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 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 lead optimisation and hit identification, 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 as inputs to neural networks for classification. We find that learned graph embeddings 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, which is based on the Matching Networks architecture. 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. 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 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 identification and development of 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 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] 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 the learning of rare cases possible.[8]

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

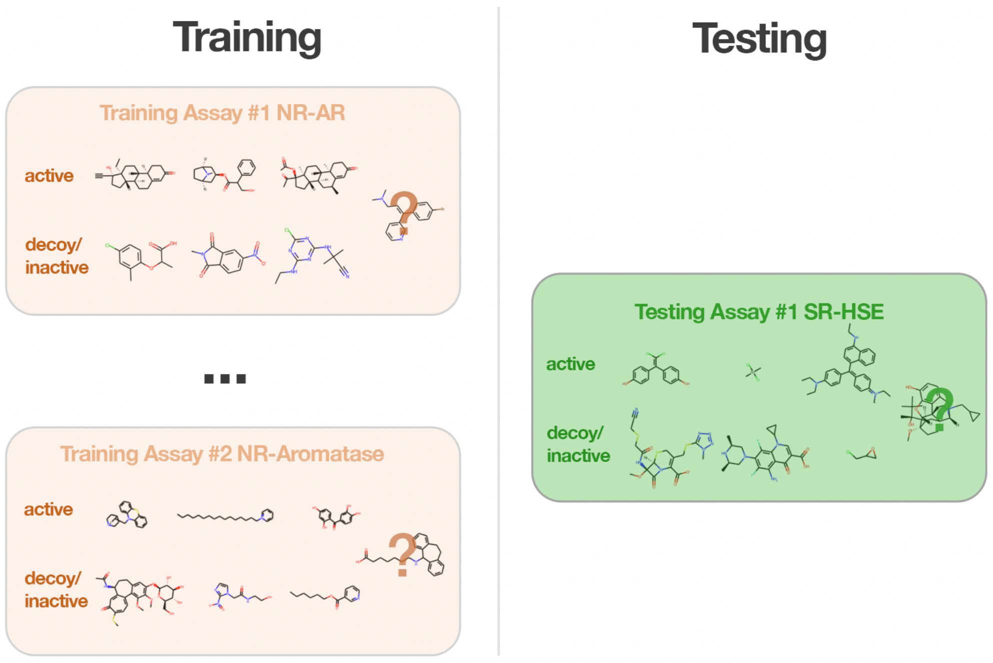


Figure 1: 2-way 3-shot few-shot classification. Training a meta-learner on a set of experimental assays, and generalising for an unseen assay in the Tox-21 dataset.

lem for hit identification and lead optimisation. The ability for 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 environments that were previously unseen during training time. Few-shot classification, a meta-learning paradigm, we train models using a variety of training tasks and optimise for 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. The *way* refers to the number of classes we have per task and the *shot* component refers to the number of samples. 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[9], where data from a set of assays reserved for training are used to train a model, which is later used to generalise for a new, unseen assay using only a support set from this new assay. We highlight that few-shot learning in the hit identification and lead optimisation is different 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, a class unseen during training, the model must generalise for new images of a lion. In the domain under study, the challenge is to train a model that is able to 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 this new experimental assay. 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, and 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 holds no specific information on the molecule's topology. In fact, this particular formula can refer to alanine, sarcosine, and lactamide. 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, and can be used as inputs to machine learning networks. In this study, we mainly explore the use of graphs as embeddings for the low-data machine learning networks. A graph is formally defined as 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 neural networks, which could be of benefit over topological molecular representations such as ECFP.[13]

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 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, referred to as one-shot learning,[4,5] to a maximum of ten examples per class.

Related Work

Several successful research undertakings have exploited the low-data learning paradigm, especially in the computer-vision domain.[4–7] Being able to learn from only a few examples is especially important in domains that suffer from a paucity of data. This inaccessibility could be attributed to privacy, safety, or ethical issues, in addition to 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 the Iterative Refinement of embeddings using Long-Short Term Memory (LSTM) networks. In our research, we build on the work by[3] and extend it through the application of other successful few-shot learning approaches, previously explored for other domains such as the computer-vision domain. The authors employ Graph Neural Networks (GNN) to learn molecular embeddings, which are then fed into the low-data architectures for classification.

Graph Neural 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 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, amongst others, can be encoded in the node feature vector, in addition to bond information. However, the latter is omitted for the purpose of this study.

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

Graph neural networks may be used to learn molecular representations.[14] Embeddings learned through neural networks afford the construction of automated features, rather than fixed fingerprints. Graph neural networks 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.

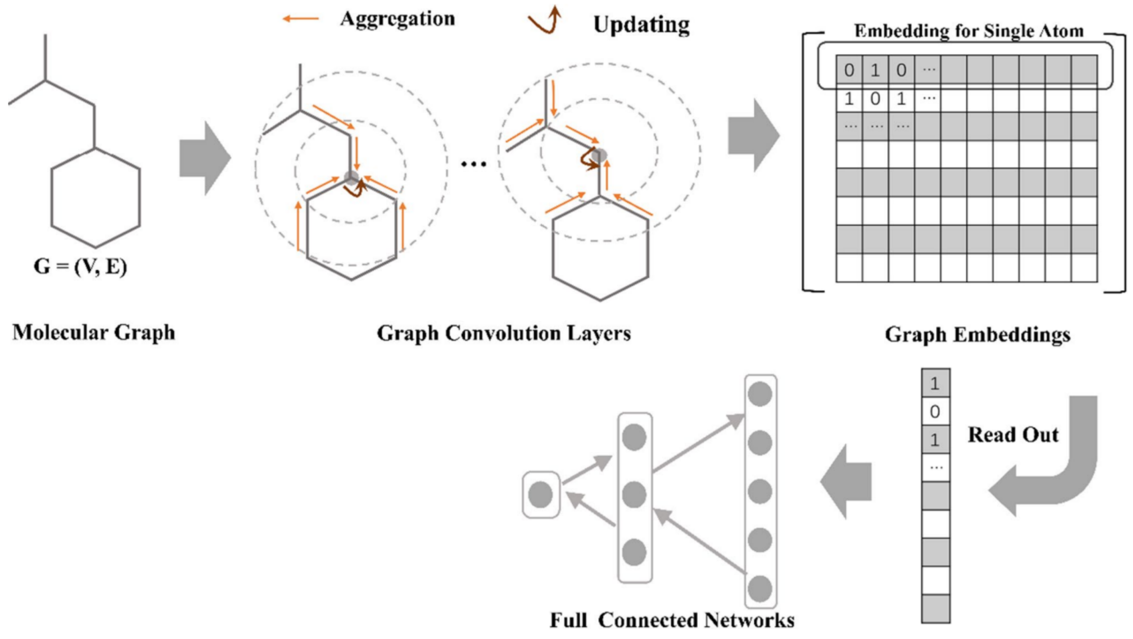


Figure 2: A typical pipeline for representing molecules using a learned embedding function, which can be processed further using feed-forward neural networks as shown. Reproduced from Jiang et al.[14].

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 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[4,17] 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.

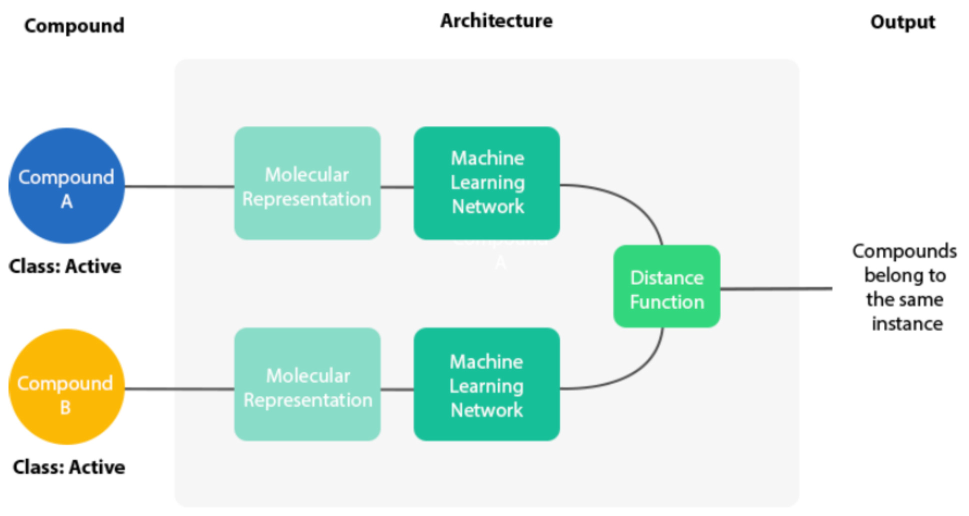


Figure 3: High level schematic of a Siamese network for molecular network.

Matching Networks

Matching Networks[5] build 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. 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.

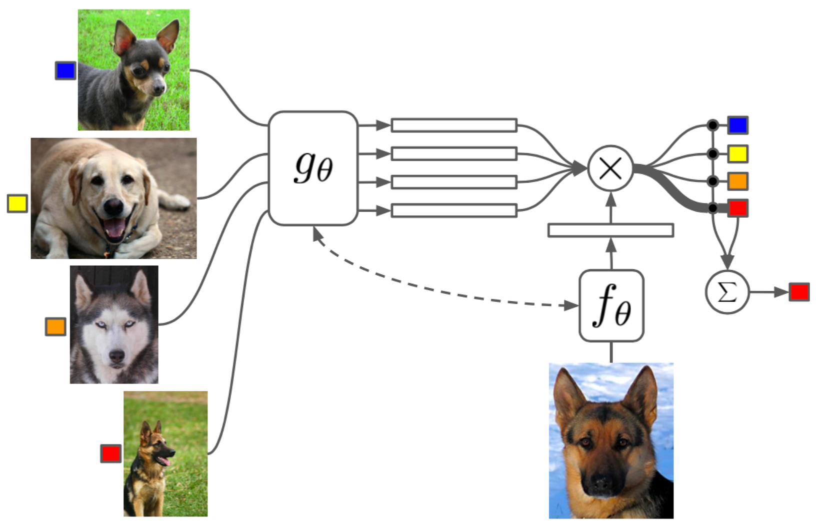


Figure 4: Matching Networks Architecture. Reproduced from Vinyals et al.[5].

Figure 4 illustrates the Matching Nets architecture. Embedding functions f and g are Convolutional Neural Networks (CNNs)[18], potentially being identical to each other, which 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. The attention mechanism a, at the end of the pipeline, is the classifier which takes a softmax over the cosine distance of the embeddings.

Prototypical Networks

Prototypical Networks, proposed by Snell et al.[6], are similar to Matching Networks. Instead of comparing the query support to each 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 related to each class, thus creating the *prototypes*. The Euclidean distance between the query data point and the prototypes is calculated to classify the query (see Figure 5). 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.

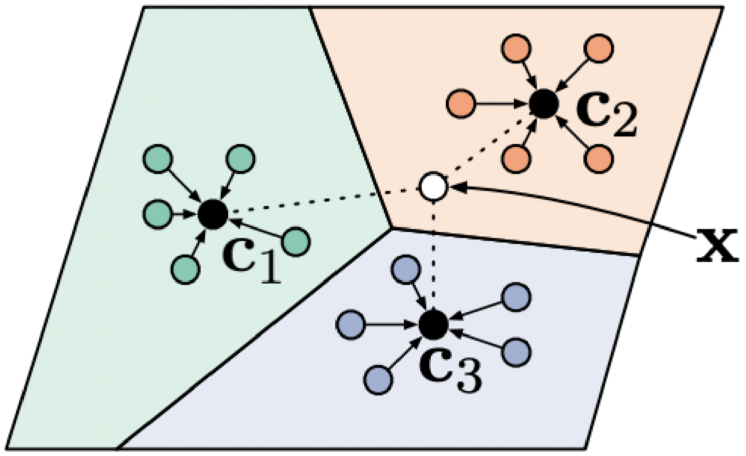


Figure 5: Few-shot learning in Prototypical Networks, where prototypes c_k are taken as the mean of embedded support examples for each class. Reproduced from Snell et al.[6].

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. 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 (see Figure 6).

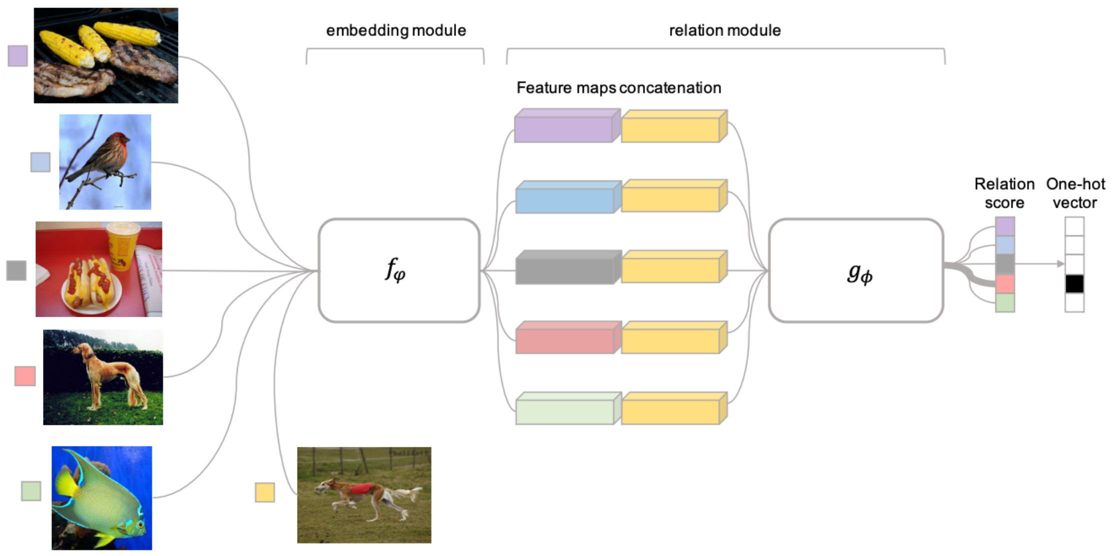


Figure 6: Few-shot learning scenario in Relation Networks for a 5-way 1-shot learning task with one query as an example. Reproduced from Sung et al.[7].

Iterative Refinement LSTM

Altae-Tran et al.[3] build on meta-learning concepts, where they train 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 the new assay. 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. In their work, the support and query molecules are embedded 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 simply 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 embedding of the query is built iteratively with that of the support set, using information from the two sets to enhance both the support and query embeddings (see Figure **??**). 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 make use of the Cosine distance to compare embeddings. Figure 7 illustrates a one-shot learning scenario encapsulating the aforementioned concepts.

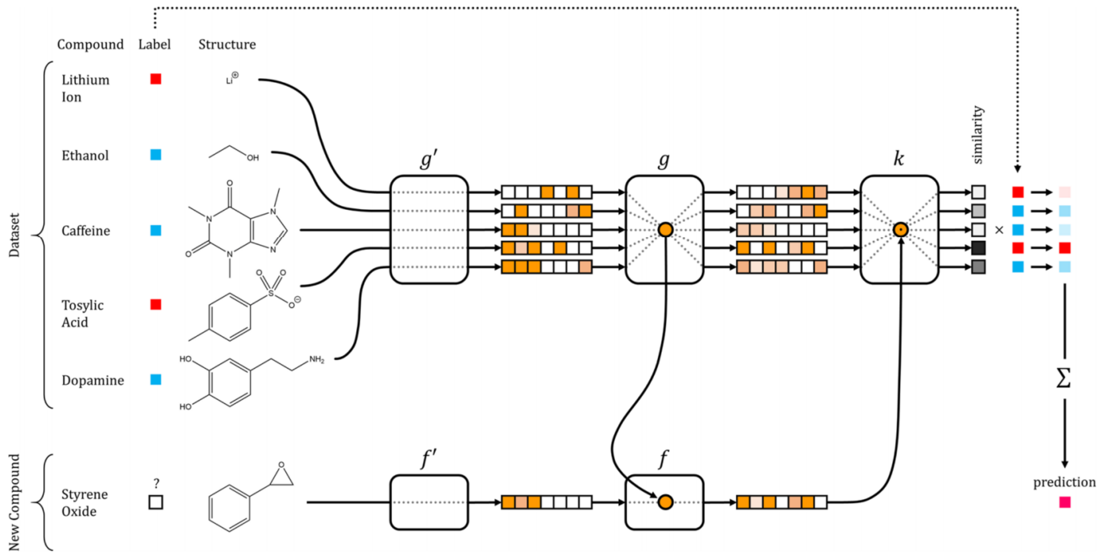


Figure 7: Schematic of one-shot learning in drug discovery based on the Matching Network[5] architecture. Reproduced from Altae-Tran et al.[3].

Their work is evaluation on the Tox21, the Side Effect Resource (SIDER)[19], and MUV datasets[20]. 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 make use of a Random Forest

(RF) with 100 decision trees as a machine learning baseline model. They also utilise a conventional Graph Convolutional Networks (GCN)[21] as an additional baseline model, which is trained using only a small support set from the test targets. They then experiment with Siamese 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, 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 into how the model performs when predicting the *positive* (active) class in the data. Predicting the "active" class correctly 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 out-performed few-shot learning. The authors report that this is due to MUV data being maximally informative, and therefore structural similarity cannot be utilised to generalise for activity prediction. The authors open-sourced their models in the DeepChem library[22]. However, the implementations are now outdated and not executable with the more recent versions of the DeepChem library, which makes reproduction of results difficult. However, we study the open-sourced implementation along with the implementation details in the original literature to successfully reproduce this work.

Methodology

In this work, we implement a Random Forest and a Graph Convolutional Network (GCN) to use as benchmark models. Additionally, we implement four few-shot machine learning architectures, namely, Siamese Networks, Matching Networks, Prototypical Networks, and Relation Networks. The state of the art, IterRefLSTMs[3], are used to enrich the resulting embeddings in latent space. Molecules are represented as graph objects, which are then processed using GCNs to produce a vectorised embedding in computational space. Throughout our work, we try to follow the implementation of Altae-Tran et al.[3] as closely as possible for the sake of reproducibility and homogenising the results for 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 data is provided 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 in order 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 we have so far 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, support sets and queries are randomly sampled to form a series of episodes for training. The support sets are composed of a number of examples per class. The number of examples per class ranges from just one example, up to ten examples.

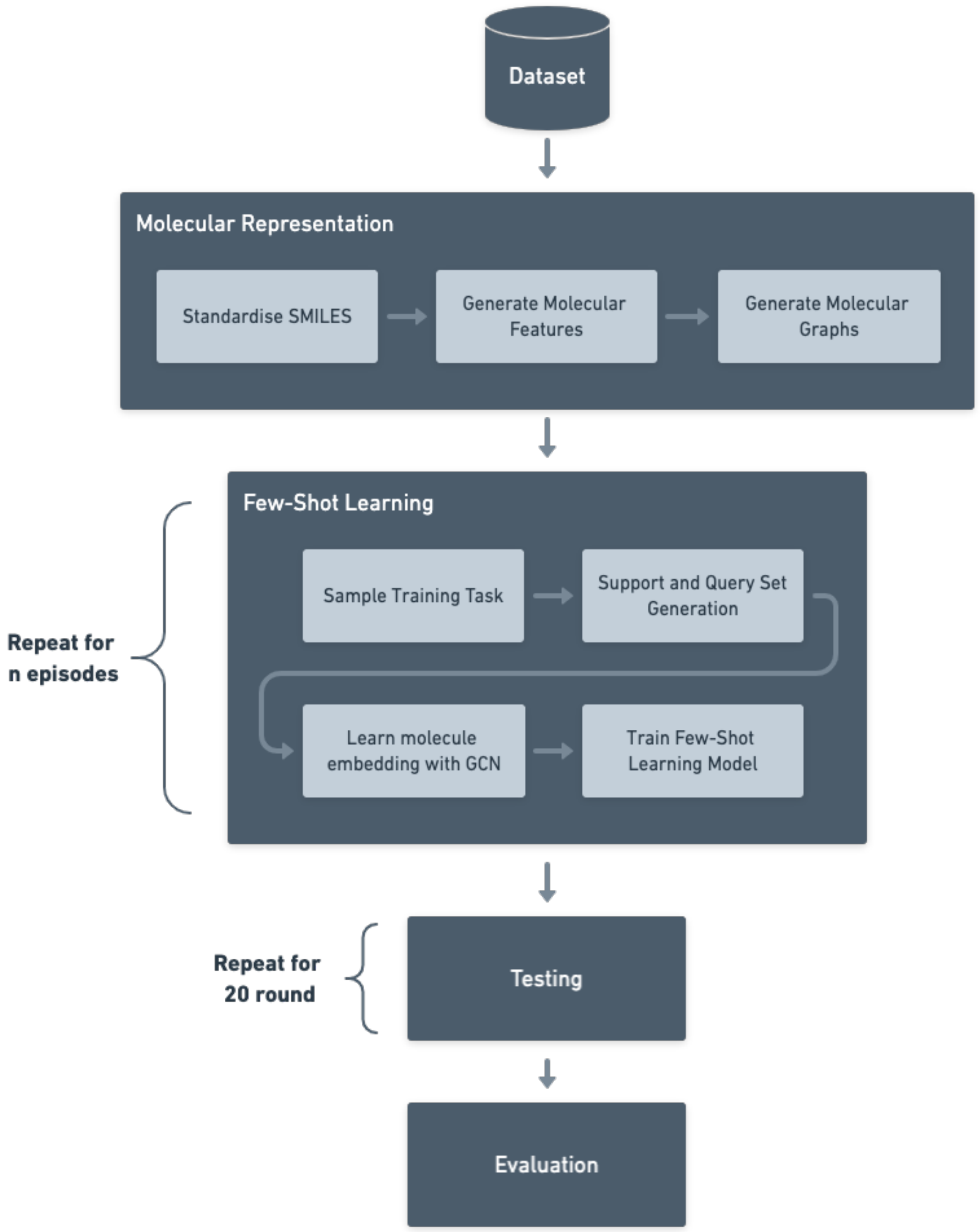
6. **Learning a molecular embedding**. The sampled molecules in the episode are used to learn a molecular embedding using a graph convolutional network.

7. **Few-Shot Learning**. The learned embeddings are processed using four different meta-learning architectures namely Siamese, Matching, Prototypical and Relation Networks. Iterative Refinement LSTMs[3] are used to enrich the few-shot learning. 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, 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, for which the mean and standard deviation of the areas under the curve for Precision-Recall Curve and Receiver Operator Characteristic curve are calculated to quantify performance.

9. **Evaluation**. Finally, we evaluate the results based on the ROC and Precision Recall Curves (PRC) 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 and PRC graphs for the experiment with the median ROC score from the 20 rounds are generated after test

completion.



Dataset
Molecular Representation
Standardise SMILES
Generate Molecular Features
Generate Molecular Graphs
Few-Shot Learning
Sample Training Task
Support and Query Set Generation
Learn molecule embedding with GCN
Train Few-Shot Learning Model
Repeat for n episodes
Repeat for 20 round
Testing
Evaluation

Figure 8: Schematic of the machine learning pipeline designed for this study. The changes across few-shot learning architectures lies in the 'Train Few-Shot Learning Model' component, otherwise, all other modules remain identical.

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[23]. 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)**[20] – Based on PubChem BioAssays, used for validating virtual screening techniques against 17 different targets[20]. 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)**[24] – 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[24]. 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.

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 select appropriate performance metrics (see Evaluation Section).

[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

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

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 a set of well-defined and consistent rules and conventions. It is of utmost importance to maintain uniformity and integrity across the different datasets (and sources) being used. Bento et al.[25] present an open source chemical structure curation pipeline based on RDKit[26] for validating and standardising chemical structures, which follow FDA/IUPAC guidelines[27,28]. Their work is available in the ChEMBL Structure Pipeline package[25] and is used to standardise the molecules in our pipeline.

We create a molecular graph from the SMILES string of the standardized molecule using RDKit, an open-source toolkit for cheminformatics. We then one-hot encode eight 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 the Deep Graph Library (DGL) LifeSci[29] library to create the graph objects and subsequently process them using the DGL library.[30]

Episodic Learning

Figure 9 illustrates a high-level overview of the episodic learning methodology. Training for few-shot learning is carried out in a series of episodes, framed as *N-way K-shot* classification tasks. We consider *N-way K-shot* classification tasks, where the support set contains N

classes and K labelled molecules. These classification tasks match the conditions at training time with those during testing, as proposed by Vinyals et al.[5]. 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 directly compare results with this study.

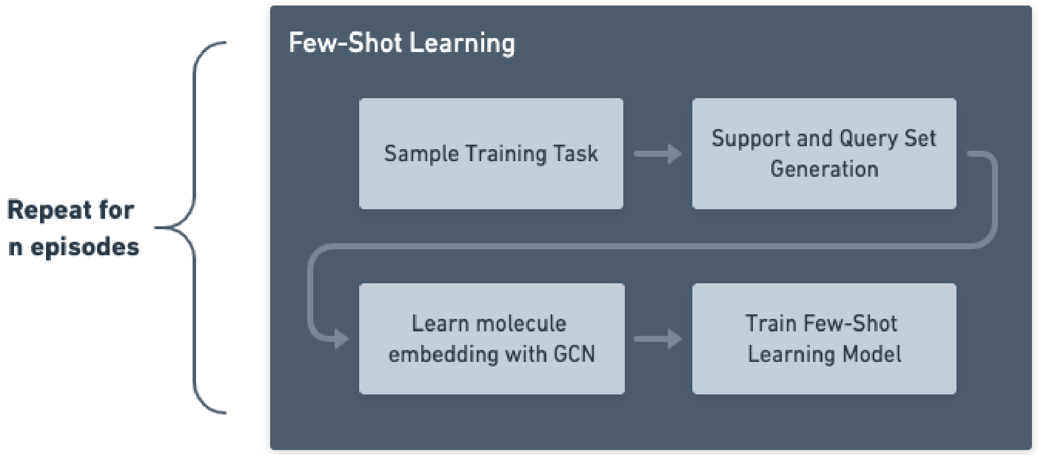

Repeat for n episodes
Few-Shot Learning
Sample Training Task
Support and Query Set Generation
Learn molecule embedding with GCN
Train Few-Shot Learning Model

Figure 9: Episodic learning schematic

Table 2 contains the composition of the support sets used in our experiments. For each episode, we sample a total of 128 query molecules, 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. 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 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.

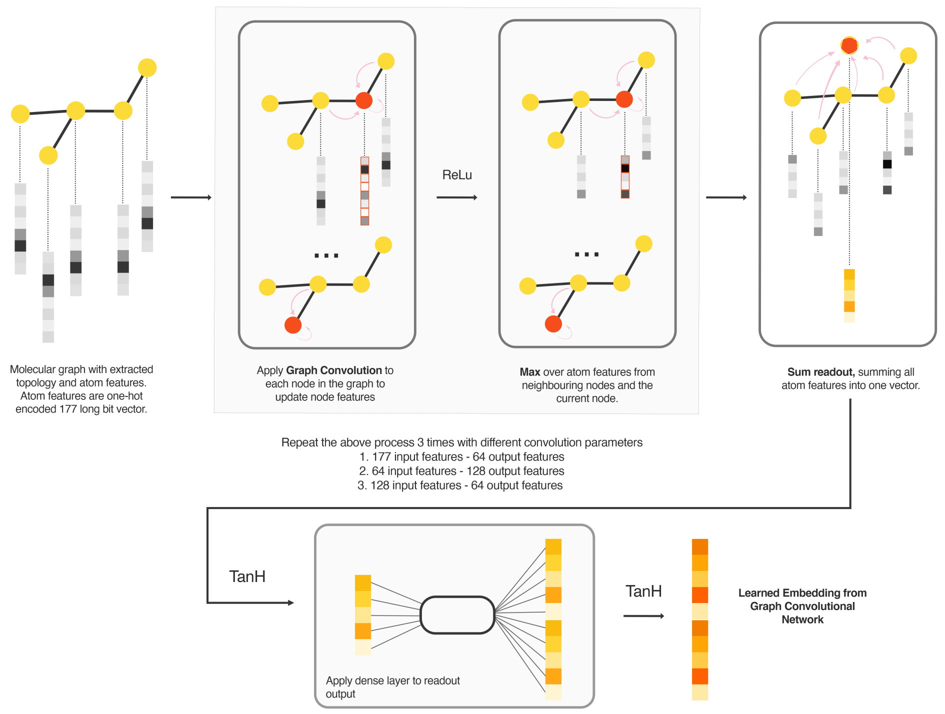

Molecular graph with extracted topology and atom features. Atom features are one-hot encoded 177 long bit vector.
Apply **Graph Convolution** to each node in the graph to update node features
ReLu
Max over atom features from neighbouring nodes and the current node.
Sum readout, summing all atom features into one vector.
Repeat the above process 3 times with different convolution parameters
1. 177 input features - 64 output features
2. 64 input features - 128 output features
3. 128 input features - 64 output features
TanH
Apply dense layer to readout output
TanH
Learned Embedding from Graph Convolutional Network

Figure 10: 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. Figure 10 illustrates the GCN pipeline to learn a molecular embedding. In our study, we make use of the convolutional operator from Kipf and Welling[21] to process graphs and learn the molecular embeddings.

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

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 $h_u = f(W_u \hat{h}_u)$[21].

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 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, for which we use 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 11.

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

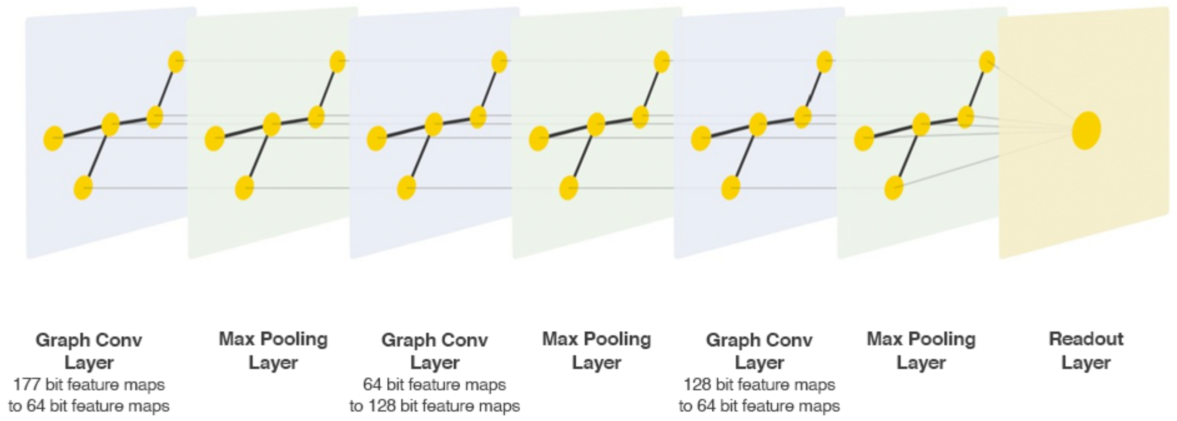

Graph Conv Layer
177 bit feature maps to 64 bit feature maps
Max Pooling Layer
Graph Conv Layer
64 bit feature maps to 128 bit feature maps
Max Pooling Layer
Graph Conv Layer
128 bit feature maps to 64 bit feature maps
Max Pooling Layer
Readout Layer

Figure 11: Graph processing layers in our GCN implementation. The graph convolution layers apply operations on each individual node's feature maps based on neighbouring nodes. The ReLU function is applied after each convolutional layer, and the tanh function is applied after the final readout layer.

Benchmark Models

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 2,048 bits are used 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, 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 binary target $(0, 1)$. This binary target signifies whether the molecule belongs to the active or inactive/decoy class in the experimental assay. 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.

We also carry out a final benchmark test in retrospect by taking a random selection of query molecules from a test target, generating the ECFP with the same aforementioned parameters 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 passed to the few-shot learning architectures. 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 using a metric or distance function. The models discussed in this section, excluding the benchmark model, use the embeddings generated from the GCN, presented in the support and query sets, to learn the kernel function.

To be able to compare the model's effectiveness objectively, the GCN architecture remains unchanged in all experiments, including the benchmark.

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 outputs from the networks are compared to learn the relationship between them. The following is the process employed for learning a classifier using Siamese Networks. The process is repeated for all training tasks.

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 1, and 0 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. This is achieved by calculating the absolute difference between the embeddings $|mol_i - mol_j|$.

4. The distance between the two embeddings is passed through a linear feedforward layer with 128 nodes and an output of 2, followed by a sigmoid function to output the probability that the two molecules belong to the same class.

5. As we are dealing with binary classification, the binary cross-entropy loss is calculated and backpropagated to the network.

Matching Networks

Matching Networks extends 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. 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 of pairs of data points between the support and query sets is computed, which is then normalised by a softmax function. 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 of embedding the molecules effectively into latent space. Therefore, a bidirectional long-short term memory (LSTM) is used, taking as input the whole support set to adjust the embedding based on the other support samples.

Two functions are defined. $g_\theta(x_i, S)$ 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 $f_\theta(x, S)$. Throughout each iteration, the hidden state and the output from the attention function are added together. To train the network using stochastic gradient descent, the cross-entropy loss is computed for each query prediction.

Prototypical Networks

Prototypical Networks[6] have similarities to the Matching Networks described above, but instead of considering the individual support set embeddings, the mean vector of the embeddings for each class within the support set is taken. This mean vector for each class is referred to as the *prototype*. Another improvement the authors make over Matching Networks, is the use of Euclidean rather than the cosine distance. In order to classify query data samples, the softmax of the inverse of the euclidean distances between each query and each prototype is taken. To train the network through stochastic gradient descent, the negative log-likelihood loss is used.

Relation Networks

For the Relation Networks[7], the classification of query samples is not done directly in latent space from the embeddings. The embeddings for the support and query samples are generated using the GCN. Following this step, the feature maps concatenations are created by concatenating the query samples with each data sample within the support set. The feature map concatenation therefore is double the length of the embedding generated by the GCN.

The relationship between the queries and the different classes within the support set is captured by passing these feature map concatenations through a feed forward neural network $g_\theta([x_i, x_j])$ to predict a relation score. $[\cdot, \cdot]$ is the concatenation between each support set data sample x_i and the query data samples x_j. The architecture of this function is defined in Table 5. The loss function used is Mean Squared Error (MSE) loss as proposed in the original paper.

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

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]. As emphasised by Vinyals et al.[5] and Snell et al.[6], training and testing conditions should match when doing few-shot learning. Therefore, the same support set 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 used are the Area Under Curve (AUC) of the Receiver Operating Characteristic (ROC) curve, and the Precision-Recall Curve (PRC). To determine the predictive power of our classifier, we make use of the ROC Area Under Curve (AUC) (ROC-AUC) as this provides a clearer picture of the relationship between the true positive and the false positive rate. The ROC-AUC affords a more nuanced approach than accuracy as it provides visibility into thresholds one can utilise to ameliorate predictions. Altae-Tran et al.[3] only report ROC-AUC results, however, this metric alone does not adequately measure the nature of the performance of the machine learning models due to the highly 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

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

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 on 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 those of 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[31].

Implementation Details

These 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 conveniently install and run packages and their dependencies. 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. Colab is a hosted

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

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

Jupyter notebook[7] service, providing access to computational resources including CPUs and GPUs to run Python code. These Colab notebooks can be shared, accessed, and run in web-browsers. The Colab instance details are specified in Table 7.

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

In this section we present the few-shot model results for the three evaluation datasets, Tox21, MUV, and DUD-E. All these datasets are highly imbalanced, where the inactive/decoys greatly outnumber the number of actives. This defining feature of these datasets presents a challenging problem, but is also further evidence 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 test on a subset of the DUD-E dataset. This was not explored in the original study. The reproduced work includes Siamese Networks[4] and the Matching Networks[5] with the Iterative Refinement LSTM (IterRefLSTM). The latter obtained the best results in Altae-Tran et al.[3]. This is followed by the presentation and discussion of the results for two newly proposed machine learning models in this domain, which are based on work of Vinyals et al.[5] for Matching Networks. These machine learning models include the Prototypical[6] and Relation[7] Networks. Finally, we evaluate the results with the state of the art, which is identified to be the work of Altae-Tran et al.[3].

[7]Accessed from: `https://jupyter.org/`. Last Accessed: 26/05/2022

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 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 presented earlier. 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 the performance of how our implementations performed against each other. The results from the Prototypical Networks (PNs) overall outperform the results from the MNs based on statistical analysis (see Table 8). 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, which is expected due to 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 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. 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 and for such a difficult 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 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 the MNs.

Table 8: Mean ROC and PRC 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. The reproduced results from the model used in Altae-Tran et al.[3] is the MatchingNet.

Tox21	**Metric**	**RF**	**Graph Conv**	**SiameseNet**	**MatchingNet**	**ProtoNet**	**RelationNet**
10+/10-	ROC	0.617 ± 0.060	0.620 ± 0.065	0.825 ± 0.043	0.824 ± 0.022	**0.826 ± 0.034**	0.814 ± 0.030
	PRC	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	0.602 ± 0.059	0.610 ± 0.062	0.828 ± 0.069	**0.824 ± 0.033**	0.823 ± 0.038	0.822 ± 0.023
	PRC	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	0.563 ± 0.068	0.558 ± 0.076	0.836 ± 0.138	0.822 ± 0.025	**0.826 ± 0.032**	0.814 ± 0.028
	PRC	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	0.534 ± 0.066	0.559 ± 0.090	0.807 ± 0.159	**0.820 ± 0.033**	**0.820 ± 0.033**	0.819 ± 0.023
	PRC	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	0.550 ± 0.061	0.548 ± 0.102	0.818 ± 0.075	0.819 ± 0.036	**0.820 ± 0.030**	0.813 ± 0.029
	PRC	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. 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. The results for the MUV dataset are shown in Table 9.

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

Table 9: Mean ROC and PRC 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. The reproduced results from the model used in Altae-Tran et al.[3] is the MatchingNet.

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

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.[32,33] Chen et al.[34] 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 decoys available 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 dataset. The results for the GPCR subset of DUD-E are shown in Table 10.

Table 10: Mean ROC and PRC 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	ProtoNet	RelationNet
10+/10-	ROC	**0.982 ± 0.018**	0.940 ± 0.039	0.784 ± 0.215	0.900 ± 0.102	0.816 ± 0.187	0.928 ± 0.008
	PRC	**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	**0.958 ± 0.023**	0.901 ± 0.058	0.761 ± 0.238	0.845 ± 0.153	0.843 ± 0.181	0.850 ± 0.149
	PRC	**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	0.854 ± 0.071	0.788 ± 0.098	0.759 ± 0.247	**0.881 ± 0.119**	0.841 ± 0.159	0.866 ± 0.132
	PRC	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	**0.858 ± 0.084**	0.763 ± 0.087	0.759 ± 0.246	0.851 ± 0.155	0.793 ± 0.211	0.848 ± 0.149
	PRC	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	0.804 ± 0.108	0.710 ± 0.121	0.771 ± 0.228	0.795 ± 0.203	**0.865 ± 0.133**	0.747 ± 0.251
	PRC	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. Prototypical Network 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 for the PRC metrics, Prototypical Networks consistently performed well, obtaining the best PRC scores throughout all Tox21 targets. Using statistical analysis, Matching Networks and Relation Networks also match the performance in some cases. The PRC 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 strongly 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 molecule and variability which might arise through machine learning. However, we reiterate that all results reported in this study are compared homogeneously using the same 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**	**Obtained ROC**	**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 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 explored how a machine learning model can *learn how to learn* and generalise using only a few examples. This study builds on the work from Altae-Tran et al.[3], who have set important foundations for this problem domain.

First, we reproduce their work effectively and provide deeper insights into the study by introducing PR-AUC reporting, over and above the ROC-AUC scores, to account for the highly imbalanced data.

Second, we also introduce two new few-shot machine learning models, namely the Protoypical Networks and Relation Networks, and explore their performance against the state of the art. The Prototypical and Relation Networks have been previously explored for the computer vision domain, but to our knowledge, have never been applied to the drug discovery domain.

While our results vary across the datasets used, they are consistent with the work of

Altae-Tran et al.[3]. The Prototypical Networks we introduce to this problem domain perform better on the Tox21 dataset based on ROC-AUC performance, while outperforming all other machine learning models in PR-AUC 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 PR-AUC provides more reliable insight into the performance of the models. The same generalising capabilities is not achieved on MUV data due to the nature of the data available within this dataset. The results on the DUD-E data does not give a clear indication of performance, and the excellent results obtained on one DUD-E target raises questions about hidden bias within the data. Therefore, we conclude that few-shot machine learning is effective for low-data ligand-based virtual screening depending on the nature of the data used. For data such as MUV, in which active compounds per target are scarce and each compound is structurally distinct from all others, the few-shot learning models struggle to generalise well. We find that on the Tox21 datasets, the Prototypical network is the best performing network, with much faster training times than our implementation of the Matching Networks. Prototypical Networks dominate all other networks in PR-AUC scores, and also have a slight edge when comparing ROC-AUC scores compared to the state of the art. The state of the art and Prototypical Networks perform significantly better than our implementation of Relation Networks. Hence, we conclude that Prototypical Networks offer better generalising capabilities for few-shot learning in ligand-based virtual screening than the Matching Networks component in the state of the art.

We also find that making use of learned embeddings through GCNs, as opposed to ECFPs, consistently results in better ROC-AUC and PR-AUC 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.

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) 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.

(18) 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.

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

(20) 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.

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

(22) 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.

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

(24) 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.

(25) 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.

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

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

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

(29) 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**,

(30) 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**,

(31) 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.

(32) 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.

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

(34) 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.