Neural Computing and Applications (2023) 35:13167–13185
https://doi.org/10.1007/s00521-023-08403-5

ORIGINAL ARTICLE

Few-shot learning via graph embeddings with convolutional networks for low-data molecular property prediction

Luis Torres[1] · Joel P. Arrais[1] · Bernardete Ribeiro[1]

Received: 3 July 2022 / Accepted: 13 February 2023 / Published online: 10 March 2023

Abstract

Graph neural networks and convolutional architectures have proven to be pivotal in improving the prediction of molecular properties in drug discovery. However, this is fundamentally a low data problem that is incompatible with regular deep learning approaches. Contemporary deep networks require large amounts of training data, which severely limits the prediction of new molecular entities from limited available data. In this paper, we address the challenge of low data in molecular property prediction by: (1) defining a set of deep learning architectures that accept compound chemical structures in the form of molecular graphs, (2) creating a few-shot learning strategy across graph neural networks and convolutional neural networks to leverage the rich information of graph embeddings, and (3) proposing a two-module meta-learning framework to learn from task-transferable knowledge and predict molecular properties on few-shot data. Furthermore, we conduct multiple experiments on two benchmark multiproperty datasets to demonstrate a superior performance over conventional graph-based baselines. ROC-AUC results for 10-shot experiments show an average improvement of $+11.37\%$ on Tox21 and $+0.53\%$ on SIDER, which are representative small-sized biological datasets for molecular property prediction.

Keywords Few-shot learning · Convolutional neural networks · Graph neural networks · Molecular property prediction

1 Introduction

Drug discovery and development is an extremely long and expensive process that aims to find innovative medical compounds ready to be formulated, synthesized and administered to a patient [1]. Despite the most recent scientific advances and ever-increasing understanding of biological systems, most of these compounds fail to be selected due to a lack of desirable molecular properties [2].

In lead optimization, only a small fraction of the molecules can pass virtual screening and enter clinical development. The higher the quality of these preclinical candidates, the higher the probability of successful drug development [3]. However, the major cost of this operation stems from exploring the entire chemical space to synthesize only a few drug candidates. Thus, the search for new classes of compounds with a suitable pharmacological profile from a small amount of labeled data is paramount [4].

Currently, artificial intelligence assists almost every step of drug discovery including target identification, lead discovery and optimization or preclinical data generation. These methods reduce the number of iterations required to discover novel and active compounds while eliminating those that are inactive, reactive and toxic [5].

With the evolution of artificial intelligence, developments in deep learning (DL) have played a crucial role in optimizing drug discovery. These algorithms motivated the application of new graph representation learning techniques to model systems of drug interaction and prediction. However, with only a few labeled molecules available, deep networks struggle to generalize well and achieve acceptable performance [6].

✉ Luis Torres
luistorres@dei.uc.pt

Joel P. Arrais
jpa@dei.uc.pt

Bernardete Ribeiro
bribeiro@dei.uc.pt

[1] Centre for Informatics and Systems of the University of Coimbra, Univ Coimbra, Coimbra, Portugal

Well-validated biological datasets (e.g., Tox21, SIDER) [7] are limited in size and very expensive to obtain. These scarce drug repositories include only a few compounds that share the same set of molecular properties. The resulting lack of biological information, including molecules sharing similar properties, bounds the performance of conventional approaches. This precondition sets the challenge of developing models to effectively predict small molecules in few-shot learning scenarios [8, 9].

Recent research has demonstrated that simple machine learning algorithms and random forest predictors are effective in learning meaningful structural information from just a few labelled compounds [10, 11]. On the other hand, transfer learning and data augmentation techniques also provide the domain knowledge required in cases where examples with supervised information are hard or impossible to obtain [12, 13].

Nonetheless, these techniques are often too expensive and resource intensive to perform in drug discovery campaigns. More recently, non-trivial few-shot learning predictors have been proposed to discover the properties of new molecules and recognize potential drug candidates for further development [14, 15]. These methods attempt to learn from a set of molecular property prediction tasks and generalize to new chemical properties given a just a few molecules available.

Small molecules can be viewed as comprehensive graph structures, where atoms are represented as nodes and chemical bonds as edges shared by neighbors in a graph [16–18]. These graph-level representations account for the spatial arrangement of atoms and bonds as well as interactions between neighboring nodes and edges. This approach is more suitable for representation learning than sequence-based methods that describe molecules as sequential features such as SMILES (Simplified Molecular Input Line Entry System) strings [19].

These unique graph features can be used by deep learning pipelines, which fail to predict molecular properties with limited available data. This limitation prompts the need to explore models that quickly adapt across tasks to predict new properties on few-shot data [20, 21].

2 Related work

Few-shot learning methods have emerged as critical tools to accelerate and optimize drug discovery. These are algorithms that target at generalizing from small data collections to predict new systems from a limited amount of labeled information. Recently, few-shot models have proven effective in modeling molecules as comprehensive graph structures used for graph-based representation learning. Graph neural networks leverage this information to build molecular embeddings by treating atoms as nodes and chemical bonds as edges. Node and edge embeddings can later be used to support the prediction of molecular properties. Deep networks such as convolutional neural networks also manipulate these continuous vectorial representations to encode molecular graphs in a form suitable for few-shot molecule prediction.

2.1 Few-shot learning

Humans have an innate ability to recognize new objects and representations quickly from just a few examples. Prior knowledge helps to distinguish new concepts based on a generalized perception of an extensive and diverse set of representations. Thus, the ability to few-shot learn different representations by observing a concept and generating meaningful and diverse variations is very important when classifying new instances of unknown concepts [22].

This strategy attempts to adapt from previously seen classes to predict unseen representations from just a few labeled examples. This idea of few-shot generalization gave rise to few-shot learning methods [23].

Few-shot learning (FSL) was introduced by Fei-Fei et al. [24] in the field of computer vision and image processing. This approach presents a fundamental feature, which is the ability to predict based on prior experience by transferring knowledge across tasks.

In drug discovery, molecular property prediction is a few-shot learning problem since only a few molecules can pass virtual screening to be further evaluated in lead optimization. At this stage, few-shot models attempt to learn a predictor from a set of molecular property tasks and generalize to new chemical properties from a small amount of labelled molecules.

Altae-Tran et al. [25] introduced an iteratively refined LSTM (IterRefLSTM) model and adapted a classic FSL algorithm to handle these molecular property prediction tasks by iteratively coevolving undirected graph features. In this pioneering study, a graph neural network acts as a molecular encoder and learns few-shot representations across tasks to provide an inductive bias from prior experience that guides the search for the optimal model parameters.

2.2 Graph neural networks

More recently, the fast adaptability of few-shot learning methods foregrounded graph neural networks (GNNs) as promising strategies to model nonlinear systems of molecule prediction.

GNNs have shown promising results by representing molecules as topological graphs of node and edge features. Graphs are characterized by specific neighborhood

aggregation functions to update node representations and iteratively build graph-level embeddings.

Graph embedding methodologies, such as graph convolutional networks (GCN) or node2vec, are able to perform feature representation learning to obtain comprehensive graph embeddings. While GCNs use node and edge aggregation to compute graph embeddings, node2vec is inspired by powerful natural language processing algorithms (word2vec) to explore the relationship between nodes and edges in a graph as words in a sentence [26].

In drug discovery, the generated embeddings can be used to learn and predict properties of molecular graph features representing molecules. Hu et al. [27] propose novel strategies to pretrain graph-based networks and assist the learning of local and global information for molecular property prediction using graph embeddings.

GNNs such as GCN, graph isomorphism networks (GIN), GraphSAGE and graph attention networks (GAT) have significantly improved the discovery of new chemical entities with enhanced therapeutic properties [28, 29]. Graph convolution architectures include a convolutional component to aggregate nodes and edges from close neighbors on the same receptive field. This type of graph networks aggregates node features by applying convolutional filters on the aggregated nodes. GraphSAGE uses a different sampling strategy by transforming the original graph convolution training procedure into a training method divided in small batches centered on the nodes. An inductive graph convolution operation also extends the aggregation to generate relevant graph features on unseen nodes and edges [30, 31]. GAT models are an extension of the conventional GCN that perform node aggregation by adding specific attention weights to certain nodes. Thus, GAT computes node aggregates based on attention scores that translate the most meaningful parts of a node's neighborhood [32]. Among graph neural network architectures, GIN presents the maximum discriminative power by generalizing the Weisfeiler–Lehman (WL) isomorphism test to capture different graph structures [33].

Guo et al. [34] proposed a meta-learning framework (Meta-GNN) across GNNs to predict molecular properties using a novel pre-training technique. Self-supervised learning objectives such as bond reconstruction and atom type prediction were included to improve generalization on few-shot data.

Structural aspects of these graph features model complex patterns and local dependencies between neighboring nodes and edges. Node and edge features can be later used to predict the behavior of active compounds or specific molecular properties of multiple drug candidates.

2.3 Convolutional neural networks

Convolutional neural network (CNN) architectures can process molecules to build up features using learnable convolutional layers [35]. These architectures consist of a set of layers with multiple neurons and a set of parameters representing the strength of the connections between neurons [36].

CNNs learn by applying a backpropagation algorithm to tell the neural network how to change its internal parameters. This shift allows CNNs to compute the representation in a layer from the representation of the previous layer. As we progress to deeper layers, the ability to distinguish global patterns and local dependencies increases [37].

Sequential layers treat the input as an image viewed as a grid where each element corresponds to a pixel. Convolutional weight matrices act as filters that operate in local receptive fields of neighboring pixels to develop structure in high-dimensional feature spaces.

In drug discovery, molecular features can be handled by convolutional architectures to model nonlinear functions of molecular structure and transform small molecules into deep representations. To explore the potential of such representations, Shi et al. [38] proposed a CNN method to establish prediction models of molecular properties for automated virtual screening and ADMET prediction.

Moreover, Ståhl et al. [35] introduced a flexible CNN architecture to incorporate global and local information of deep representations to effectively predict undirected graph features representing molecules.

By converting these molecular features into continuous embedded descriptors, CNNs can use deep representations to infer the complex concepts of molecular structures and boost the prediction of molecular properties [39, 40].

3 Proposed approach

Few-shot molecular property prediction depicts a practical learning scenario where a model f is provided with a small number of training examples. This model intends to predict new molecular properties from just a few examples by generalizing from prior learning experience.

In this direction, the problem of low data can be defined to the following extent: Given the molecular property labels $y_i \in Y$ of a small set of molecules D, the goal is to learn a function f to map a molecule $d_i \in D$ to a given molecular property $y \in Y$ in the test data. This can be formalized by

$$f : d_i \mapsto y_i. \tag{1}$$

GNNs have proven critical when optimizing the prediction of *drug-like* properties by exploring unique graph features [41]. Generally, the objective is to obtain an embedding for a set of nodes and edges in a graph to generalize from node-edge representation learning. Subsequently, these continuous vectorial representations are fed into a simple classifier to predict a given property or molecular label.

Conventionally, a GNN f converts a graph representation of a node $v \in V$ to an embedding h based on the contextual information of a subgraph of its neighboring nodes $u \in N(V)$ and edges $e = (v, u)$. These embedded descriptors preserve the original graph structure and properties to map them to a reduced space of comprehensive representations (see Fig. 1).

However, most of the aforementioned graph-based networks neglect the practical use of such representations and fail to leverage node and edge embeddings to enhance the prediction of molecular properties. Hence, the potential of these graph embeddings has yet to be fully explored successfully.

To address the limitations of existing studies and exploit the singular properties of graph-level representations, we propose an innovative approach, FS-GNNConv, for molecular property prediction. The proposed GNN-CNN architecture explores the local dependencies of molecular graph embeddings to learn complex patterns and produce stronger features for representation learning. The challenge of low-data is systematically addressed by proposing a two-module meta-learning framework to quickly adapt to new molecular properties across few-shot tasks. This strategy saves resources and promotes fast adaptation to new experimental tasks, similar or marginally identical to those found in training. The major contributions of this work can be summarized as follows

(i) a two-module GNN-CNN architecture that accepts the compound chemical structure to exploit the rich information of graph embeddings;
(ii) a few-shot learning (FSL) strategy to learn from task-transferable knowledge and predict the behavior of active compounds in new experimental systems;
(iii) a meta-learning framework to iteratively optimize model parameters and successively gather generic knowledge across tasks to predict task-specific molecular properties;
(iv) experiments on real multiproperty prediction data to demonstrate the predictive power of the proposed model when inferring specific target properties adaptively.

3.1 Embedding module: graph isomorphism network

GNNs are extremely effective in modeling molecular properties by describing the molecular structure as a graph.

In a molecular graph, each node represents an atom, and each edge represents a chemical bond between atoms. Both can be described by multiple features encoding structural and stereochemical attributes.

Let a molecular graph be defined as $G = (V, E)$ with V as the set of nodes and E as the set of edges $e = (v, u)$ connecting each pair of nodes in a neighborhood $N(V)$, where $v \in V, u \in N(v)$. We denote $M = \{G_1 \ldots, G_N\}$ as the set of molecular graphs and Y as the set of molecular property labels $Y = \{y_1 \ldots, y_N\}$. The objective is to predict a molecular property y_i by learning a nonlinear function f that maps a molecular graph G to an embedding h_G,

$$f : G \mapsto h_G \tag{2}$$

where h_G is the graph-level embedding used to assist the prediction.

Recent research in this field suggests two main groups of GNNs based on the neighborhood aggregation function for graph embedding. Spectral GNNs decompose each graph to approximate the spectral filters of GNN message-passing layers. On the other hand, spatial GNNs compute the neighborhood aggregation from the node spatial relations between neighboring nodes and edges in the graph [42].

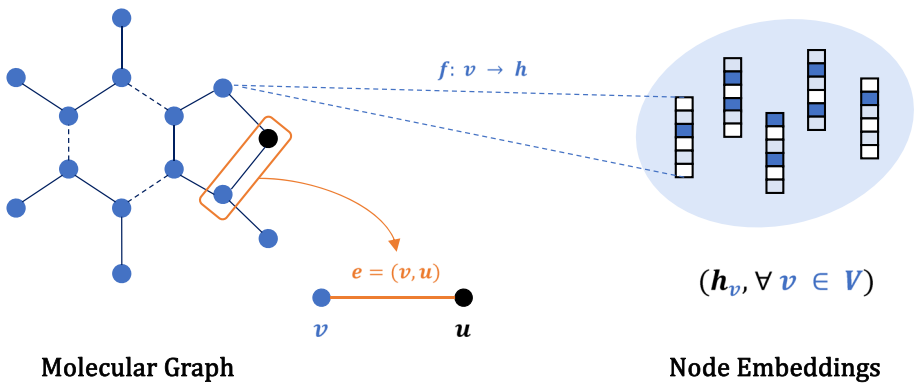

Fig. 1 Graphical depiction of a molecule representation as a molecular graph. Node embeddings h_v are mapped from molecular graphs to a low-dimensional feature space of vectorial embeddings

Spatial-based GNNs operate under two arbitrarily differentiable functions: a neighborhood aggregation function AGGREGATE and a COMBINE step to merge and update node and edge features.

GNNs are iterative message-passing networks. Multiple message-passing iterations l update nodes h_v^l using prior representations of a node h_v^{l-1} and representations of its close neighbors in the graph h_u^{l-1}. During a message-passing iteration, node embeddings h_v with $v \in V$ are updated using representations of neighboring nodes $u \in N(v)$ and edges $e = (v, u)$ [43].

In this work, we apply a spatial-based GIN [33] as the first module to compute graph embeddings for further learning and prediction. In this case, the GNN implements both COMBINE and AGGREGATE functions as the sum of node and edge features [27]. Thus, node embeddings are updated for each message-passing iteration l by

$$m_{N(v)}^l = \text{AGGREGATE}^l(\{h_u^{l-1}, \forall u \in N(v)\}, \{h_e^{l-1} : e = (v, u)\}) \quad (3)$$

$$h_v^l = \sigma(\text{MLP}^l(\text{COMBINE}^l(h_v^{l-1}, m_{N(v)}^l))) \quad (4)$$

where m is the "neural message" passed through the network, h_u^l are the embeddings of neighboring nodes, and h_e^l is the feature vector of an edge between nodes u and v. An UPDATE step includes a multi-layer perceptron *MLP* to introduce nonlinearity and a nonlinear activation function σ (ReLU). More specifically, the GNN updates node representations by

$$h_v^l = \text{ReLU}\left(\text{MLP}^l\left(\sum_{u \in N(v) \cup v} h_u^{l-1} + \sum_{e=(v,u): u \in N(v) \cup v} h_e^{l-1}\right)\right). \quad (5)$$

The idea is that a message is generated from the information about neighboring nodes and combined with previous embedded representations of a node v, h_v^{l-1} to obtain the updated embedding h_v^l. The original inputs for aggregation are the initial node and edge attributes h_v^0 and h_e^0. After l iterations, the final embedding, $h_v^l, \forall v \in V$, incorporates information about the node (atom) and the contextual subgraph of nodes and edges (bonds between atoms) in a l-hop neighborhood [44].

Finally, a READOUT mean-pooling operation is performed to obtain an embedding h_G. This graph-level representation is obtained by averaging the node embeddings h_v at the final message-passing layer l

$$h_G = \text{mean}(\{h_v^l : v \in V\}). \quad (6)$$

These graph embeddings can be saved for further learning, as we will present in the next section.

A graphical depiction of the proposed model architecture is shown in Fig. 2. For graph operations, the nodes being operated on are displayed in blue, with neighboring nodes shown in black. For AGGREGATE, COMBINE and UPDATE, the operations are shown for a single node $v \in V$ and performed on all nodes v in the graph, simultaneously. In this case, we consider graph operations for $L = 5$ message-passing layers and the READOUT operation is performed at the final layer. In the convolution operations, h_{conv} describes the deep representations extracted by a CNN g from graph embeddings of size = 300. Different blue squares denote different values of the node-level and graph-level embeddings h_v and h_G, respectively. Different orange squares denote different values of deep representations h_{conv}.

Node and edge features are described by atom and bond attributes. Node attributes include the atom number (AN) and atom chirality (AC) to describe the type of atom, how it is connected and the spatial interaction behavior with neighboring nodes. Edge attributes include bond type (BT) and bond direction (BD) to specify the structural aspects of chemical bonds and the spatial orientation of the edges. Formally, the input node and edge features are described as

$$h_v^0 = (v_{\text{AN}}, v_{\text{AC}}) \quad (7)$$

$$h_e^0 = (e_{\text{BT}}, e_{\text{BD}}) \quad (8)$$

with $(,)$ as a concatenation operation and e and v as edge and node attributes, respectively.

Pre-trained GIN, GCN and other graph-based architectures have been widely used in drug discovery applications. In this sense, the GIN model is pre-trained using a recent pre-training technique [27] to achieve better parameter initialization and learn global and more generic descriptors.

3.2 Prediction module: convolutional neural network

CNNs inherit many of the properties of the artificial neural networks to develop structures in a feature space where the complexity stratifies along with different layers. These layers are made up of neurons including a set of learnable weights and biases.

Convolutional layers consist of several convolutional filters (weight matrices) capable of extracting local features from the input. In the forward pass, these representations are propagated across convolutional layers while convolutional filters slide through the spatial dimensions of the input. The output feature maps are the result of the convolution operation between the convolutional filters and different positions in the input vector [29].

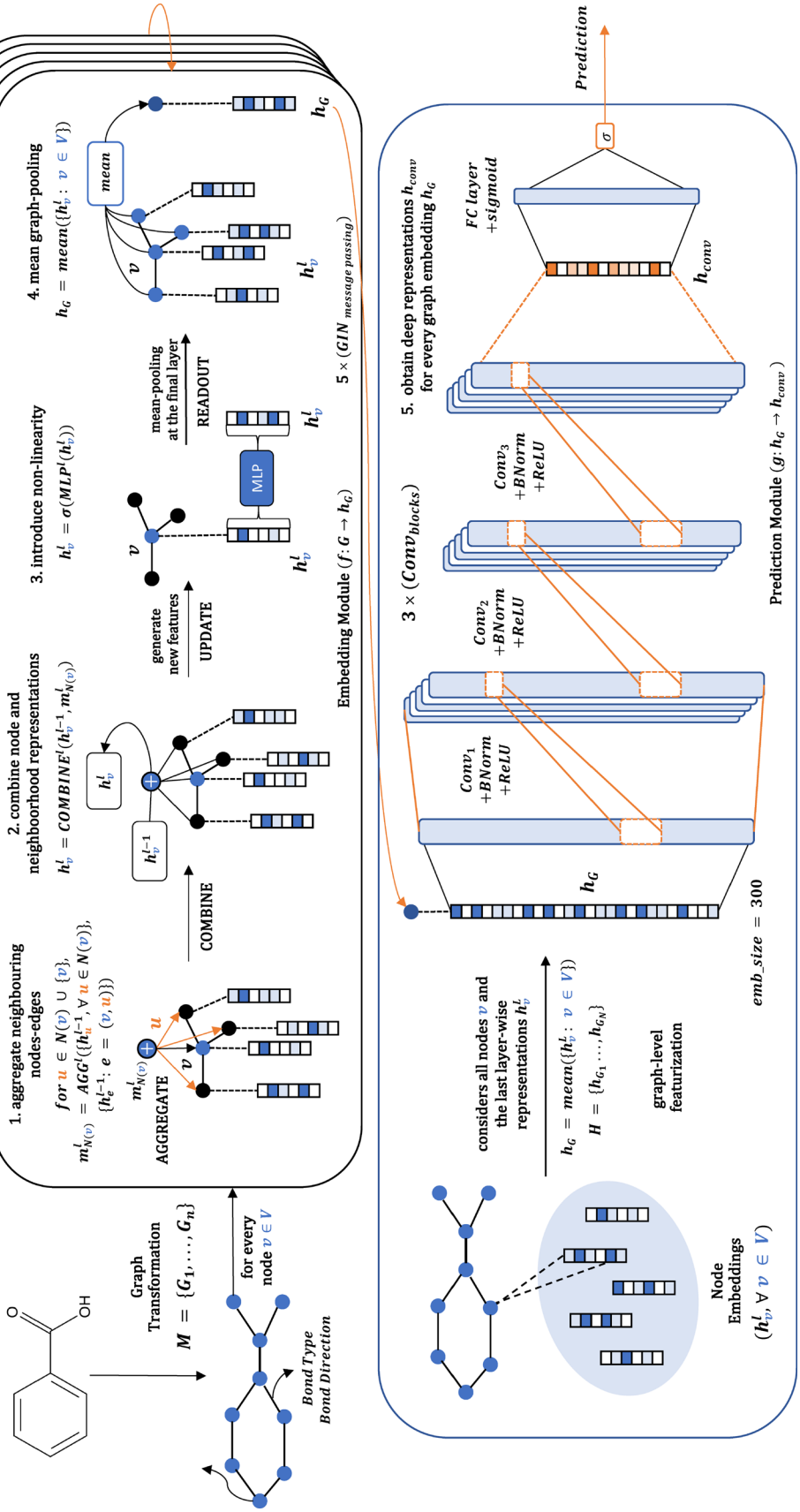

Fig. 2 Graphical depiction of the proposed neural network architecture

These sequential layers emerge as detectors of local patterns by restricting the connections with neurons to small regions of the input. The subsequent increase in the number of convolutional filters and in the combinatorial size of the feature space allows the extraction of more complex and generic descriptors [37].

In this study, one-dimensional CNNs are explored to build a molecular property prediction module. By incorporating the graph structure information enclosed in graph embeddings, CNNs can discriminate important patterns between close and distant neighbors in the graph.

To this end, node representations h_v are used to calculate graph embeddings h_G through node averaging and mean pooling, as described in the previous section. Embeddings are then used as input feature vectors for further computation.

First, we collect the embeddings $H = \{h_{G_1} \ldots, h_{G_N}\}$ obtained from the original graphs G_i. Then, we perform the nonlinear mapping of h_G to a deep representation vector h_{conv} using a CNN g. This relation can be defined by

$$g : h_G \mapsto h_{\text{conv}}. \tag{9}$$

Deep representations h_{conv} are then propagated to become increasingly smaller and more complex as we progress to deeper layers.

The prediction module g presents a conventional architecture with 3 CNN blocks. One-dimensional input embeddings of size $= 300$ are convoluted with filters of size 3×1 followed by batch normalization with momentum $= 1, \epsilon = 1e-5$ and ReLU activation (see Table 1).

Table 1 CNN architecture details

Layer	conv1D	conv1D	conv1D	FC
dims $[input, output]$	$[300, 128]$	$[128, 64]$	$[64, 64]$	$[64, 1]$
batch-normalization	Yes	Yes	Yes	–
nonlinearity	ReLU	ReLU	ReLU	sigmoid

The convolution operation consists in a set of multiplication operations and later sum. The convolution between convolutional filters and the elements of input embeddings h_G, over which these filters slide, returns the feature map of convolutional outputs h_{conv}. This process is repeated across layers to return the output of the convolution operation between weight matrices W and the regions to which they are connected in the input vector (see Fig. 3) [37].

After convolution, batch normalization helps to coordinate the updates of multiple layers in the CNN module by scaling the output feature maps of convolutional layers. This is done by normalizing the activations of each input variable per mini-batch, such as the neural activations from previous layers. This process of standardization stabilizes and speeds up model convergence while reducing the generalization error [45].

Then, ReLU (Rectified Linear Unit) units apply a function that converts the values present in the input vectors of elements x to non-negative values: $\max(0, x)$. This operation sets a threshold at 0, where the negative values are nullified, accelerating the process of convergence due to the linear profile of ReLU activation [46].

Hence, the feature maps for each l-th convolutional layer take the form

$$h^l = \max(0, \text{BatchNorm}(W^{l-1,l} * h^{l-1} + b^l)) \tag{10}$$

where $W^{l-1,l}$ is the weight matrix connecting units of layer $l-1$ to units in layer l and b^l the bias vector for a layer l. $*$ is the convolution operation to return the output units for neural activation and h^l, h^{l-1} are the hidden vectors in layers l and $l-1$, respectively.

Finally, a dense fully connected layer (FC) followed by sigmoid activation uses the output of the last convolutional

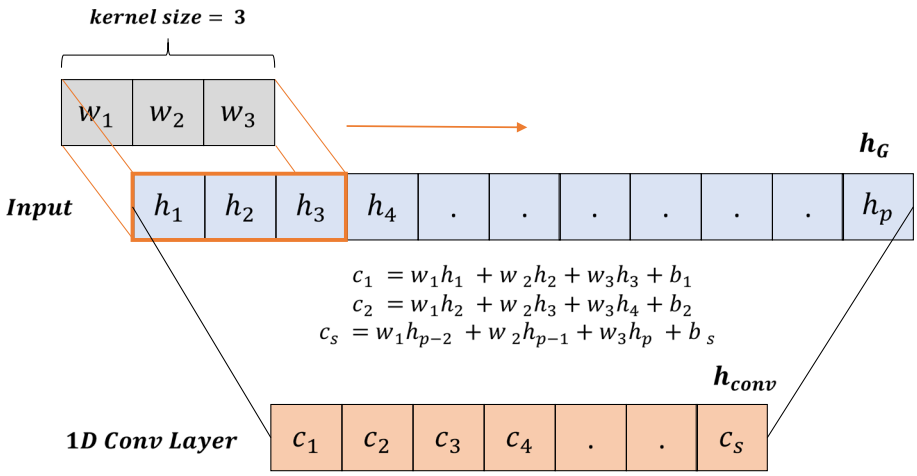

Fig. 3 Schematic of the one-dimensional (1D) convolutional operation

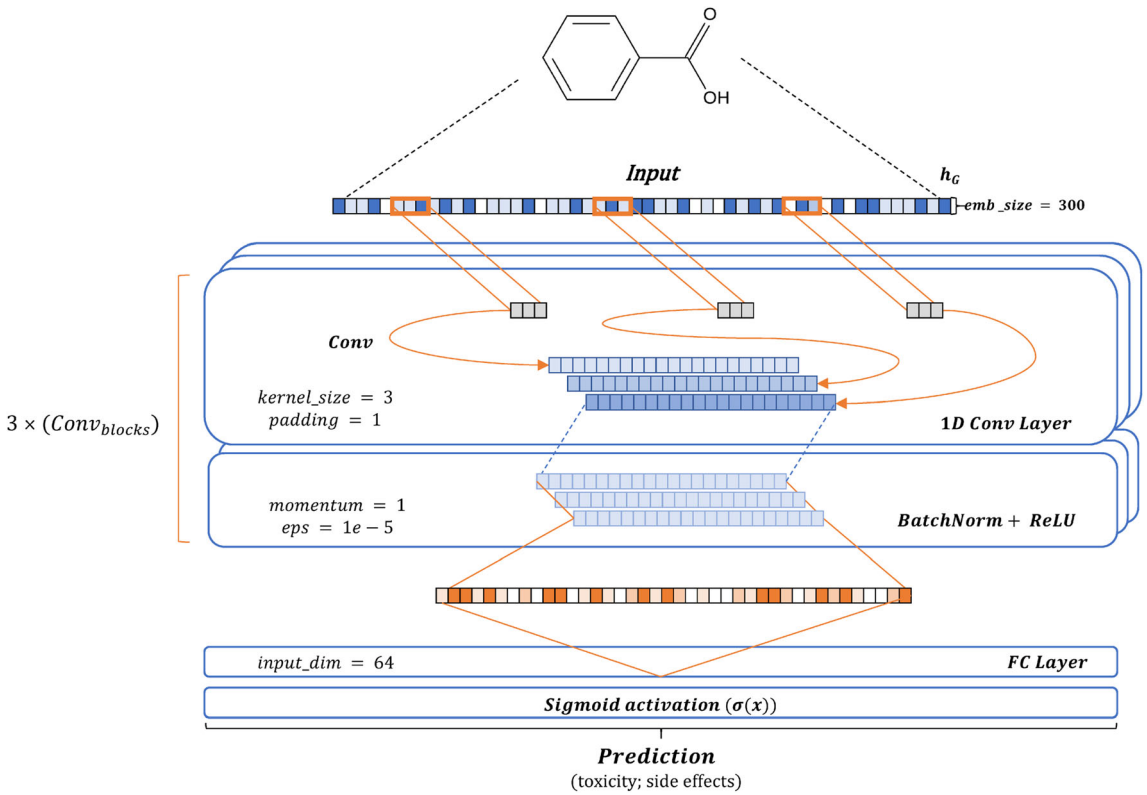

Fig. 4 Graphical representation of the CNN module architecture

layer $l = 3$ to compute the prediction (condensed in a value $\in \{0, 1\}$).

The CNN module treats graph-level embeddings as images. Each embedding is viewed as a grid and elements in the same receptive field are convoluted to build a representation that accounts for the dependencies between neighboring nodes and edges. Local connections between close and distant regions of graph embeddings are then explored to compute increasingly smaller and complex representations.

The convolution of graph embeddings h_G transforms molecular graphs into deep vectorial representations h_{conv}. By propagating such representations across convolutional layers, CNNs model complex patterns and local dependencies within molecular structures as a function for molecular property prediction.

A detailed representation of the CNN prediction module is depicted in Fig. 4.

4 Training and inference

In this section, we introduce a few-shot meta-learning framework on the basis of model-agnostic meta-learning (MAML) Finn et al. [47] to predict new molecular properties across tasks.

Meta-learning relies on prior knowledge to systematically revisit previous learning episodes and define a promising strategy based on experience. Fewer examples are required with an increasing number of learning tasks making the process faster and more efficient. This type of non-trivial learning adapts and generalizes to new representations from limited available data [48, 49].

From this perspective, we address the challenge of low data by optimizing the proposed model across several learning tasks. In meta-training, the model is trained on a labeled support set and evaluated on a disjoint query set for each task. Previous learning episodes are then used to predict new tasks and optimize the algorithm to the new task at hand. This process is repeated across tasks to return a predictor that generalizes well to unseen representations in few-shot data.

The goal is to predict a molecular property (e.g., toxicity, side effects) of a query molecule x so that $\{f_\theta(x), g_{\theta^*}(h(x))\} : M \Rightarrow \{0, 1\} \in Y$, where M is the space of all molecular graphs G, $h(x)$ is the output embedding from a GNN f_θ, g_{θ^*} is a CNN and Y are the molecular property labels.

In this study, we train two meta-models, a GNN f_θ and a CNN g_{θ^*} with parameters θ and θ^* across tasks t from a distribution $\rho(T)$. Meta-training and meta-testing sets are sampled for each molecular property task t. Both include a

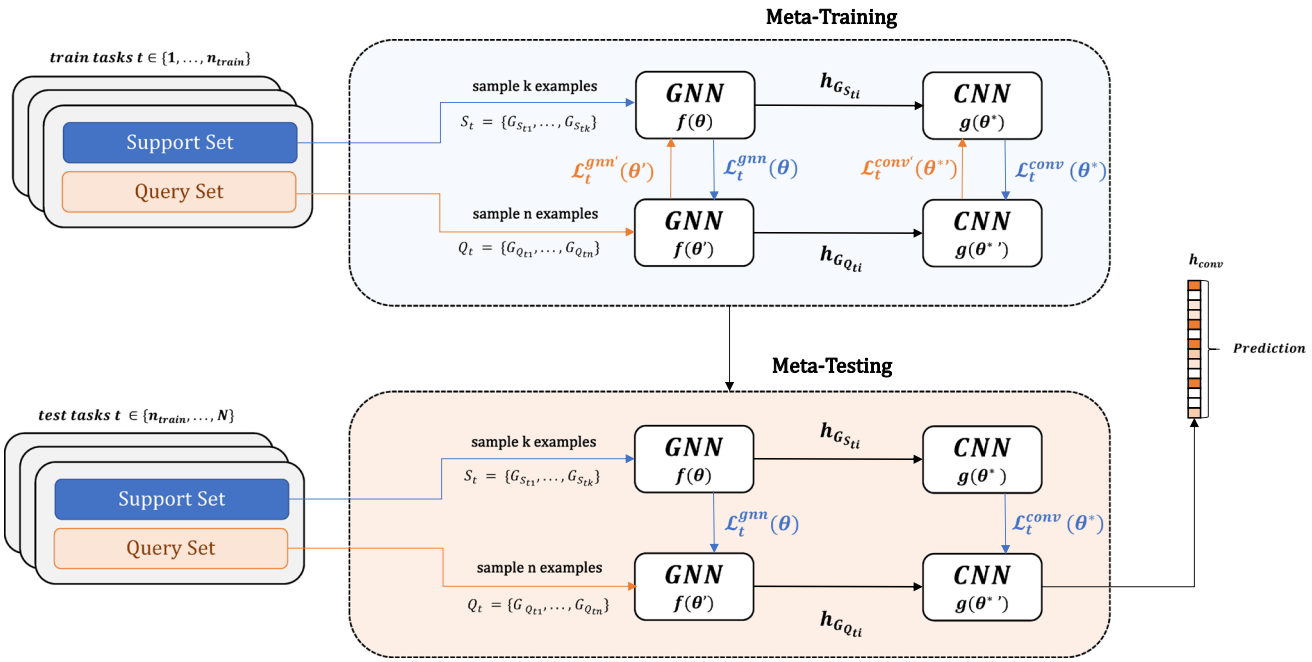

Fig. 5 Schematic of the proposed meta-learning framework for few-shot molecular property prediction

support set S for training and a query set Q for evaluation. For each task t, both models are parameterized by a task-specific support set S_t and query set Q_t for each task.

Particularly, under a k-shot meta-training, both models f_θ and g_{θ^*} with parameters θ and θ^*, are constantly updated, trained on S_t and evaluated on Q_t for each task.

First, for each k-shot task t, k support samples $G_{S_{t_i}}$ are randomly sampled and fed into the GNN-CNN two-module architecture to compute the support losses $\mathcal{L}_t^{\text{gnn}}$ and $\mathcal{L}_t^{\text{conv}}$.

Subsequently, the support losses are used to update the model parameters $\theta \rightarrow \theta'$, $\theta^* \rightarrow \theta^{*'}$, and both models are evaluated on a query set Q_t to compute the query losses $\mathcal{L}_t^{\text{gnn}'}$ and $\mathcal{L}_t^{\text{conv}'}$ using the remaining n samples for each task.

In practice, we update the model parameters to adapt to a new task t. Tasks include k support samples to compute the support losses for both modules, $\mathcal{L}_t^{\text{gnn}}(\theta)$ and $\mathcal{L}_t^{\text{conv}}(\theta^*)$. In this process, we apply a few gradient steps

$$\theta_t = \theta - \alpha \nabla_\theta \mathcal{L}_t^{\text{gnn}}(\theta) \tag{11}$$

$$\theta_t^* = \theta^* - \alpha^* \nabla_{\theta^*} \mathcal{L}_t^{\text{conv}}(\theta^*) \tag{12}$$

with α and α^* as the step sizes for the gradient descent updates.

In meta-testing, we randomly sample a support set of size k for a new task t to optimize the model parameters $\theta \rightarrow \theta'$, $\theta^* \rightarrow \theta^{*'}$ through a few gradient descent steps. Molecular properties are finally predicted in a disjoint query set with the remaining samples. Thus, the updated model parameters are used to generalize to new samples and compute deep representations h_{conv} to assist the prediction of molecular properties.

In this framework, it is expected to obtain optimized parameters that map the molecular graphs to different task-specific properties. The goal is to generalize well to new tasks in the test data after a few gradient steps. A schematic of the proposed meta-learning framework is shown in Fig. 5.

When training the model on a specific data collection, tasks or assays are divided in training and testing tasks. As stated previously, training consists in a set of learning episodes. For each episode, a task from the training tasks is randomly sampled and a support set of size $n_+ + n_-$ (with n_+ positive and n_- negative samples) and a batch of queries are randomly sampled from that task. Each learning episode takes a few gradient descent steps to minimize the loss function using Adam optimizer.

The accuracy of the model is evaluated separately for each test task at test time. For each test task, a support set of size $n_+ + n_-$ is randomly sampled from the data from that task. ROC-AUC scores are then evaluated on the remaining data for that task.

In the results Sect. 5.3, we use the notation $(5+, 5-)$ and $(10+, 10-)$ to represent $n_+ = 5$, $n_- = 5$ and $n_+ = 10$, $n_- = 10$, respectively. Appendix Sect. 7 provides further details about the tasks considered for each data collection.

4.1 Cost-sensitive loss for imbalanced classification

The loss for both modules $\mathcal{L}^{\text{gnn}}$ and $\mathcal{L}^{cnn}$ is the binary cross-entropy loss over the predicted properties y' and the molecular property ground-truth labels y with k as the number of samples,

$$\mathcal{L} = -\frac{1}{k}\sum_{i=1}^{k} y_i \ \log(y_i') + (1 - y_i) \ \log(1 - y_i'). \tag{13}$$

However, the problem of class imbalance in few-shot data prevents us from obtaining superior performance for either the Tox21 or SIDER benchmarks. To address this issue, we introduce a customized version of binary cross-entropy loss to establish a weight for the minority class as a weighted version of the original objective. This customized loss function takes into account the distribution of each class to penalize failed predictions for rare instances, which greatly impact the loss value.

The weighted version of binary cross-entropy defines a weight p for the minority class,

$$\mathcal{L} = -\frac{1}{k}\sum_{i=1}^{k} p \ y_i \ \log(y_i') + (1 - y_i) \ \log(1 - y_i') \tag{14}$$

with p defined as the ratio between positive and negative samples. For instance, if a dataset contains 100 positive and 500 negative examples of a single class, then p for the class should be equal to $\frac{500}{100} = 5$. Since different tasks present different positive/negative distributions, we determine p by exploring multiple values between 1 and 50 and selecting those that return superior performance ($p = 35$ for Tox21 and $p = 1$ for SIDER due to task variability).

5 Experiments

The present section reports multiple experiments on two benchmark datasets (Tox21 and SIDER).

The Tox21 dataset comprises qualitative toxicity measurements for 7831 compounds for 12 biological targets including nuclear receptors (NR) and stress response pathways (SR). Each sample represents a compound with 12 binary labels for 12 toxicology experiments [50].

Table 2 Tox21 comprises qualitative toxicity measurements related to 12 biological targets

Tox21 benchmark			
Datatype	Task type	# Tasks	#Compounds
SMILES	Classification	12	7831

Table 3 SIDER includes a database of marketed medicines grouped into 27 different system organ classes

SIDER benchmark			
Datatype	Task type	# Tasks	#Compounds
SMILES	Classification	27	1427

Tox21 is a machine-learning challenge formerly won by a multitask learning approach across deep networks. The main goal is to predict the toxicity of small molecules for a specific NR or SR. In a few-shot learning setting, we use a different proportion of training and testing tasks, disregarding the original train-test split. For a total of 12 tasks, the data were split into 9 tasks for training and 3 for testing (see Table 2).

SIDER is a collection of 1427 well-validated drugs and adverse drug reactions (ADRs) grouped into 27 system organ classes [51]. SIDER data are extracted from several public articles and publications containing labeled information on marketed drugs, including side effect frequencies, drug-target interactions and drug/side effect classification. The goal is to predict whether a compound triggers a side effect for 27 organ systems. For a total of 27 tasks, the data were split into 21 tasks for training and 6 for testing (see Table 3).

Both datasets are broadly distinct and represent diverse collections of molecular scaffolds. Thus, it is expected that models perform differently on both data collections.

Raw data of molecules are given in the form of SMILES strings and converted into molecular graphs using the Python library Rdkit.Chem [52]. These SMILES strings are converted into node and edge features to best describe the structure and spatial arrangement of the molecules used in the experiments.

5.1 Baselines

The proposed FS-GNNConv model is compared with three graph-based models:

(i) GIN: Pre-trained version of GIN.
(ii) GCN: Pre-trained GCN model. The GNN includes a convolutional component for node aggregation. Nodes are seen as pixels, and neighbors in the same receptive field are used to compute node embeddings as the output of the convolution [53].
(iii) GraphSAGE: Pre-trained GraphSAGE model. Graph-based network that samples and aggregates neighboring embeddings to leverage relevant graph features. This is an inductive framework that exploits these node attributes to efficiently

Table 4 Average ROC-AUC scores for binary classification with 5-shots on benchmark datasets Tox21 and SIDER

Dataset	Task	GIN	GCN	GraphSAGE	FS-GNNConv (GIN + CNN)	$\triangle(\boldsymbol{AUC})$
5-shot $(5+, 5-)$						
Tox21	**SR-HSE**	61.44 ± 1.17	65.76 ± 2.49	64.19 ± 2.50	$\boldsymbol{76.37 \pm 0.48}$	$+10.61$
	SR-MMP	57.55 ± 0.90	64.85 ± 1.28	63.56 ± 3.89	$\boldsymbol{77.60 \pm 0.33}$	$+12.75$
	SR-p53	59.15 ± 1.13	63.02 ± 1.49	61.75 ± 3.45	$\boldsymbol{72.67 \pm 0.59}$	$+9.65$
	Average	59.38	64.54	63.17	***75.55***	$+\mathbf{11.01}$
SIDER	**R.U.D.**	69.77 ± 1.08	60.62 ± 1.56	62.62 ± 0.64	$\boldsymbol{70.11 \pm 0.63}$	$+0.34$
	P.P.P.C.	$\boldsymbol{77.05 \pm 0.66}$	71.89 ± 1.25	74.16 ± 1.19	70.95 ± 0.84	-6.10
	E.L.D.	70.24 ± 1.03	62.78 ± 0.96	64.50 ± 0.75	$\boldsymbol{70.55 \pm 0.45}$	$+0.31$
	C.D.	68.66 ± 0.90	60.82 ± 1.11	61.81 ± 0.76	$\boldsymbol{70.68 \pm 0.58}$	$+2.02$
	N.S.D.	65.23 ± 0.70	58.77 ± 2.27	59.00 ± 1.41	$\boldsymbol{67.61 \pm 0.74}$	$+2.38$
	I.P.P.C.	$\boldsymbol{72.92 \pm 1.03}$	65.62 ± 1.95	67.01 ± 0.76	72.01 ± 0.68	-0.91
	Average	***70.64***	63.42	64.85	**70.32**	$-\mathbf{0.32}$

Table 5 Average ROC-AUC scores for binary classification with 10-shots on benchmark datasets Tox21 and SIDER

Dataset	Task	GIN	GCN	GraphSAGE	FS-GNNConv (GIN + CNN)	$\triangle(\boldsymbol{AUC})$
10-shot $(10+, 10-)$						
Tox21	**SR-HSE**	57.05 ± 0.56	64.92 ± 1.42	65.24 ± 3.20	$\boldsymbol{77.57 \pm 0.39}$	$+12.33$
	SR-MMP	54.76 ± 0.23	66.01 ± 0.28	64.59 ± 3.78	$\boldsymbol{77.99 \pm 0.36}$	$+11.98$
	SR-p53	53.29 ± 0.29	63.07 ± 0.14	62.52 ± 2.78	$\boldsymbol{72.55 \pm 0.48}$	$+9.48$
	Average	55.03	64.67	64.12	***76.04***	$+\mathbf{11.37}$
SIDER	**R.U.D.**	69.02 ± 0.47	61.01 ± 0.19	63.80 ± 0.41	$\boldsymbol{70.58 \pm 0.48}$	$+1.56$
	P.P.P.C.	$\boldsymbol{77.63 \pm 0.82}$	70.88 ± 0.71	73.55 ± 0.68	71.18 ± 0.61	-6.45
	E.L.D.	70.60 ± 0.43	62.41 ± 0.20	64.81 ± 0.43	$\boldsymbol{71.09 \pm 0.41}$	$+0.49$
	C.D.	67.49 ± 0.61	60.62 ± 0.14	63.20 ± 0.40	$\boldsymbol{71.36 \pm 0.53}$	$+3.87$
	N.S.D.	62.80 ± 0.60	58.99 ± 0.45	59.32 ± 0.68	$\boldsymbol{67.67 \pm 0.67}$	$+4.87$
	I.P.P.C.	$\boldsymbol{73.28 \pm 0.57}$	65.98 ± 0.17	68.79 ± 0.60	72.10 ± 0.48	-1.18
	Average	70.13	63.32	65.58	***70.66***	$+\mathbf{0.53}$

generate representations on previously unseen data [30].

All baselines were pre-trained with the GCN, GIN and GraphSAGE models of Hu et al. [27] to improve performance. A meta-learning framework was applied to all baselines to achieve comparable results.

5.2 Evaluation metrics

Binary classification of molecular properties is evaluated by ROC-AUC scores on the query set of each test task. For a given test task, we randomly sample a support set with k examples to collect the data points for that task. Then, we evaluate the ROC-AUC scores for the model on the remainder of the data points for each test task in a disjoint query set.

In model evaluation, each task is considered independent, and we report the results for a 2-way binary classification with 5-shots and 10-shots. To show more robust results, this procedure is reported 20 times for each test task using 20 randomly sampled support sets to calculate the average ROC-AUC scores. The notation (n_+, n_-) indicates random support sets with n_+ positive samples and n_- negative samples.

Experimental results including the mean and standard deviation of ROC-AUC scores for $(5+, 5-)$ and $(10+, 10-)$ random support sets are displayed in Tables 4 and 5.

5.3 Results

In this work, we systematically address the low-data problem in molecular property prediction by introducing a

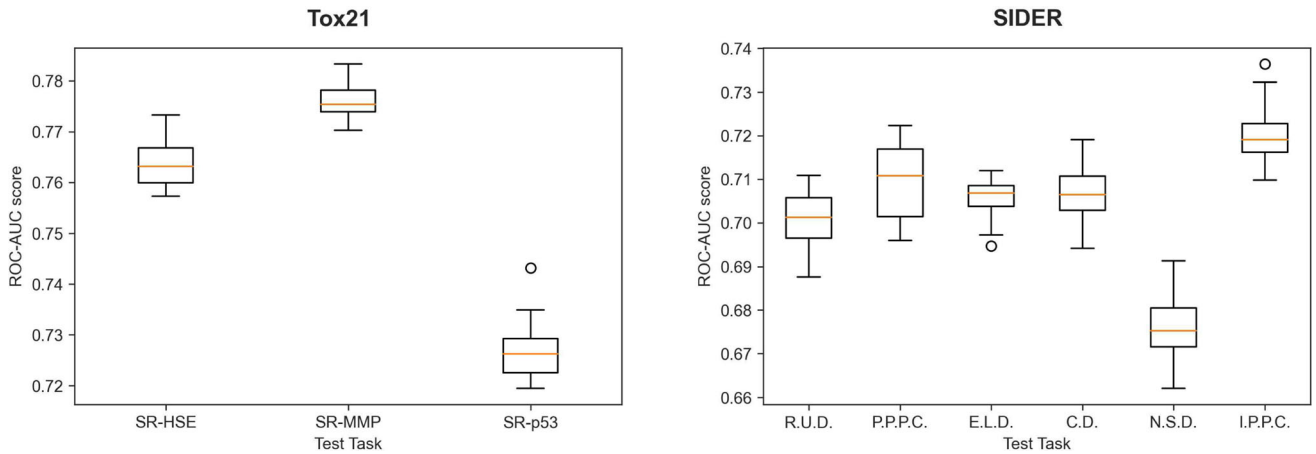

Fig. 6 Distribution of ROC-AUC scores of the proposed model for 20 experiments with 20 random $(5+, 5-)$ support sets

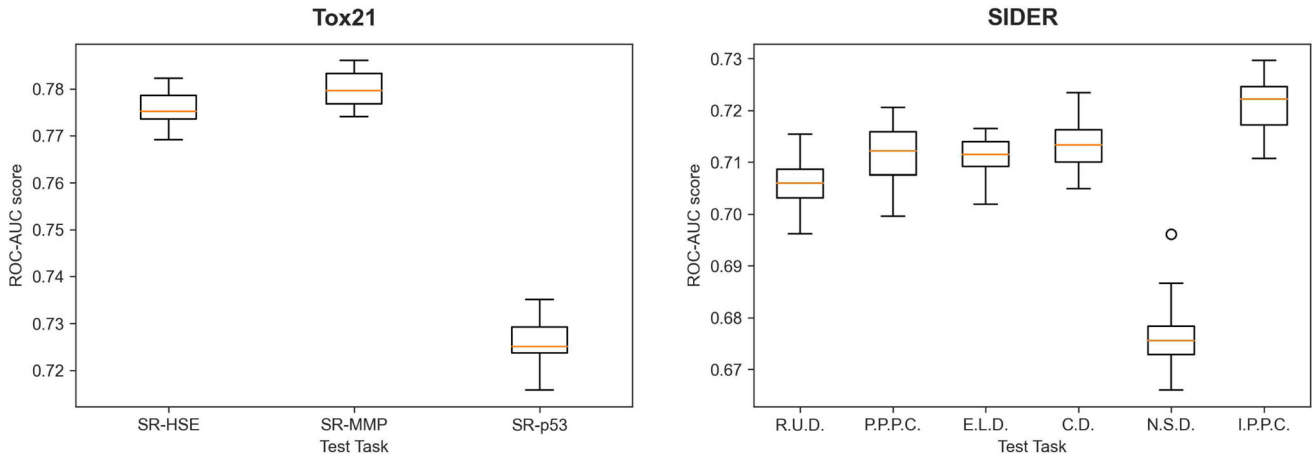

Fig. 7 Distribution of ROC-AUC scores of the proposed model for 20 experiments with 20 random $(10+, 10-)$ support sets

two-module architecture FS-GNNConv, to effectively learn deep representations from graph embeddings. In addition, we demonstrate that the proposed model outperforms different graph-based baselines.

Note that the work presented undertakes standard meta-learning practices (MAML) to iteratively adapt and generalize to new experimental tasks. Here, few-shot learning experiments model the behavior of small molecules in new experimental tasks given just a few samples of these new systems.

This section reports experimental results for few-shot models across a number of tasks on the Tox21 and SIDER datasets.

The results in Table 4 and 5 are the average ROC-AUC scores obtained on 20 experiments with 20 different $(5+, 5-)$ and $(10+, 10-)$ random support sets, respectively.

The results in Table 4 confirm that the proposed model outperforms the best baseline method on Tox21 for all 3 test tasks and for 4 test tasks on SIDER. For 5-shot experiments, we observe an average overall improvement on Tox21 of $+11.01\%$.

Table 5 reports analogous results for the 10-shot experiment. In this case, the proposed model outperforms the best baseline method on Tox21 for all 3 test tasks and for 4 test tasks on SIDER. We also observe an average overall improvement on Tox21 of $+11.37\%$ and $+0.53\%$ on SIDER.

A graphical representation of these results is shown in Figs. 6 and 7. For both datasets, SIDER and Tox21, there are clear differences in performance between tasks, suggesting that the model generalizes better to some tasks than others in the test data. Due to a lower amount of tasks and greater number of samples per task, the model performs better on Tox21 showing lower variances and greater ROC-AUC scores.

For the 5-shot and 10-shot experiments, the standard deviations indicate the lower variances of the proposed model when compared with the graph-based baselines.

This translates into a more stable performance that provides more robust results.

It is clear that the baseline methods do not present a stable performance on a number of tasks. Simply put, they may generalize well for one task but perform poorly for most tasks.

In this scenario, most few-shot models struggle to deal with larger support set sizes and show a more robust improvement in the presence of less data. On that account, we report superior performance with $(5+, 5-)$ random support sets for most baseline methods. However, the same does not apply to the proposed model. This can be explained by the convolutional component, which adds a significant boost with larger support sets $(10+, 10-)$.

Since SIDER has more tasks than Tox21, it is difficult to achieve great overall performance for an extensive set of separate tasks. In contrast, due to the larger size of Tox21, the proposed model performs better by exploiting the generalization capabilities of the CNN module.

Nonetheless, we still experience some underlying limitations of few-shot models. As shown in transfer learning experiments presented in Sect. 5.4, few-shot models struggle to classify completely different tasks with little or no degree of similarity between them. Experimentation also demonstrates that few-shot methods find it difficult to generalize to unrelated tasks regardless of the direction from which we transfer the knowledge. These results indicate that there is a long path to achieve broader generalization and predict completely unrelated tasks of a disjoint system.

Finally, in Sect. 5.5, we explore t-SNE visualizations to visually compare graph embeddings and deep representations. For each dataset, we show the differences between t-SNE cluster plots obtained by the proposed model and the graph-based baselines. It has been reported that deep representations mapped to the reduced space perform better in discriminating both types of molecules (positive or negative) for each molecular property.

All documentation and code scripts to reproduce the results are available to facilitate further experimentation.

Table 6 Mean ROC-AUC scores of models trained on Tox21 to predict SIDER tasks

SIDER from Tox21			
GIN	GCN	GraphSAGE	FS-GNNConv
49.43 ± 0.07	52.90 ± 0.38	53.02 ± 0.42	50.54 ± 0.18

Table 7 Mean ROC-AUC scores of models trained on SIDER to predict Tox21 tasks

Tox21 from SIDER			
GIN	GCN	GraphSAGE	FS-GNNConv
50.38 ± 0.23	52.97 ± 0.76	52.26 ± 0.29	51.71 ± 0.31

5.4 Case study: transfer learning with few-shot models

Previous experiments report the ability of few-shot learning to transfer information from one training task to rather similar testing tasks. To complement this work, we also test whether the proposed model is able to transfer knowledge from Tox21 to predict new tasks in the SIDER benchmark, and vice versa. In practice, the goal is to learn a model trained to predict the toxicity on different nuclear receptors (NR) and stress response pathways (SR) (Tox21) and use it to predict the side effects on real patients over 27 organ systems (SIDER). Conversely, we aim to predict the toxicity on Tox21 from a model trained to predict side effects on SIDER.

Consistent experimentation is conducted to evaluate whether few-shot models are able to generalize to unrelated tasks when provided with very little or no supervised information similar or closely related to the test data.

From this perspective, we assess the ability of few-shot learning to generalize by transferring knowledge between two broadly distinct data repositories. In Table 6, we report the mean ROC-AUC scores for all 27 SIDER tasks for models trained on Tox21. This experiment is repeated 20 times with 20 different $(5+, 5-)$ random support sets. The reverse experiment with the same settings is reported in Table 7.

We conclude that none of the few-shot models reported achieve acceptable performance for rather distinct data collections, attesting to the lack of predictive power for unrelated tasks.

5.5 Case study: t-SNE visualization of graph embeddings and deep representations

Well-understood methods such as principal component analysis (PCA) map high-dimensional data into low-dimensional feature spaces by retaining the global structure to preserve data variance globally across the entire dataset.

t-distributed stochastic neighborhood embedding (t-SNE) works differently by observing closely located datapoints. To this end, t-SNE computes a metric to measure the distance between datapoints and a given number of neighbors and models this relation by a t-distributed

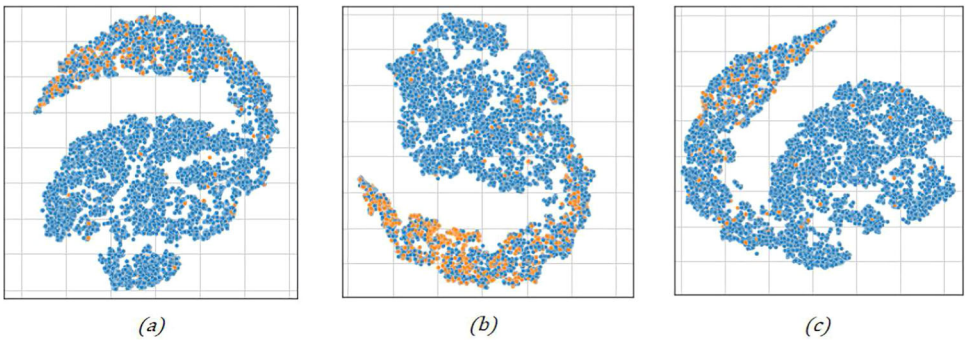

Fig. 8 t-SNE visualizations of deep representations h_{conv} generated by FS-GNNConv for the Tox21 dataset for $(5+, 5-)$ random support sets. The orange dots represent positive labels and the blue points the negative labels. SR-HSE, SR-MMP and SR-p53 tasks are described by plots (**a**), (**b**) and (**c**), respectively

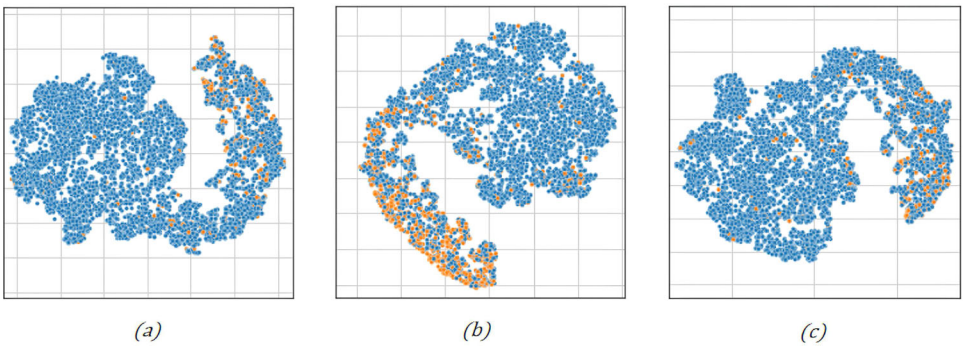

Fig. 9 t-SNE visualizations of deep representations h_{conv} generated by FS-GNNConv for the Tox21 dataset for $(10+, 10-)$ random support sets. The orange dots represent positive labels and the blue points the negative labels. SR-HSE, SR-MMP and SR-p53 tasks are described by plots (**a**), (**b**) and (**c**), respectively

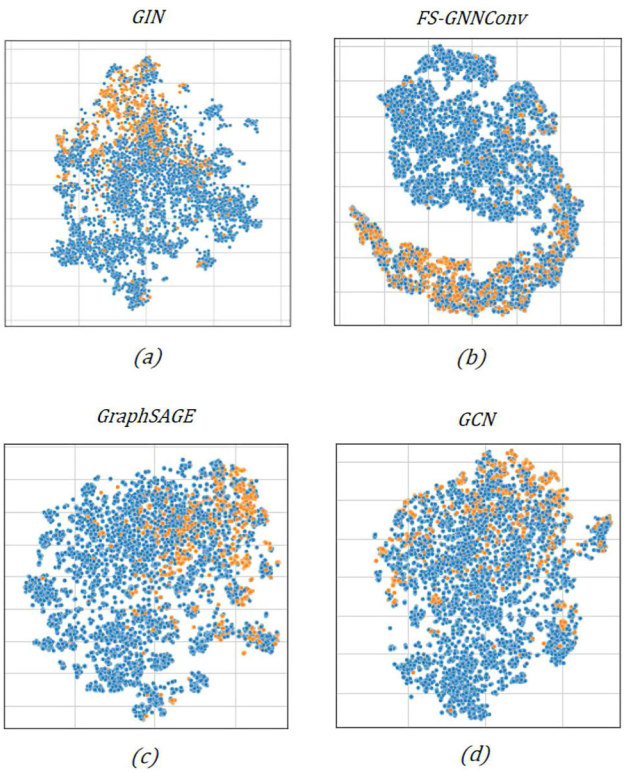

Fig. 10 t-SNE visualizations of graph embeddings generated by GraphSAGE, GIN, GCN and deep representations for the Tox21 SR-MMP task for $(5+, 5-)$ random support sets. The orange dots represent positive labels and the blue points the negative labels

distribution. Then, it tries to find an optimal embedding such that graphs in the original n-dimensional space are mapped to close locations in a low-dimensional space [54].

t-SNE works remarkably well to retain the local structure so that clusters of graph embeddings and deep representations in the reduced space are interpretable as molecules that were also similar in the high-dimensional space.

Deep representations for different molecular property tasks on the Tox21 dataset: {SR-HSE, SR-MMP, SR-p53} using t-SNE visualizations are shown in Figs. 8 and 9. In Fig. 10, we compare t-SNE visualizations of graph embeddings and deep representations. Blue dots denote negative samples for each test task. The orange dots represent positive samples.

One important feature of t-SNE is the perplexity parameter. Perplexity balances the importance of local and global structure in the plotted result. For lower values, we focus on local aspects, while large perplexity values denote a global sense of geometry in high-dimensional spaces. To balance both views, we fine-tuned the perplexity parameter and fixed a value of 30.

In Figs. 8 and 9, it is noticeable that our model performs well in discriminating both types of molecules (positive or negative) since positive datapoints are found closer to each other in the reduced space. For cases (a) and (c), we can see clusters of orange dots progressively separating from blue datapoints. In case (b), most orange points are separated

from the blue datapoints located in the upper region, denoting two well-defined clusters of molecules.

It is clear that our model achieves better performance when discriminating between positive and negative samples than the other baseline methods. In Fig. 10, GIN, GCN and GraphSAGE show sparsely located positive and negative samples, making it difficult to identify well-defined groups of molecules. Conversely, deep representations obtained by FS-GNNConv place positive samples on the bottom left corner to form clusters of molecules closely related to each other.

In addition, we observe local dependencies separating both positive and negative samples. In a broader view, an elongated shape is visible, which might indicate the existence of complex interaction patterns shared by deep representations expressing a global connectivity among molecules.

6 Conclusion

The main goal of this paper is to tackle the challenge of low-data in few-shot molecular property prediction. We systematically address this issue by introducing an architecture to effectively learn deep representations from graph embeddings. In this work, we demonstrate that the proposed model outperforms different graph-based methods.

Small data networks (Tox21 and SIDER) simulate an environment favorable for low-data learning where few-shot models unequivocally outperform simple deep learning approaches. Both benchmarks include high-level measurements of toxicity and side-effect frequency, making predictions volatile and highly uncertain. This behavior makes few-shot learning results particularly interesting and gives a strong indication of superior performances in small biological datasets.

In this work, we proposed a new few-shot two-module architecture, called FS-GNNConv, to address the low-data problem of molecular property prediction. A GNN module encodes the topological structure of molecular graphs as a set of node (atoms) and edge features (chemical bonds). The resulting graphs are then converted into embedded representations. By exploiting the rich information of these embedded descriptors, a CNN propagates deep representations across convolutional layers to generalize to new chemical properties and unseen classes of molecular scaffolds.

A meta-learning framework for optimizing a two-module network across tasks was developed promoting quickly adaptation to new molecular properties on few-shot data. Analysis of the experimental results demonstrated the predictive power and robustness of the proposed model over standard graph-based methods on multi-property prediction data. The results showed that FS-GNNConv takes a step forward to generalize to new experimental tasks, marginally identical to the tasks found in training.

As shown in Sect. 5.3, the proposed model outperforms the best baseline method presented for the majority of test tasks with an average overall improvement of $+11.37\%$ and $+0.53\%$ for Tox21 and SIDER, respectively (for $(10+, 10-)$ random support sets). We posit that the novel proposed framework fully explores the potential of graph-level embeddings to generalize to new molecular properties in contrast with the other GNN competitors.

Future work includes the exploration of few-shot models to generalize to unrelated drug discovery tasks with no degree of structural similarity among molecules. It would also be a promising direction to extend the ideas to regression tasks encouraging predictions to a larger spectrum of molecular properties. We believe that this study demonstrates that starting with few-shot models as powerful non-trivial predictors can help to improve broader generalization in the molecular property prediction problem.

Appendix

This appendix section provides some details regarding model training and optimization.

Meta-learning algorithm

A meta-learning framework optimizes the model parameters across tasks. The algorithm for model training and optimization is displayed below.

Algorithm 1 : FS-GNNConv

Require: Support data: (S_t, Y_t), Test data: (Q_t, Y_t'), $\alpha, \beta, \alpha^*, \beta^*$: update step sizes (e.g. learning rate)
$\theta \leftarrow$ Pre-trained GNN
while ***not done*** **do**
  Sample a batch of tasks $t \sim \rho(T)$
  for all t **do**
    Sample k examples $\{G_{S_{t_1}}, G_{S_{t_2}}, ...G_{S_{t_k}}\} \in S_t$
    for $i = 1$ to k **do**
      $y_{t_i}, h_{G_{S_{t_i}}} = \text{GNN}(G_{S_{t_i}}, \theta)$
    end for
    $\mathcal{L}_t^{gnn} \leftarrow \{y_{t_1}, y_{t_2}, ..., y_{t_k}\}$
    for $i = 1$ to k **do**
      $y_{t_i}, h_{conv_{t_i}} = \text{CNN}(h_{G_{S_{t_i}}}, \theta^*)$
    end for
    $\mathcal{L}_t^{conv} \leftarrow \{y_{t_1}, y_{t_2}, ..., y_{t_k}\}$
    $\theta' = \theta - \alpha \nabla \mathcal{L}_t^{gnn}$
    $\theta^{*'} = \theta^* - \alpha^* \nabla \mathcal{L}_t^{conv}$
    Sample n examples $\{G_{Q_{t_1}}, G_{Q_{t_2}}, ...G_{Q_{t_n}}\} \in Q_t$
    for $j = 1$ to n **do**
      $y'_{t_j}, h_{G_{Q_{t_j}}} = \text{GNN}(G_{Q_{t_j}}, \theta')$
    end for
    $\mathcal{L}_t^{gnn'} \leftarrow \{y'_{t_1}, y'_{t_2}, ..., y'_{t_n}\}$
    for $j = 1$ to n **do**
      $y'_{t_j}, h_{conv_{t_j}} = \text{CNN}(h_{G_{Q_{t_j}}}, \theta^{*'})$
    end for
    $\mathcal{L}_t^{conv'} \leftarrow \{y'_{t_1}, y'_{t_2}, ..., y'_{t_n}\}$
  end for
  $\theta \leftarrow \theta\beta\nabla_\theta \sum_{t\sim\rho(T)} \mathcal{L}_i^{gnn'}$
  $\theta^* \leftarrow \theta^*\beta^*\nabla_{\theta^*} \sum_{t\sim\rho(T)} \mathcal{L}_i^{cnn'}$
end while

In this case, the GNN is denoted by $\text{GNN}(\theta)$ and the CNN by $\text{CNN}(\theta^*)$. First, for each k-shot task t, k support set samples are fed into the network to compute the support losses $\mathcal{L}_t^{\text{gnn}}$ and $\mathcal{L}_t^{\text{conv}}$. Subsequently, all support losses are used to update the model parameters $\theta \rightarrow \theta'$, $\theta^* \rightarrow \theta^{*'}$. Finally, n examples are sampled from a query set to further evaluate the model and compute the query losses $\mathcal{L}_t^{\text{gnn}'}$ and $\mathcal{L}_t^{\text{conv}'}$. In meta-testing, the updated model generalizes to new examples to assist the prediction of a molecular property.

Details of model training

As stated before, training is performed across several learning episodes. Each episode randomly samples a support set of size $(5+, 5-)$ or $(10+, 10-)$ and a query set of size 128. Models provided for comparison were trained across $(n_{TRAIN} \times epochs)$ episodes with n_{TRAIN} number of training tasks and *epochs* number of epochs. In most cases, the models stopped improving significantly after 500 epochs.

In meta-testing, ROC-AUC scores are evaluated for each test task separately. In the same way as in training, a support set is randomly sampled and the model is evaluated in the remaining data points for the task at hand. This process is repeated 20 times for 20 different support sets to report the average ROC-AUC scores for each test task.

In this work, we do not focus on hyperparameter optimization, especially for model baselines. Consequently, we did not put a great effort into optimizing model hyperparameters, leaving this task for future work. More specifically, we consider a learning rate of $1e{-}4$ and an update step of 5 for training and 10 for testing.

Data are split into test and training tasks: Tox21 is split into 9 tasks for training and 3 for testing for a total of 12 tasks; SIDER is split into 21 tasks for training and 6 for testing for a total of 27 tasks.

Task details

Tox21 Task Details: {NR-AR, NR-ARLBD, NR-AhR, NR-Aromatase, NR-ER, NR-ER-LBD, NRPPAR-gamma, SR-ARE, SR-ATAD5} were used for training. Tasks {SR-HSE, SR-MMP, and SR-p53} were used for evaluation.

SIDER Task Details: {H.D.: "Hepatobiliary disorders", M.N.D.: "Metabolism and nutrition disorders", P.I: "Product issues", E.D.: "Eye disorders", I.M.C.T.D.: "Investigations, musculoskeletal and connective tissue disorders", G.D.: "Gastrointestinal disorders", S.C." "Social circumstances", I.S.D: "Immune system disorders", R.S.B.D.: "Reproductive system and breast disorders", N.B.M.U.: "Neoplasms benign, malignant and unspecified (incl cysts and polyps)", G.D.A.C: "General disorders and administration site conditions", E.D.: "Endocrine disorders", S.M.P: "Surgical and medical procedures", V.D: "Vascular disorders", B.L.S.D: "Blood and lymphatic system disorders", S.S.T.D.: "Skin and subcutaneous tissue disorders", C.F.G.T.D.: "Congenital, familial and genetic disorders", "I.I.": "Infections and infestations", R.T.M.D.: "Respiratory, thoracic and mediastinal disorders", P.D.: "Psychiatric disorders"} were used for training. Tasks {R.U.D.: "Renal and urinary disorders", P.P.P.C.: "Pregnancy, puerperium and perinatal conditions", E.L.D.: "Ear and labyrinth disorders", C.D.: "Cardiac disorders", N.S.D.: "Nervous system disorders", I.P.P.C.: "Injury, poisoning and procedural complications"} were used for evaluation.

Author contributions All authors contributed to the study conception and design. Methodology, material preparation, data collection and analysis were performed by Luis Torres. The first draft of the manuscript was written by Luis Torres, and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Funding Open access funding provided by FCT|FCCN (b-on). This work is funded by the FCT—Foundation for Science and Technology, I.P./MCTES through national funds (PIDDAC), within the scope of CISUC R &D Unit - UIDB/00326/2020 or project code UIDP/00326/2020.

Data availability The datasets analyzed during the current study were obtained from the repository http://snap.stanford.edu/gnn-pretrain/data/ of Hu et al. [27].

Code availability All the documentation and code scripts to reproduce the results are available in the repository, https://github.com/ltorres97/FS-GNNConv, to facilitate further experimentation.

Declarations

Conflict of interest The authors have no financial or proprietary interests in any material discussed in this article.

References

1. Leelananda SP, Lindert S (2016) Computational methods in drug discovery. Beilstein J Org Chem 12:2694–2718. https://doi.org/10.3762/bjoc.12.267
2. Keseru GM, Makara GM (2006) Hit discovery and hit-to-lead approaches. Drug Discov Today 11(15–16):741–748. https://doi.org/10.1016/j.drudis.2006.06.016
3. Rong Y, Bian Y, Xu T, Xie W, Wei Y, Huang W, Huang J (2020) Self-supervised graph transformer on large-scale molecular data. In: Advances in neural information processing systems, vol 2020-December. https://doi.org/10.48550/arXiv.2007.02835
4. Hughes JP, Rees SS, Kalindjian SB, Philpott KL (2011) Principles of early drug discovery. Br J Pharmacol 162(6):1239–1249. https://doi.org/10.1111/j.1476-5381.2010.01127.x
5. Paul D, Sanap G, Shenoy S, Kalyane D, Kalia K, Tekade RK (2021) Artificial intelligence in drug discovery and development. Drug Discov Today 26(1):80–93. https://doi.org/10.1016/j.drudis.2020.10.010
6. Gawehn E, Hiss JA, Schneider G (2016) Deep learning in drug discovery. Mol Inf 35(1):3–14. https://doi.org/10.1002/minf.201501008
7. Wu Z, Ramsundar B, Feinberg EN, Gomes J, Geniesse C, Pappu AS, Leswing K, Pande V (2018) MoleculeNet: a benchmark for molecular machine learning. Chem Sci. https://doi.org/10.1039/c7sc02664a
8. Waring MJ, Arrowsmith J, Leach AR, Leeson PD, Mandrell S, Owen RM, Pairaudeau G, Pennie WD, Pickett SD, Wang J, Wallace O, Weir A (2015) An analysis of the attrition of drug candidates from four major pharmaceutical companies. Nat Rev Drug Discov 14(7):475–486. https://doi.org/10.1038/nrd4609
9. Abbasi K, Poso A, Ghasemi J, Amanlou M, Masoudi-Nejad A (2019) Deep transferable compound representation across domains and tasks for low data drug discovery. J Chem Inf Model 59:4528–4539. https://doi.org/10.1021/acs.jcim.9b00626
10. Feinberg EN, Joshi E, Pande VS, Cheng AC (2020) Improvement in ADMET prediction with multitask deep featurization. J Med Chem 63(16):8835–8848. https://doi.org/10.1021/acs.jmedchem.9b02187
11. Lei T, Li Y, Song Y, Li D, Sun H, Hou T (2016) ADMET evaluation in drug discovery: 15. Accurate prediction of rat oral acute toxicity using relevance vector machine and consensus modeling. J Cheminform 8(1):6. https://doi.org/10.1186/s13321-016-0117-7
12. Zhang Y, Wang L, Wang X, Zhang C, Ge J, Tang J, Su A, Duan H (2021) Data augmentation and transfer learning strategies for reaction prediction in low chemical data regimes. Organ Chem Front 8:1415–1423. https://doi.org/10.1039/d0qo01636e

13. Cai C, Wang S, Xu Y, Zhang W, Tang K, Ouyang Q, Lai L, Pei J (2020) Transfer learning for drug discovery. J Med Chem. https://doi.org/10.1021/acs.jmedchem.9b02147
14. Deng Y, Qiu Y, Xu X, Liu S, Zhang Z, Zhu S, Zhang W (2022) Meta-DDIE: predicting drug-drug interaction events with few-shot learning. Brief Bioinform. https://doi.org/10.1093/bib/bbab514
15. Ma J, Fong SH, Luo Y, Bakkenist CJ, Shen JP, Mourragui S, Wessels LFA, Hafner M, Sharan R, Peng J, Ideker T (2021) Few-shot learning creates predictive models of drug response that translate from high-throughput screens to individual patients. Nat Cancer 2:233–244. https://doi.org/10.1038/s43018-020-00169-2
16. Kearnes S, McCloskey K, Berndl M, Pande V, Riley P (2016) Molecular graph convolutions: moving beyond fingerprints. J Comput Aided Mol Design. https://doi.org/10.1007/s10822-016-9938-8
17. Yang K, Swanson K, Jin W, Coley C, Eiden P, Gao H, Guzman-Perez A, Hopper T, Kelley B, Mathea M, Palmer A, Settels V, Jaakkola T, Jensen K, Barzilay R (2019) Analyzing learned molecular representations for property prediction. J Chem Inf Model 59(8):3370–3388. https://doi.org/10.1021/acs.jcim.9b00237. arXiv:1904.01561
18. Coley CW, Barzilay R, Green WH, Jaakkola TS, Jensen KF (2017) Convolutional embedding of attributed molecular graphs for physical property prediction. J Chem Inf Model 57(8):1757–1772. https://doi.org/10.1021/acs.jcim.6b00601
19. Weininger D (1988) SMILES, a chemical language and information system: 1: introduction to methodology and encoding rules. J Chem Inform Comput Sci. https://doi.org/10.1021/ci00057a005
20. Wang Y, Abuduweili A, Dou D (2021) Property-aware adaptive relation networks for molecular property prediction. NeurIPS (NeurIPS) 1–17. https://doi.org/10.48550/arXiv.2107.07994. arXiv:2107.07994
21. Ding K, Wang J, Li J, Shu K, Liu C, Liu H (2020) Graph prototypical networks for few-shot learning on attributed networks. In: International conference on information and knowledge management, proceedings, pp 295–304. https://doi.org/10.1145/3340531.3411922
22. Wang Y, Yao Q, Kwok JT, Ni LM (2020) Generalizing from a few examples: a survey on few-shot learning. ACM Comput Surv https://doi.org/10.1145/3386252. arXiv:1904.05046
23. Sun Q, Liu Y, Chua TS, Schiele B (2019) Meta-transfer learning for few-shot learning. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition, pp 403–412. https://doi.org/10.1109/CVPR.2019.00049
24. Fei-Fei L, Fergus R, Perona P (2006) One-shot learning of object categories. IEEE Trans Pattern Anal Mach Intell. https://doi.org/10.1109/TPAMI.2006.79
25. Altae-Tran H, Ramsundar B, Pappu AS, Pande V (2017) Low data drug discovery with one-shot learning. ACS Cent Sci 3(4):283–293. https://doi.org/10.1021/acscentsci.6b00367. arXiv:1611.03199
26. Grover A, Leskovec J (2016) Node2vec: scalable feature learning for networks. vol 13-17-August-2016. https://doi.org/10.1145/2939672.2939754
27. Hu W, Liu B, Gomes J, Zitnik M, Liang P, Pande V, Leskovec J (2020) Strategies for pre-training graph neural networks. CoRR. https://doi.org/10.48550/ARXIV.1905.12265, arXiv:abs/1905.12265
28. Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: 5th International conference on learning representations. In: ICLR 2017—conference track proceedings. https://doi.org/10.48550/arXiv.1609.02907
29. Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. In: Advances in neural information processing systems, pp 3844–3852. https://doi.org/10.48550/arXiv.1606.09375
30. Hamilton WL, Ying R, Leskovec J (2017) Inductive representation learning on large graphs. Advances in neural information processing systems 2017-December, pp 1025–1035. https://doi.org/10.48550/arXiv.1706.02216
31. Xu K, Li C, Tian Y, Sonobe T, Kawarabayashi KI, Jegelka S (2018) Representation learning on graphs with jumping knowledge networks. In: 35th international conference on machine learning, ICML 2018 12, pp 8676–8685. https://doi.org/10.48550/arXiv.1806.03536
32. Veličković P, Casanova A, Liò P, Cucurull G, Romero A, Bengio Y (2018) Graph attention networks. In: International conference on learning representations, ICLR. https://doi.org/10.1007/978-3-031-01587-8_7
33. Xu K, Jegelka S, Hu W, Leskovec J (2019) How powerful are graph neural networks? In: 7th international conference on learning representations, ICLR 2019. https://doi.org/10.48550/arXiv.1810.00826
34. Guo Z, Zhang C, Yu W, Herr J, Wiest O, Jiang M, Chawla NV (2021) Few-shot graph learning for molecular property prediction. In: The web conference 2021—proceedings of the world wide web conference, WWW 2021. https://doi.org/10.1145/3442381.3450112
35. Ståhl N, Falkman G, Karlsson A, Mathiason G, Boström J (2018) Deep convolutional neural networks for the prediction of molecular properties: challenges and opportunities connected to the data. J Integr Bioinform. https://doi.org/10.1515/jib-2018-0065
36. Lecun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444. https://doi.org/10.1038/nature14539
37. Karpathy A (2017) Stanford University CS231n: convolutional neural networks for visual recognition. http://cs231n.stanford.edu/
38. Shi T, Yang Y, Huang S, Chen L, Kuang Z, Heng Y, Mei H (2019) Molecular image-based convolutional neural network for the prediction of ADMET properties. Chemom Intell Lab Syst. https://doi.org/10.1016/j.chemolab.2019.103853
39. Lusci A, Pollastri G, Baldi P (2013) Deep architectures and deep learning in chemoinformatics: the prediction of aqueous solubility for drug-like molecules. J Chem Inf Model 53(7):1563–1575. https://doi.org/10.1021/ci400187y
40. Liu K, Sun X, Jia L, Ma J, Xing H, Wu J, Gao H, Sun Y, Boulnois F, Fan J (2019) Chemi-net: a molecular graph convolutional network for accurate drug property prediction. Int J Mol Sci. https://doi.org/10.3390/ijms20143389
41. Ryu S, Kwon Y, Kim WY (2019) A Bayesian graph convolutional network for reliable prediction of molecular properties with uncertainty quantification. Chem Sci 10(36):8438–8446. https://doi.org/10.1039/c9sc01992h
42. Kim BH, Ye JC (2020) Understanding graph isomorphism network for rs-fMRI functional connectivity analysis. Front Neurosci. https://doi.org/10.3389/fnins.2020.00630
43. Wu Z, Pan S, Chen F, Long G, Zhang C, Yu PS (2021) A comprehensive survey on graph neural networks. IEEE Trans Neural Netw Learn Syst 32(1):4–24. https://doi.org/10.1109/TNNLS.2020.2978386. arXiv:1901.00596
44. Gilmer J, Schoenholz SS, Riley PF, Vinyals O, Dahl GE (2017) Neural message passing for quantum chemistry. In: 34th international conference on machine learning, ICML 2017, vol 3, pp 2053–2070. https://doi.org/10.48550/arXiv.1704.01212
45. Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. In: 32nd international conference on machine learning, ICML 2015, vol 1, pp 448–456. https://doi.org/10.48550/arXiv.1502.03167

46. Arora R, Basu A, Mianjy P, Mukherjee A (2018) Understanding deep neural networks with rectified linear units. In: 6th international conference on learning representations, ICLR 2018—conference track proceedings. https://doi.org/10.48550/arXiv.1611.01491
47. Finn C, Abbeel P, Levine S (2017) Model-agnostic meta-learning for fast adaptation of deep networks. In: 34th international conference on machine learning, ICML 2017, vol 3. https://doi.org/10.48550/arXiv.1703.03400
48. Sun Q, Liu Y, Chua TS, Schiele B (2019) Meta-transfer learning for few-shot learning. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition. IEEE Computer Society, vol 2019-June, pp 403–412. https://doi.org/10.1109/CVPR.2019.00049
49. Duan Y, Andrychowicz M, Stadie B, Ho J, Schneider J, Sutskever I, Abbeel P, Zaremba W (2017) One-shot imitation learning. In: Advances in neural information processing systems, vol. 2017-December, pp 1088–1099. https://doi.org/10.48550/arXiv.1703.07326
50. Mayr A, Klambauer G, Unterthiner T, Hochreiter S (2016) DeepTox: toxicity prediction using deep learning. Front Environ Sci. https://doi.org/10.3389/fenvs.2015.00080
51. Kuhn M, Letunic I, Jensen LJ, Bork P (2016) The SIDER database of drugs and side effects. Nucl Acids Res 44(D1):1075–1079. https://doi.org/10.1093/nar/gkv1075
52. Landrum G (2021) RDKit: open-source cheminformatics software. http://www.rdkit.org/
53. Zhang S, Tong H, Xu J, Maciejewski R (2019) Graph convolutional networks: a comprehensive review. Comput Soc Netw. https://doi.org/10.1186/s40649-019-0069-y
54. Van Der Maaten L, Hinton G (2008) Visualizing data using t-SNE. J Mach Learn Res 9:2579–2605