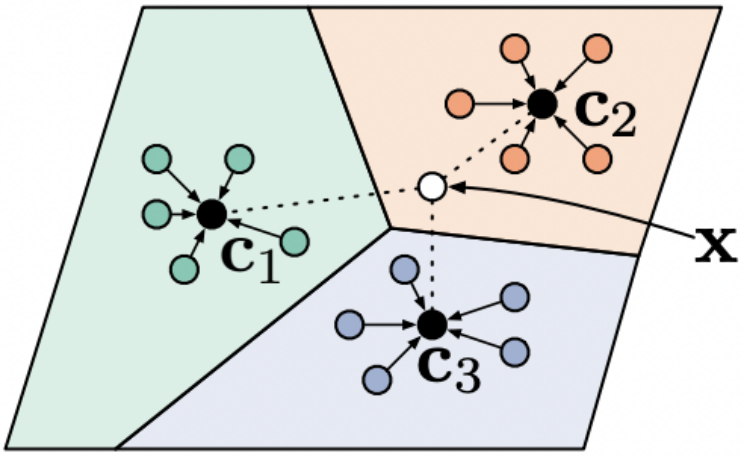

Figure 5: Few-shot learning in Prototypical Networks, where prototypes c_k are taken as the mean of embedded support examples for each class. Reproduced from Snell et al.[6].

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. Unlike previously mentioned networks, this network uses a feedforward neural network to learn a distance function in feature space. After embedding the support and query examples using an embedding function, each query example is concatenated with each feature map from the support set. The relationship between the queries and the different classes within the support set is captured by passing the feature map concatenations through a feed-forward neural network $g_\theta([x_i, x_j])$ to predict a relation score. The output class can be inferred from this relation score vector. $[\cdot, \cdot]$ is the concatenation between each support set data sample x_i and the query data samples x_j. The Mean Squared Error (MSE) is used as the loss function, as proposed in the original paper.

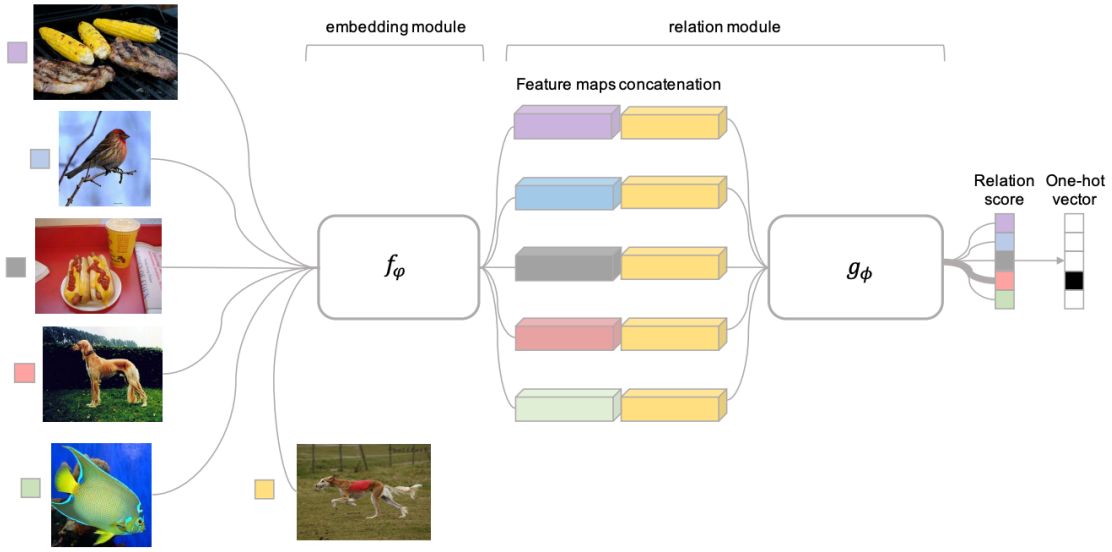

Figure 6: Few-shot learning scenario in Relation Networks for a 5-way 1-shot learning task with one query as an example. Reproduced from Sung et al.[7].

Iterative Refinement LSTM

Altae-Tran et al.[3] build on meta-learning concepts by training machine learning models on molecular data from a set of experimental assay targets (from the Tox21, SIDER, and MUV datasets) reserved for training. The model is then used to generalise for the activity of molecules in new, previously unseen experimental assays using only a small support set from these new assays. These test assays are related but not identical to the ones reserved for training. The number of molecules sampled for each class in the support set ranges from one to a maximum of ten molecules. The support and query molecules are embedded in their work using a graph convolutional network. Bond information and distinction between bond types was not considered in their study. We note that the *pool* layers do not coarsen the graphs but only apply a max function over neighbouring nodes.

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 query embedding is built iteratively with that of the support set, using information from the two sets to enhance both the support and query embeddings. Once the embeddings have been iteratively refined, the authors apply a metric-based function to classify the queries using the support set embeddings. To emulate the Matching Networks, the authors use the Cosine distance to compare embeddings. Figure 7 illustrates a one-shot learning scenario encapsulating the concepts mentioned earlier.

Their work is evaluated on the Tox21, the Side Effect Resource (SIDER)[18], and MUV datasets[19]. For every dataset, a subset of the targets is reserved for training and the rest for testing. Training is carried out as explained in the Matching Networks paper, in which training conditions match those at test time[5]. The authors use a Random Forest (RF) with 100 decision trees as a machine learning baseline model. They also utilise a conventional Graph Convolutional Networks (GCN)[20] as an additional baseline model, which is trained using only a small support set from the test targets. They then experiment with Siamese

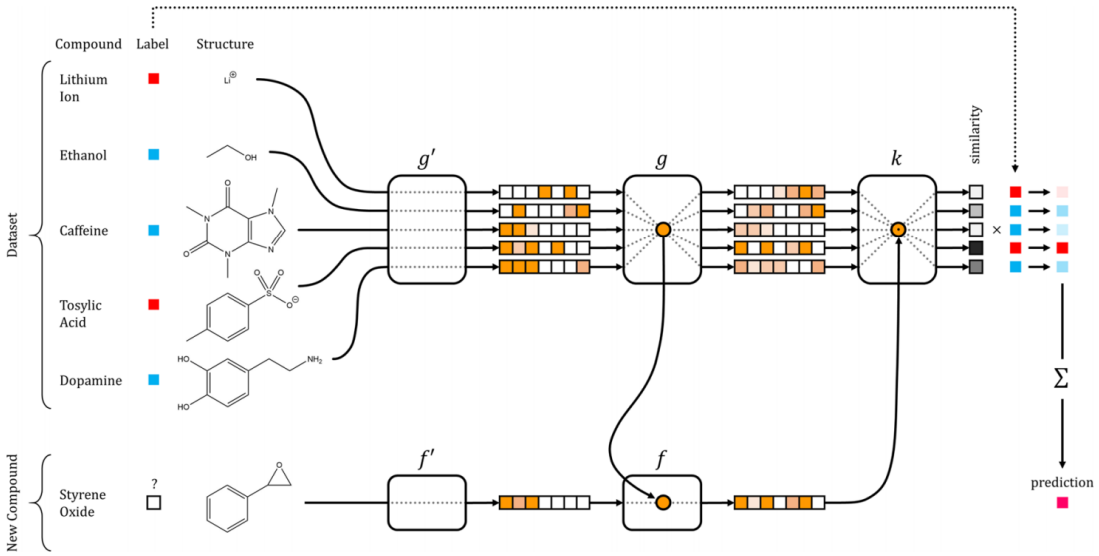

Figure 7: Schematic of one-shot learning in drug discovery based on the Matching Network[5] architecture. Reproduced from Altae-Tran et al.[3].

Networks[4], Matching Networks[5], and an adaptation of the Matching Networks by applying the iterative refinement concepts explained earlier.

The authors utilise ROC-AUC scores to report the performance of the models. Considering the extreme imbalance of the data in the utilised datasets, favouring the negative (inactive/decoy) class, we note that the PR-AUC score would be a more appropriate evaluation measure. PR-AUC is based on the relationship between precision and recall, providing a clearer picture of how the model performs when predicting the *positive* (active) class in the data. Correctly predicting the "active" class is of significant importance in virtual screening.

On the Tox21 and SIDER datasets, their proposed machine learning architecture achieves good ROC-AUC performance. The mean score for 10-shot learning on the median held-out task on Tox21 achieves a score of 0.823 ± 0.002, while for one-shot learning, the model achieves a mean score of 0.827 ± 0.001. The reasons why one-shot learning achieved better performance than 10-shot learning is uncertain, as we expect the model to perform better with larger support sets. However, this might be attributed to variance in the data between experiments. On MUV data, the baseline machine learning models performed few-shot learning. The authors report that this is due to MUV data being maximally informative; therefore,

structural similarity cannot be utilised to generalise activity prediction. The authors open-sourced their models in the DeepChem library[21]. However, the implementations are now outdated and not executable with the more recent versions of the DeepChem library, which makes reproducing results difficult. However, we study the open-sourced implementation and the implementation details in the original literature to reproduce this work successfully.

Methodology

Building on the work of Altae-Tran et al.[3], we implement a Random Forest and a GCN as benchmark models. Additionally, we implement four few-shot machine learning architectures: Siamese Networks, Matching Networks, Prototypical Networks, and Relation Networks. IterRefLSTMs[3], used in the state-of-the-art, are used to enrich the resulting embeddings in latent space. Molecules are represented as graph objects and subsequently processed using GCNs to produce a vectorised embedding in computational space. We try to follow the implementation of Altae-Tran et al.[3] as closely as possible for reproducibility and homogenising of results for effective comparison.

Machine Learning Pipeline

The machine learning pipeline for this study consists of nine main parts, which are illustrated in Figure 8 and described hereunder:

1. **Data Acquisition**. We utilise three main publicly available datasets for this study, namely, Tox21, MUV, and the GPCR subset of DUD-E. The latter is a new contribution to the study by Altae-Tran et al.[3]. All data is available as SMILES strings with a flag for the experimental assays in the dataset recording whether the molecule is active or an inactive/decoy.

2. **Standardise molecule**. The SMILES strings are first standardised to transform all

molecular representations according to a set of well-defined and consistent rules and conventions to ensure validity and uniformity.

3. **Molecular features generation**. The molecular graph generated from the standardised SMILES representation is enriched with atom descriptors to add information to the molecular representation.

4. **Molecular graphs generation**. The molecular representations are transformed into graph objects, consisting of nodes and edges representing atoms and bonds, respectively. The connectivity between atoms is represented via an adjacency matrix.

5. **Episode generation**. Effective few-shot learning necessitates that conditions at training match those at testing[5]. Therefore, N-way K-shot support sets and queries are randomly sampled to form a series of episodes for training. K in each support set is constrained between one and ten examples per class. For every episode, we have two classes, the active class and the inactive/decoy class.

6. **Learning a molecular embedding**. The sampled molecules in each episode are used to learn a molecular embedding using GCNs.

7. **Few-Shot Learning**. The learned embeddings are processed using four different meta-learning architectures: Siamese, Matching, Prototypical and Relation Networks. IterRefLSTMs[3] are used to enrich the model. A subset of experimental assays in each dataset is reserved for training, while the rest are reserved for testing.

8. **Testing**. The trained models are subsequently used to test on new experimental assays, previously unseen during training, to gauge the generalising capability of a model trained for a low-data scenario. Support sets are randomly sampled and trained on the remaining molecules in the dataset for 20 rounds. The mean and standard deviation of the areas under the curve for Precision-Recall (PR-AUC) and Receiver Operator Characteristic (ROC-AUC) are calculated to quantify performance.

9. **Evaluation**. Finally, we evaluate the results based on the ROC-AUC and PR-AUC scores from the 20 test rounds. We apply statistical analysis for results obtained across different experiments to determine the best-performing techniques for each support set composition. Confusion matrices, ROC-AUC and PR-AUC graphs for the experiment with the median ROC-AUC score from the 20 rounds are generated after test completion.

Datasets

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

- **Tox21**[9] – Mainly used for lead optimisation, containing toxicity data for 12 targets[22]. 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 are used 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-548, MUV-600, MUV-644, MUV-652, MUV-689, MUV-692, MUV-712, MUV-713, MUV-733, MUV-737, and 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) (DUD-E)**[23] – Used for benchmarking virtual screening techniques by introducing active compounds against specific targets. Many *decoys* with similar physical properties but different topologies are made available

[1]Accessed from: `https://deepchemdata.s3-us-west-1.amazonaws.com/datasets/tox21.csv.gz`. Last Accessed: 26/05/2022

[2]Accessed from: `https://deepchemdata.s3-us-west-1.amazonaws.com/datasets/muv.csv.gz`. Last Accessed: 26/05/2022

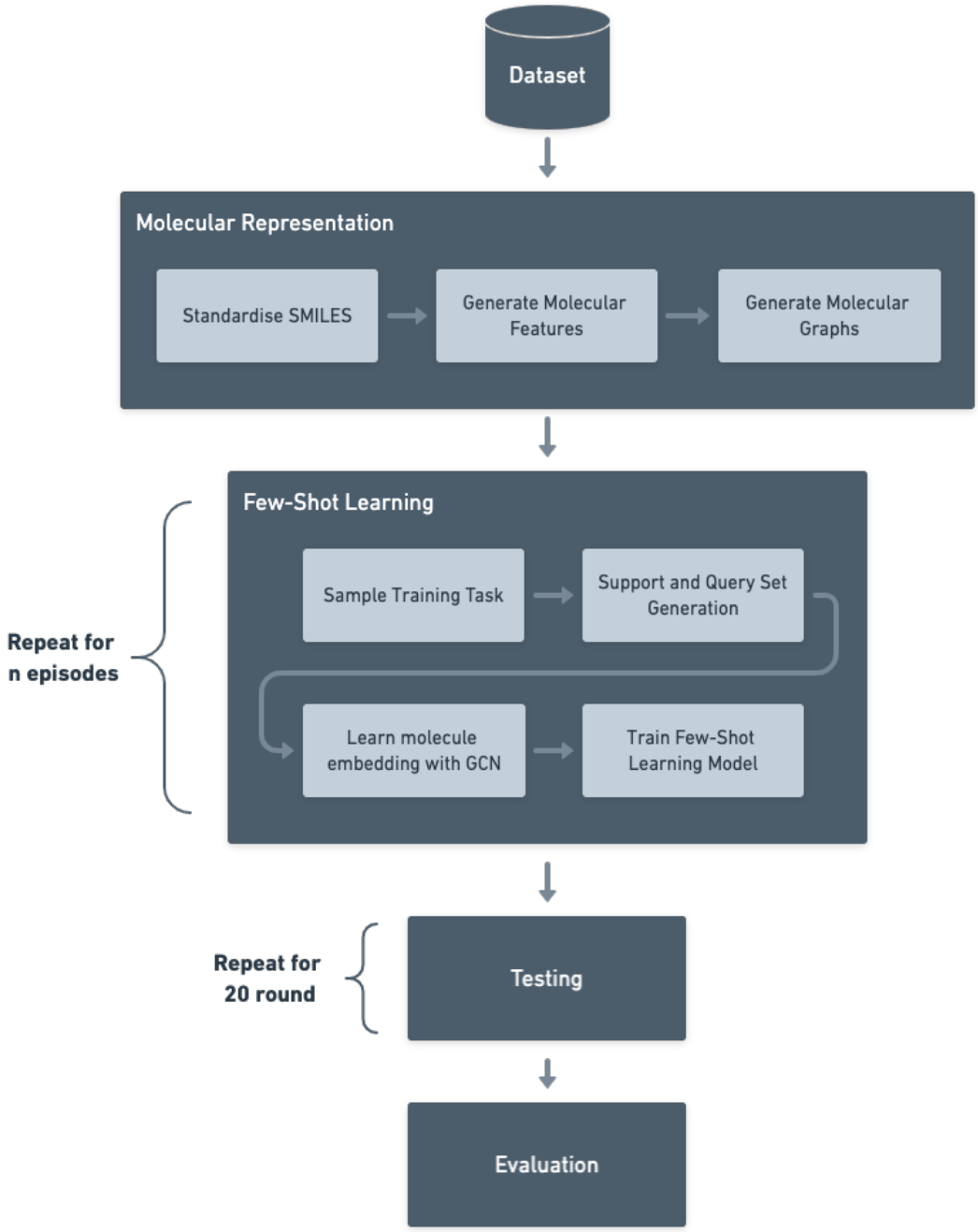

Figure 8: Schematic of the machine learning pipeline designed for this study. The changes across few-shot learning architectures lie in the 'Train Few-Shot Learning Model' component. Otherwise, all other modules remain identical. Figure 7 visualises the GCN learned embedding as g' and f', and the few-shot learning steps as g, f, k.

for each active molecule. We used the GPCR subset of the DUD-E dataset[23] for

this research study. The data was obtained directly from the DUD-E website.[3] The AA2AR, DRD3, and ADRB1 targets are used for training. Two targets, ADRB2 and CXCR4, are reserved for testing. We note that this is an additional contribution to the study from Altae-Tran et al.[3].

Table 1 shows the excessive imbalance of these datasets, highlighting the scarceness of data on active compounds in this domain. Due to this class imbalance, we also report PR-AUC metrics in addition to the ROC-AUC metrics presented in the study by Altae-Tran et al.[3].

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 standardise the molecules according to well-defined and consistent rules and conventions. Maintaining uniformity and integrity across the different data is of utmost importance. Bento et al.[24] present an open source chemical structure curation pipeline based on RDKit[25] for validating and standardising chemical structures, which follow FDA/IUPAC guidelines[26,27]. Their work is available in the ChEMBL Structure Pipeline package[24] and is used to standardise the molecules in our pipeline.

We create a molecular graph from the SMILES string of the standardised molecule using RDKit, an open-source toolkit for cheminformatics. We then one-hot encode eight atom features in each molecule: 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

[3]Accessed from: `http://dude.docking.org/subsets/gpcr`. Last Accessed: 26/05/2022

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 the Deep Graph Library (DGL) LifeSci [28] library to create the graph objects and subsequently process them using the DGL library. [29]

Episodic Learning

Training for few-shot learning is carried out in a series of episodes, as shown in Figure 8. We consider N-way K-shot classification tasks for each episode, where the support set contains N classes and K labelled molecules. The tasks in our research are binary classification tasks. Therefore N is always set to two to represent the active and the inactive/decoy class, respectively. Experiments with varying values of K are carried out to generate the support sets, with a minimum of one data point, to a maximum of ten data points (molecules) per class. The combinations for K active and inactive/decoy classes are not exhaustive, but we follow the support set composition used in Altae-Tran et al. [3] to compare results with this study directly.

Table 2 contains the composition of the support sets used in our experiments. For each episode, we sample a total of 128 query molecules 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. The choice of the support set composition was based on the methodology presented in Altae-Tran et al. [3].

Table 2: Support set composition

Actives	Inactives/Decoys	Support Set Size
10	10	20
5	10	15
1	10	11
1	5	6
1	1	2

Machine Learning Models

Before processing the molecular graph, the model first learns an embedding using GCNs. Four different architectures, including Siamese, Matching, Prototypical and Relation Networks, subsequently process the learned graph embeddings to train our meta-learner.

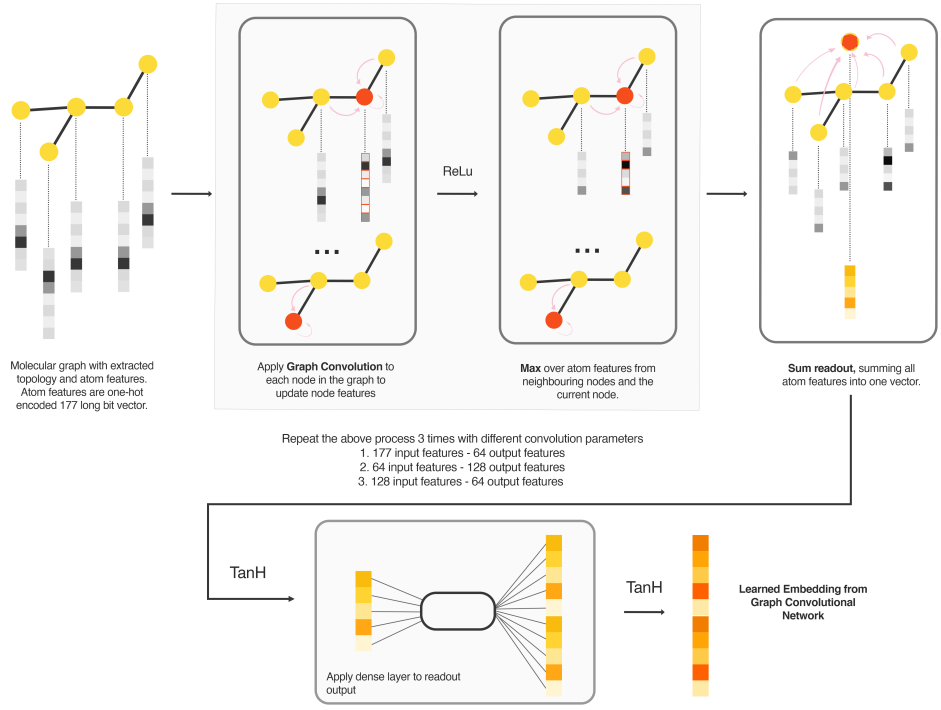

Figure 9: Learning an embedding through a Graph Convolutional Network (GCN). The molecule, represented as a graph object with nodes, edges, and atom features, is processed using graph convolutions. A max message-passing function over the current and neighbouring nodes follows each convolution layer. 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.

Graph Convolutional Networks

Graph convolutional networks (GCNs) are used to learn embeddings for the support and query molecules in latent space. The SMILE molecules are first converted to graph representations and later embedded as vectors in latent space using GCNs and a final dense neural network. Figure 9 illustrates the GCN pipeline to learn a molecular embedding.

Our study uses the convolutional operator from Kipf and Welling [20] to process graphs and learn molecular embeddings. We note that back-propagation occurs throughout the whole few-shot learning process, which includes the GCNs. The assays used in testing are unseen during training. However, the molecules used in testing can be encountered during training for different experimental assays. Since they are included in the same dataset (e.g. Tox21 data), these assays are related but not identical.

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

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

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 $h_u = f(W_u \hat{h}_u)$ [20].

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 remains the same. Finally, we apply a global pooling layer (readout), in which we sum over the node features of every node in the graph (see Equation 4).

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

A linear transformation is applied to the output from the readout layer, followed by a non-linear activation function, which uses a hyperbolic tangent function (tanh), outputting

the final molecule embedding. Table 3 contains the architecture utilised for the GCN in this study, and is illustrated in Figure 9.

Table 3: Graph Convolution Network Architecture

Layer Type	Input Dimension	Output Dimension	Non-Linearity
GraphConv	177	64	ReLU
Max Pooling	64	64	
GraphConv	64	128	ReLU
Max Pooling	128	128	
GraphConv	128	64	ReLU
Max Pooling	64	64	
SumPool Readout	64	64	tanh
Linear	64	128	tanh

Benchmark Models

We use 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. The Random Forest model uses ECFP representations of the molecules of size 2,048 bits for the classification task, using a radius of two. Meanwhile, the same GCN architecture used for the few-shot learning models is used for our benchmark. The designated architecture for the graph convolution network is outlined in Table 4. The only addition to the architecture is a final linear, fully-connected layer that takes as input 128 features, which is the size of the embedding used for the experiments to follow. It 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 binary target $(0, 1)$. This binary target signifies whether the molecule belongs to the experimental assay's active or inactive/decoy class. These two models are trained on a small support set, sampled from the targets assigned for testing. The remaining molecules that are not utilised for training are used to predict the respective class.

We also perform a final benchmark test in retrospect by taking a random selection of query molecules from a test target, generating the ECFP with the same parameters as mentioned earlier and then calculating the Tanimoto distance to classify the remaining test

molecules based on this distance. However, we find that this does not hold any significant predictive capability.

Table 4: Benchmark Neural Network for Few-Shot Learning

Layer	Input Dimension	Output Dimension	Non-Linearity
GraphConv	177	64	Relu
Max Pooling	64	64	
GraphConv	64	128	Relu
Max Pooling	64	64	
GraphConv	128	64	Relu
Max Pooling	64	64	
Sum Pooling	64	64	TanH
Linear	64	128	TanH
Linear	128	1	Sigmoid

Few-Shot Learning Models

The generated molecular embeddings from the GCN are used as inputs for the few-shot learning architectures. The models discussed in this section, excluding the benchmark model, use the embeddings generated from the GCN, grouped in support and query sets, to learn the kernel function. These models include Siamese Networks, Matching Networks, Prototypical Networks and Relation Networks. Table 5 tabulates the network used to generate the relation score in the Relation Networks architecture. The GCN architecture remains unchanged in all experiments to compare the model's effectiveness objectively. Figure 10 illustrates the rationale behind few-shot machine learning for this problem domain. The molecules used in training and testing can be shared; however, the target information imparting activity in a specific experimental assay must be different. This separation entails that we present the machine learning model with information from previously unseen experimental assays during testing. For example, training can be done on molecular activity for nuclear receptor assays in Tox21 and then tested on the remaining assays, which would have never been seen during training.

We reproduce the work of Altae-Tran et al.[3] from scratch and apply the IterRefLSTM

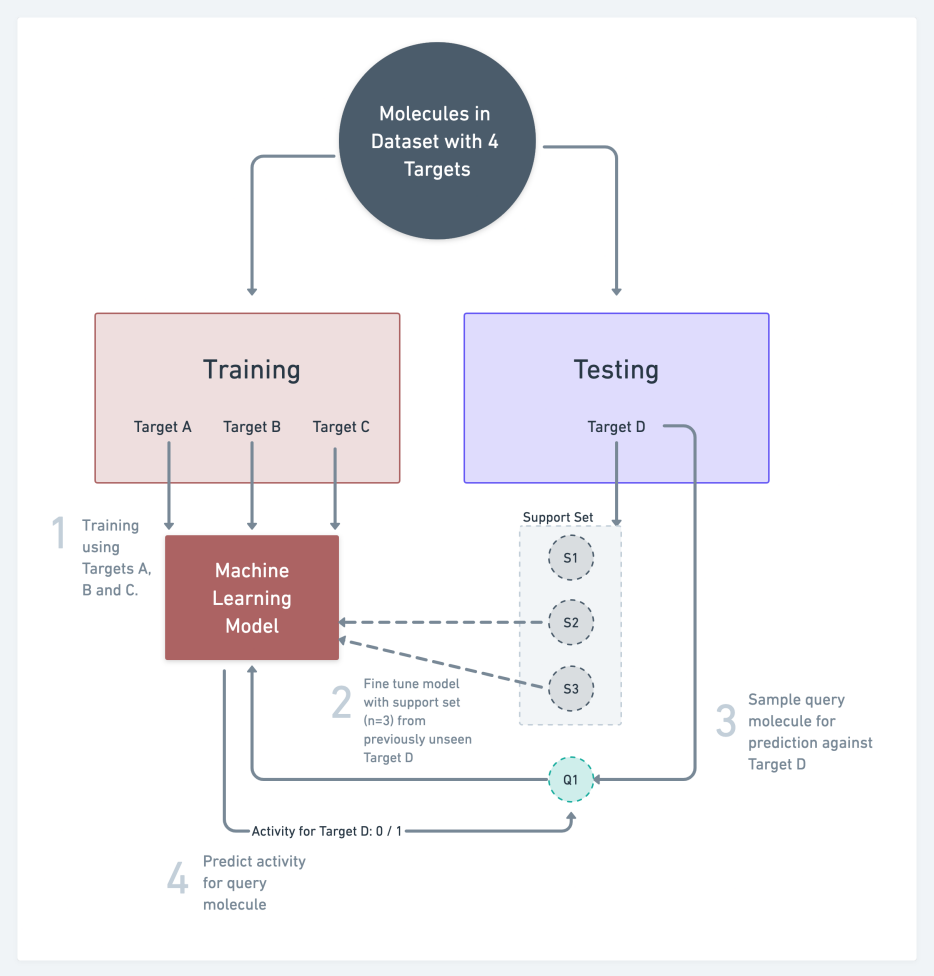

Figure 10: Schematic illustrating how the few-shot learning machine learning on a molecular dataset works. The same molecules can be used during training and testing; however, these molecules would impart different information based on the target or experimental assay. The targets used in testing are previously unseen, except for the few molecules sampled in the support set ($S_{1\text{-}3}$), which are used to fine-tune the ML model and impart some information about Target D. Query molecule Q_1 is sampled from Target D to be predicted using the ML model.

to the embeddings in all architectures 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]. As emphasised by Vinyals et al.[5] and Snell et al.[6], training and testing conditions should match during few-shot learning. Therefore, the same support set

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

Table 5: The architecture for generating the relation score using function g_θ.

Layer Type	Input Dimension	Output Dimension	Non-Linearity
Linear	256	128	ReLU
Linear	128	64	ReLU
Linear	64	8	ReLU
Linear	8	2	Sigmoid

composition used to train the model is used during test time. For example, if we train using 10-shot learning, testing is carried out with 10-shot support sets. We remind the reader that testing is carried out on a new, previously unseen target. After the support set has been sampled, the rest of the data for the target being tested is used as query data. This process is repeated 20 times, and the mean and standard deviation of the ROC-AUC and PR-AUC scores from these 20 rounds are reported as the final classification result.

Evaluation

The evaluation metrics we use are the Area Under Curve (AUC) of the Receiver Operating Characteristic (ROC) curve and the Precision-Recall Curve (PR-AUC). To determine our classifier's predictive power, we use the ROC Area Under Curve (AUC) (ROC-AUC) as this affords a more nuanced approach than the accuracy metric, providing visibility into thresholds one can utilise to ameliorate predictions. Altae-Tran et al. [3] also report ROC-AUC results; however, we believe this metric alone does not adequately measure the performance of the machine learning models due to the highly imbalanced nature of the data at hand. In virtual screening, detecting 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, PR-AUC is used to evaluate how well the model can classify the active class.

We apply statistical analysis to the ROC-AUC and PR-AUC scores from the 20 test

rounds for each experiment to establish whether there are significant differences between the few-shot learning models. The scores are compared against the scores from the model that obtained the best result for the same conditions. Comparison of results between two models is carried out using the Mann-Whitney U-test, also referred to as the Wilcoxon rank sum test[30].

Implementation Details

The machine learning models were developed using Python 3.7. Most packages were installed using Pip 21.0.1; however, Conda 4.10.3 was also used to install packages not found on the Python Package Index (PyPi)[5]. Pip and conda are package management systems for Python, allowing users to install and run packages and their dependencies conveniently. The specific versions of each toolkit are specified in Table 6.

Table 6: Python libraries utilised for this project.

Package	**Version**	**Description**
PyTorch	1.9.0	Machine learning framework
Scikit-Learn	1.0.1	Machine Learning Library
Deep Graph Library (DGL)	0.7.2	Deep learning on graphs
DGL-LifeSci	0.2.8	Cheminformatics graph functions
RDKit	2021.09.2	Cheminformatics Toolkit
DeepChem	2.6.0.dev	Cheminformatics Machine Learning
Pandas	1.1.5	Data manipulation and preparation
Numpy	1.19.5	Adds support for multi-dimensional arrays
ChemBL Structure Pipeline	1.0.0	Used to standardise molecules
NetworkX	2.6.3	Used to visualise graphs
TQDM	4.59	Progress bars library
SciPy	1.7.1	Statistical Analysis

All the experiments were run on Google Colaboratory[6], Colab in short, and this platform's details are specified in Table 7.

[5]Accessed from: `https://pypi.org/`. Last Accessed: 26/05/2022

[6]Accessed from: `https://colab.research.google.com/`. Last Accessed: 26/05/2022

Table 7: Hardware provisioned in Google Colab.

Type	Model	Details
CPU	Intel (R) Xeon	2.20 Ghz 4 Cores
GPU	Nvidia Tesla P100	16 GB using Cuda 11.1
RAM	N/A	25 GB

Results

This section presents the few-shot model results for the three evaluation datasets, Tox21, MUV, and DUD-E. All three datasets are highly imbalanced, where the inactive/decoy molecules greatly outnumber the number of actives. This imbalance presents a challenging problem but proves that low-data machine learning is highly beneficial in this domain. We first present the work we reproduced from Altae-Tran et al.[3], which we also use to test on a subset of the DUD-E dataset. The reproduced work includes Siamese Networks[4] and the Matching Networks[5] with the IterRefLSTM. The latter obtained the best results in Altae-Tran et al.[3] and is referred to as the state-of-the-art in this study. Further building on the Matching Networks architecture by Vinyals et al.[5], we present and discuss results for two newly proposed machine learning models in this domain. These machine learning models include the Prototypical[6] and Relation[7] Networks. These architectures have been previously explored for the computer vision domain but, to our knowledge, have never been applied to the drug discovery domain. Finally, we evaluate the results with the state-of-the-art, which is identified to be the work of Altae-Tran et al.[3].

Tox21

In line with the results reported by Altae-Tran et al.[3], the few-shot learning models on Tox21 outperform the benchmark models significantly. The Matching Networks with IterRefLSTM performs well and obtain the best ROC-AUC results in some experiments. Our few-shot learning architecture implementation is identical to the work of Altae-Tran et al.[3]; however, variations in how the model learns could be present. Hence, we focus mainly on how our

implementations performed against each other. The results from the Prototypical Networks (PNs) overall significantly outperform the results from the MNs (the state-of-the-art approach) based on statistical analysis (see Table 8). Meanwhile, MNs and PNs, outperform Relation Networks (RNs) in both ROC-AUC and PR-AUC performance.

Results for one-shot learning do not provide a clear-cut choice between our implementations for MNs and PNs with the IterRefLSTM. This performance is expected due to the similarity in the architecture of these methods. In a one-shot learning scenario, MNs and PNs are conceptually similar. The main difference lies in the distance function used since for MNs we use the cosine distance, while for PNs, we use the Euclidean distance, as proposed in the original literature, which introduced these two techniques. 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 and for such a problematic scenario (i.e. learning with only one example from each class), results are promising. The *prototypes* in PNs are a mean of all embeddings for each class in the support set. The Euclidean distance between *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 the MNs. One observation which can be made is that there is not a significant improvement in performance from one-shot learning to 10-shot learning. This uniformity may be attributed to the methodology used for training. Few-shot learning conditions during training must match the ones during testing, but during training, we use a sequence of episodes, in which each we match the few-shot conditions during testing. Having a sequence of episodes means that training sees many molecules, albeit in a few-shot scenario. Hence, the model itself may already be maximally informative due to the number of episodes it is exposed to. Thus, when we get to the testing stage, presenting one-shot or 10-shot support sets to fine-tune the model does not seem to make an impactful difference.

Table 8: Mean ROC-AUC and PR-AUC Scores with standard deviation for ML Models for the Tox21 Test Targets over 20 rounds of testing. The bold text illustrates the best-obtained value. The first column shows the composition of the support set. The reproduced results use a reproduction from the MatchingNet model in Altae-Tran et al.[3]. The actual values are tabulated in the supporting information document in tables S1, S2, and S3.

Tox21	Metric	RF	Graph Conv	SiameseNet	MatchingNet	ProtoNet	RelationNet
10+/10-	ROC-AUC	0.617 ± 0.060	0.620 ± 0.065	0.825 ± 0.043	0.824 ± 0.022	**0.826 ± 0.034**	0.814 ± 0.030
	PR-AUC	0.158 ± 0.102	0.150 ± 0.095	0.226 ± 0.107	0.367 ± 0.105	**0.384 ± 0.105**	0.360 ± 0.102
5+/10-	ROC-AUC	0.602 ± 0.059	0.610 ± 0.062	0.828 ± 0.069	**0.824 ± 0.033**	0.823 ± 0.038	0.822 ± 0.023
	PR-AUC	0.148 ± 0.090	0.152 ± 0.094	0.190 ± 0.094	0.369 ± 0.110	**0.388 ± 0.111**	0.355 ± 0.104
1+/10-	ROC-AUC	0.563 ± 0.068	0.558 ± 0.076	0.836 ± 0.138	0.822 ± 0.025	**0.826 ± 0.032**	0.814 ± 0.028
	PR-AUC	0.128 ± 0.084	0.126 ± 0.075	0.099 ± 0.093	0.301 ± 0.103	**0.384 ± 0.096**	0.325 ± 0.103
1+/5-	ROC-AUC	0.534 ± 0.066	0.559 ± 0.090	0.807 ± 0.159	**0.820 ± 0.033**	**0.820 ± 0.033**	0.819 ± 0.023
	PR-AUC	0.112 ± 0.059	0.128 ± 0.080	0.106 ± 0.086	0.339 ± 0.115	**0.362 ± 0.106**	0.318 ± 0.108
1+/1-	ROC-AUC	0.550 ± 0.061	0.548 ± 0.102	0.818 ± 0.075	0.819 ± 0.036	**0.820 ± 0.030**	0.813 ± 0.029
	PR-AUC	0.118 ± 0.068	0.123 ± 0.082	0.198 ± 0.102	0.352 ± 0.121	**0.373 ± 0.102**	0.342 ± 0.093

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. Our results show that the few-shot learning techniques explored in this study are better suited for lead optimisation (such as toxicity information) rather than hit identification, for which MUV is mostly designed. 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, our tests and statistical analysis show 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-AUC 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 inconsistent, 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. The results for the MUV dataset are shown in Table 9.

Table 9: Mean ROC-AUC and PR-AUC Scores with standard deviation for ML Models for MUV Test Targets over 20 rounds of testing. The bold text illustrates the best-obtained value. The first column shows the composition of the support set. The reproduced results use a reproduction from the MatchingNet model in Altae-Tran et al.[3]. The actual values are tabulated in the supporting information document in tables S4, S5, S6, S7, and S8.

MUV	Metric	RF	Graph Conv	SiameseNet	MatchingNet	ProtoNet	RelationNet
10+/10-	ROC-AUC	**0.728 ± 0.145**	0.713 ± 0.133	0.562 ± 0.046	0.628 ± 0.096	0.599 ± 0.085	0.490 ± 0.071
	PR-AUC	**0.066 ± 0.073**	0.009 ± 0.012	0.001 ± 0.000	0.007 ± 0.010	0.003 ± 0.002	0.002 ± 0.001
5+/10-	ROC-AUC	**0.696 ± 0.132**	0.666 ± 0.115	0.550 ± 0.054	0.516 ± 0.085	0.576 ± 0.055	0.502 ± 0.072
	PR-AUC	**0.071 ± 0.076**	0.015 ± 0.022	0.001 ± 0.001	0.003 ± 0.002	0.003 ± 0.002	0.003 ± 0.002
1+/10-	ROC-AUC	**0.599 ± 0.104**	0.585 ± 0.116	0.648 ± 0.158	0.492 ± 0.082	0.540 ± 0.053	0.547 ± 0.090
	PR-AUC	**0.021 ± 0.032**	0.006 ± 0.008	0.001 ± 0.002	0.002 ± 0.001	0.003 ± 0.002	0.003 ± 0.002
1+/5-	ROC-AUC	**0.587 ± 0.106**	0.585 ± 0.126	0.613 ± 0.179	0.461 ± 0.046	0.494 ± 0.050	0.500 ± 0.000
	PR-AUC	**0.027 ± 0.040**	0.006 ± 0.008	0.001 ± 0.002	0.002 ± 0.001	0.002 ± 0.001	0.002 ± 0.000
1+/1-	ROC-AUC	0.573 ± 0.103	**0.577 ± 0.147**	0.620 ± 0.138	0.507 ± 0.037	0.505 ± 0.030	0.484 ± 0.060
	PR-AUC	**0.022 ± 0.036**	0.006 ± 0.007	0.004 ± 0.011	0.002 ± 0.000	0.003 ± 0.001	0.002 ± 0.001

GPCR subset of the DUD-E

The few-shot learning model trained on the ADRB2 target achieves stellar performance based on ROC-AUC and PR-AUC scores. The results are close to a perfect classifier, which raises concerns about the underlying data. We hypothesise 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.[31,32] Chen et al.[33] 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 between the actives of different targets (inter-target analogue bias). They also provide evidence of 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 decoys available for the CXCR4 target, the RF model excels and outperforms the few-shot learning models. The GCN benchmark model also performed significantly better than few-shot learning models, which implies 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 dataset. The results for the GPCR subset of DUD-E are presented in Table 10.

Table 10: Mean ROC-AUC and PR-AUC Scores with standard deviation for ML Models for DUD-E GPCR Test Targets over 20 rounds of testing. The bold text illustrates the best-obtained value. The first column shows the composition of the support set. The actual values are tabulated in the supporting information document in tables S9, and S10.

DUDE-GPCR	Metric	RF	Graph Conv	SiameseNet	MatchingNet	ProtoNet	RelationNet
10+/10-	ROC-AUC	**0.982 ± 0.018**	0.940 ± 0.039	0.784 ± 0.215	0.900 ± 0.102	0.816 ± 0.187	0.928 ± 0.008
	PR-AUC	**0.872 ± 0.102**	0.504 ± 0.225	0.489 ± 0.475	0.535 ± 0.451	0.552 ± 0.445	0.562 ± 0.020
5+/10-	ROC-AUC	**0.958 ± 0.023**	0.901 ± 0.058	0.761 ± 0.238	0.845 ± 0.153	0.843 ± 0.181	0.850 ± 0.149
	PR-AUC	**0.762 ± 0.119**	0.428 ± 0.247	0.495 ± 0.465	0.506 ± 0.477	0.559 ± 0.439	0.523 ± 0.447
1+/10-	ROC-AUC	0.854 ± 0.071	0.788 ± 0.098	0.759 ± 0.247	**0.881 ± 0.119**	0.841 ± 0.159	0.866 ± 0.132
	PR-AUC	0.360 ± 0.136	0.230 ± 0.176	0.474 ± 0.445	0.521 ± 0.455	0.504 ± 0.463	**0.541 ± 0.433**
1+/5-	ROC-AUC	**0.858 ± 0.084**	0.763 ± 0.087	0.759 ± 0.246	0.851 ± 0.155	0.793 ± 0.211	0.848 ± 0.149
	PR-AUC	0.378 ± 0.123	0.221 ± 0.153	0.482 ± 0.444	0.516 ± 0.438	**0.519 ± 0.474**	0.490 ± 0.427
1+/1-	ROC-AUC	0.804 ± 0.108	0.710 ± 0.121	0.771 ± 0.228	0.795 ± 0.203	**0.865 ± 0.133**	0.747 ± 0.251
	PR-AUC	0.301 ± 0.168	0.116 ± 0.121	0.500 ± 0.417	0.511 ± 0.470	**0.543 ± 0.439**	0.500 ± 0.465

Comparison with the State of the Art

We tabulate the ROC-AUC results obtained by Altae-Tran et al.[3] in Table 11 and compare them to the best results obtained from our implementations. The best network is selected using statistical analysis, comparing the results from 20 test rounds for each experiment across all the techniques employed. Instances in which we have more than one best network tabulated, such as the SR-MMP 10+/10- example in Table 11, indicate that we do not find any statistically significant difference between the results obtained from that specific technique. The PN architecture has the highest frequency of being identified as the best network on Tox21 data based on ROC-AUC scores. We remind the reader that while we also report the PR-AUC score from our experiments, this metric is not available in the study by Altae-Tran et al.[3], hence why the results reported in Table 11 contain only ROC-AUC results. We highlight that Prototypical Networks consistently performed well based on PR-AUC metrics, obtaining the best scores throughout all Tox21 targets. Using statistical analysis, Matching Networks and Relation Networks also match the performance in some cases. The PR-AUC is used to determine how well the model predicts active compounds,

as it is the ratio of true positives divided by the sum of true positives and false positives. Therefore, we firmly believe that in machine learning experiments for virtual screening, this metric should be used in addition to ROC-AUC scores. We attribute any improvement in ROC-AUC for Matching Networks in our implementation over the state-of-the-art to the featurisation of molecules and variability which might arise through machine learning. However, we reiterate that all results reported in this study are compared homogeneously using our implementations of all machine learning architectures.

Table 11: Comparison of our best ROC-AUC scores against the state-of-the-art (SOTA) results from Altae-Tran et al.[3] on the Tox21 dataset. Values are mean values with standard deviation over 20 rounds of testing. Best values are highlighted in bold text.

Target	Support Set	SOTA	SOTA ROC-AUC	Obtained ROC-AUC	Best Networks
SR-HSE	10+/10-	MN	0.772 ± 0.002	**0.793 ± 0.002**	MN
SR-HSE	5+/10-	MN	0.771 ± 0.002	**0.791 ± 0.003**	RN
SR-HSE	1+/10-	MN	0.671 ± 0.007	**0.788 ± 0.001**	MN
SR-HSE	1+/5-	MN	0.729 ± 0.003	**0.789 ± 0.001**	RN
SR-HSE	1+/1-	MN	0.767 ± 0.001	**0.779 ± 0.007**	PN
SR-MMP	10+/10-	MN	0.838 ± 0.001	**0.845 ± 0.015**	MN/PN/RN
SR-MMP	5+/10-	MN	0.847 ± 0.001	**0.853 ± 0.007**	MN/PN
SR-MMP	1+/10-	SN	0.809 ± 0.020	**0.849 ± 0.005**	PN
SR-MMP	1+/5-	MN	0.799 ± 0.002	**0.853 ± 0.001**	MN
SR-MMP	1+/1-	MN	0.835 ± 0.001	**0.851 ± 0.008**	MN
SR-p53	10+/10-	MN	0.823 ± 0.002	**0.850 ± 0.004**	PN
SR-p53	5+/10-	MN	0.830 ± 0.001	**0.852 ± 0.009**	PN
SR-p53	1+/10-	SN	0.726 ± 0.173	**0.848 ± 0.005**	PN
SR-p53	1+/5-	MN	0.795 ± 0.005	**0.840 ± 0.005**	PN
SR-p53	1+/1-	MN	0.827 ± 0.001	**0.838 ± 0.004**	MN

ECFP vs Graph-Learned Embeddings

We also tested whether the molecular representation affects the performance in few-shot learning on Tox21 data. We only employ the Prototypical Networks architecture for this particular experiment, as these performed consistently well in our other experiments. ECFPs are based on the topology and atom descriptors, whereby 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 embedding in latent space 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. The values obtained from these experiments are tabulated in the supporting information document in Table S11.

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 obtained the most consistent results. As the decrease in training times is substantial, by over 150% between MNs and both PNs and RNs, we believe this puts the latter two networks at an advantage. Faster training times allow a faster turnaround of results 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 explored if a machine learning model can *learn how to learn* and generalise using only a few examples in the virtual screening domain. This study builds on the work from Altae-Tran et al.[3], who have set essential foundations for this domain. We reproduce their work effectively and provide deeper insights into the study by introducing PR-AUC reporting, in addition to the reported ROC-AUC scores in their study, to increase robustness against highly imbalanced data. We also introduce Prototypical Networks and Relation Networks, two new few-shot machine learning models, to this domain and compare results

to the state-of-the-art.

While performance varies across the datasets, this difference is consistent with the reported results from Altae-Tran et al.[3]. The Prototypical Networks outperform all other machine learning models, including the state-of-the-art model, based on ROC-AUC and PR-AUC performance on the Tox21 dataset. Additionally, given the highly imbalanced nature of the data, our PR-AUC results provide more robust insights into the performance of the models. The state-of-the-art and Prototypical Networks perform significantly better than our implementation of Relation Networks. We also observe that Prototypical Networks achieve this improved performance with much faster training times than our implementation of the state-of-the-art.

Due to the nature of the data used, where active compounds per target are highly scarce and each compound is structurally distinct, MUV data does not provide enough information for the machine learning model to generalise effectively for few-shot learning. Results on the DUD-E GPCR subset are also inconclusive, and for these datasets, our baseline experiments using conventional machine learning techniques perform better. We conclude that Prototypical Networks offer better generalising capabilities for few-shot learning in ligand-based virtual screening, specifically for lead optimisation, than the Matching Networks component used in the state-of-the-art. However, this is dependent on the nature of the data used. Finally, we also observe that using learned embeddings through GCNs, as opposed to ECFPs, consistently results in better ROC-AUC and PR-AUC performances.

Acknowledgement

This work is partially supported by the '*Discovery of COVID-19 Inhibitors*' (DisCO) project. Project DisCO is financed by the Malta Council for Science & Technology, for and on behalf of the Foundation for Science and Technology, through the Infectious Diseases Programme (grant agreement number: IDP.RD.2021-05).

Supporting Information Available

We provide the ROC-AUC and PR-AUC results for each individual target used for testing in supporting information (`supporting_information.pdf`).

Data and Software Availability

All the data used for the validation of this study (Tox21, DUD-E, and MUV) is publicly available. The models are implemented in the Python programming language using Jupyter Notebooks which are run directly in Google Colab. The raw data and code is freely available at `https://github.com/danielvlla/Few-Shot-Learning-for-Low-Data-Drug-Discovery`, and is released under an MIT licence.

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) 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.

(10) Rogers, D.; Hahn, M. Extended-connectivity fingerprints. *Journal of chemical information and modeling* **2010**, *50*, 742–754.

(11) 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**,

(12) David, L.; Thakkar, A.; Mercado, R.; Engkvist, O. Molecular representations in AI-driven drug discovery: a review and practical guide. *Journal of Cheminformatics* **2020**, *12*, 1–22.

(13) 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.

(14) 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.

(15) 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.

(16) Bronstein, M. M.; Bruna, J.; Cohen, T.; Veličković, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478* **2021**,

(17) LeCun, Y.; Bengio, Y., et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks* **1995**, *3361*, 1995.

(18) Kuhn, M.; Letunic, I.; Jensen, L. J.; Bork, P. The SIDER database of drugs and side effects. *Nucleic acids research* **2016**, *44*, D1075–D1079.

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

(21) Ramsundar, B.; Eastman, P.; Walters, P.; Pande, V. *Deep learning for the life sciences: applying deep learning to genomics, microscopy, drug discovery, and more*; " O'Reilly Media, Inc.", 2019.

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

(23) 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.

(24) 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.

(25) RDKit, Open-source cheminformatics. https://www.rdkit.org. 2012; Last Accessed on 25/11/2021.

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

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

(28) 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**,

(29) 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**,

(30) Mann, H. B.; Whitney, D. R. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics* **1947**, 50–60.

(31) 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.

(32) Wallach, I.; Heifets, A. Most ligand-based classification benchmarks reward memoriza-

tion rather than generalization. *Journal of chemical information and modeling* **2018**, *58*, 916–932.

(33) 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.

For Table of Contents Only