Machine Learning (2019) 108:1757–1781
https://doi.org/10.1007/s10994-019-05782-6

Attentional multilabel learning over graphs: a message passing approach

Kien Do[1] · **Truyen Tran**[1] · **Thin Nguyen**[1] · **Svetha Venkatesh**[1]

Received: 11 April 2018 / Accepted: 9 January 2019 / Published online: 8 March 2019

Abstract
We address a largely open problem of multilabel classification over graphs. Unlike traditional vector input, a graph has rich variable-size substructures which are related to the labels in some ways. We believe that uncovering these relations might hold the key to classification performance and explainability. We introduce Graph Attention model for Multi-Label learning (GAML), a novel graph neural network that can handle this problem effectively. GAML regards labels as auxiliary nodes and models them in conjunction with the input graph. By applying the neural message passing algorithm and attention mechanism to both the label nodes and the input nodes iteratively, GAML can capture the relations between the labels and the input subgraphs at various resolution scales. Moreover, our model can take advantage of explicit label dependencies. It also scales linearly with the number of labels and graph size thanks to our proposed *hierarchical attention*. We evaluate GAML on an extensive set of experiments with both graph-structured inputs and classical unstructured inputs. The results show that GAML significantly outperforms other competing methods. Importantly, GAML enables intuitive visualizations for better understanding of the label-substructure relations and explanation of the model behaviors.

Keywords Multilabel learning · Graph classification · Graph neural networks · Message passing

Editors: Masashi Sugiyama, Yung-Kyun Noh.

This work is partially supported by the Telstra-Deakin Centre of Excellence in Big Data and Machine Learning.

✉ Kien Do
dkdo@deakin.edu.au

Truyen Tran
truyen.tran@deakin.edu.au

Thin Nguyen
thin.nguyen@deakin.edu.au

Svetha Venkatesh
svetha.venkatesh@deakin.edu.au

[1] Applied AI Institute, Deakin University, Geelong, Australia

1 Introduction

Drug development costs billions of dollars over many years with multiple stages of refinement and trial (Mullard 2014). For this reason, repurposing approved drugs is a critical alternative to the full development cycle, offering a huge saving of money, time, and lives. A canonical task in drug repurposing is to predict the drug effect on multiple related diseases. This can naturally be formulated as a multilabel learning problem. Different from standard domains like text or image, drugs are usually represented as variable size graphs of atoms linked by chemical bonds. The irregularity and complexity of rich graph structures make multilabel learning over molecular graphs very challenging. At the same time, graphs bring about new kinds of information not previously seen in unstructured data, as evidenced in the recent surge of research in graph representation learning (Hamilton et al. 2017b; Zhang et al. 2017). Hence, a new proper treatment of the multilabeling over graphs is needed.

We hypothesize that the key for classification performance and explainability lies in uncovering the relations between labels and subgraphs. Towards this goal, we design a new graph neural network (Scarselli et al. 2009) called GAML (which stands for Graph Attention model for Multi-Label learning). GAML treats all label classes as nodes (termed label nodes) and merges them with other nodes (called input nodes) of an input graph to form a unified label-input graph. In the joint graph, relations between labels and substructures can be effectively captured through the interaction across the label nodes and the input nodes. Specifically, we leverage the message passing algorithm (Pham et al. 2017; Schlichtkrull et al. 2017; Gilmer et al. 2017) to simultaneously update the local substructure at every input node and to propagate the substructure-contained messages from all the input nodes to the label nodes. By using attention (Bahdanau et al. 2014; Xu et al. 2015), each label node can extract the most related substructures to update its own state which will later be used to predict the presence of the corresponding class. Attention also enables insightful visualization which helps explain the prediction. To account for large number of classes and big input graphs, we propose a new type of attention named *hierarchical attention*. Different from the standard approach that calculates the score matrix between every input and label node directly, our attention mechanism uses some intermediate *attentional factors* to save computation. In our model, implicit dependencies among the labels are captured via common attended substructures. However, when explicit dependencies among the labels are available (e.g, through expert knowledge), GAML can easily integrate them by adding links and exchange messages between related label nodes. Moreover, since the node update procedure runs iteratively, our model can learn the label-subgraph (or label-substructure) relations at various resolution scales.

The flexibility and scalability of GAML make it attractive to many real-world problems. In this paper, we focus on two major drug–multitarget prediction problems: predicting *drug–protein binding*, and *drug–cancer response*. In the first problem, a drug is tested against multiple target proteins; and in the second problem, a drug is tested against multiple cancer types. We also evaluate our method on classical vector input which can be seen as a special graph with a singleton node. In both cases, GAML proves to be superior against rival multilabel learning techniques. Finally, to get a clear picture of the learned label-substructure patterns, we generate visualizations using real drug molecules extracted from our datasets.

In summary, our contributions are:

– Proposing a novel neural graph neural network named GAML that addresses an open problem of multilabel classification over graphs. Our model can effectively capture the (multi-way) relations among the labels and the input subgraphs. It can also incorporate explicit label dependencies and is scalable to many labels and big graphs.

– Demonstrating the advantages of GAML through a comprehensive suite of experiments with quantitative evaluation and visualization.

2 Related work

Multilabel classification with label dependencies Most work in multilabel learning focuses on capturing the implicit or explicit label dependencies. One strategy is applying Canonical Correlation Analysis (CCA) to map input and label into a common latent space. Then from this space, the model will reconstruct the target label. Extensions of this approach including both shallow (Li and Guo 2015; Sun et al. 2011) and deep (Yeh et al. 2017) models. For graphical model-based approach, the work in Ghamrawi and McCallum (2005) uses Conditional Random Fields to model the three way relation between every pair of labels i, j and the input $\boldsymbol{x}$ using a feature function $\phi(y_i, y_j, \boldsymbol{x})$. Meanwhile, the work in Guo and Gu (2011) constructs a fully connected cyclic Bayesian Network over labels and perform structure learning on this network. The probability of a label y_i conditioned on the input $\boldsymbol{x}$ and other labels $y_{\neg i}$ is modeled using a logistic regression network. Both methods are computationally expensive and require inexact inference for large number of labels.

To model the joint distribution of labels but still keep computation reasonable, some methods exploit chain rule factorization. The most notable one is Probabilistic Classifier Chain (Dembczynski et al. 2010) which builds a separate binary classifier for each label with input to the model is the combination of the original input and the previously predicted labels. Other methods follow that idea but use recurrent neural networks (Chen et al. 2017; Wang et al. 2016) to learn the relations better. However, the critical issues of these methods are *ordering* and *poor inference* (since the output label at one step depends on the value of the previous predicted labels not their distribution, which is very unstable). Although some tricks like beam search (Wang et al. 2016), or automatic order selection (Chen et al. 2017) have been implemented to improve the results, they can only solve part of the problem.

Expert knowledge about label dependencies represented as trees (Deng et al. 2014) or graphs (Bi and Kwok 2011; Chen et al. 2018) has been exploited for multilabel/multiclass classification. In Chen et al. (2018), the authors build a graph neural network over the predefined label graph. The input vector is copied for every label node and is concatenated to the label embedding vector to form an initial state for that label node. Their method, however, is limited to the vector input only whereas our model directly works on graph input with vector input is the special case.

Multilabel classification with graph inputs Although graph classification has attracted a significant interest in recent years (Takigawa and Mamitsuka 2017), there has been a limited body of work on multilabel graph classification (Kong and Philip 2012). The line of work on image tagging considers multilabel learning over a grid of pixels (Gong et al. 2013; Wang et al. 2016; Wei et al. 2016). However, the standard treatment using CNN usually focuses on attention over feature maps instead of exploiting the structural relations of objects in the original image. A recent work in visual question answering that pushes forward the idea of object graph is Teney et al. (2017), but the QA setting is different from ours. A special case of our multilabel learning over graphs is multilabel learning over set (Pham et al. 2017) where input is a collection of nodes with no explicit links.

Graph neural networks By leveraging the representation power of deep neural networks such as CNN and RNN, a wide range of methods for learning over graphs (Defferrard et al. 2016; Gilmer et al. 2017; Hamilton et al. 2017a; Kipf and Welling 2016a; Li et al. 2016; Niepert

et al. 2016; Pham et al. 2017; Scarselli et al. 2009) has been proposed recently. These methods can be grouped into more general categories such as Spectral Graph based (Bruna et al. 2013; Defferrard et al. 2016; Kipf and Welling 2016a), Message Passing based (Gilmer et al. 2017; Pham et al. 2017; Schlichtkrull et al. 2017), Random Walk based (Grover and Leskovec 2016; Perozzi et al. 2014), Neural Net based (Li et al. 2016). Among them, Message Passing Graph Neural Networks (MPGNNs) are very powerful since they can handle various kinds of graphs including attributed graphs whose edges and nodes both have types. MPGNNs have found many applications in bioinformatics such as drug activity classification (Pham et al. 2018), chemical properties prediction (Gilmer et al. 2017), protein interface prediction (Fout et al. 2017) and drug generation (Jin et al. 2018). However, none of these methods properly handle multilabel classification problems in which modeling multi-way relations among labels and molecular subgraphs is the key factor.

Graph representation learning Many supervised learning problems over graphs (including the multilabel classification problem we are working on) assume precomputed graph embedding. Unsupervised learning methods for graphs (Narayanan et al. 2016; Shervashidze et al. 2011; Yanardag and Vishwanathan 2015) often exploit the common substructures among graphs to ensure that graphs with similar structure will be represented as close points in the embedding vector space. Graph embedding can also be achieved through graph reconstruction or generation. This approach includes VAE/GAN based models (Kipf and Welling 2016b; Simonovsky and Komodakis 2018; Wang et al. 2017) and sequence based models (Li et al. 2018; You et al. 2018).

3 Preliminaries

In this section we provide the mathematical formulation of multilabel classification and the background knowledge about graph neural networks on which GAML is built. For clarity and consistency, we use the following notations throughout the paper (unless being stated explicitly): bold letters denote vectors ($\boldsymbol{x}$ is a vector); capital letters denote matrices (W is a matrix); normal letters denote scalars (s is a scalar); $\{\cdot\}$ denotes a set; $f(\cdot)$ denotes a function f with arguments separated by commas. Table 1 lists most common notations used in the paper.

3.1 Multilabel classification

Multilabel classification is a supervised learning problem in which each input example may be associated with more than one output class. Denote by X the input vector space and $\mathcal{C} \equiv \{1, 2, ..., C\}$ the set of all classes labeled from 1 to C. Multilabel learning estimates a function f that maps X onto the power set of $\mathcal{C}$, written as $f : X \mapsto \mathcal{P}_{\mathcal{C}}$. Because each element of $\mathcal{P}_{\mathcal{C}}$ is a subset of $\mathcal{C}$, it can be represented as a binary vector $\boldsymbol{y}$ of length C with $y_c = 1$ indicates that class c appears in the subset and $y_c = 0$ otherwise ($c = \overline{1, C}$).

3.2 Graph notations

Consider an attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. Each node i is associated with a node feature vector $\boldsymbol{v}_i$ which captures important properties of a node. For example, if the graph $\mathcal{G}$ is a drug molecule (as depicted in Fig. 1), each node is an atom and the node's properties could be its atomic number, its charge, its valance, etc.

Table 1 Notations used in the paper

Notation	Description
$c \in \mathcal{C}$	Index of class, or label c, which is a member of set $\mathcal{C}$
$\mathcal{P}_{\mathcal{C}}$	Powerset of $\mathcal{C}$
t	Inference (message passing) step
y_c	Label at label node c
$\boldsymbol{v}_i$	Input feature vector at node i
$\boldsymbol{x}_i$	State of input node i
$\boldsymbol{m}_i, \boldsymbol{\mu}_i$	Message coming to node i
$\mathcal{N}(i)$	Neighborhood of node i
e_{ij}	Type of edge (i, j)
s_{ij}	Unormalized attention score of node i to node j
$a_{ij}, b_{ij}, \alpha_{ij}, \beta_{ij}$	Attention probabilities of node i to node j
$\boldsymbol{\eta}_i$	Gate at node i
$\mathcal{G}, \mathcal{V}, \mathcal{E}$	Graph $\mathcal{G}$ with set of nodes $\mathcal{V}$ and set of edges $\mathcal{E}$
relu(x)	rectifier linear unit, which is max$(x, 0)$
$\boldsymbol{l}_c$	State of label node c
y_c	Output at node c
f, g	Element-wise nonlinear function

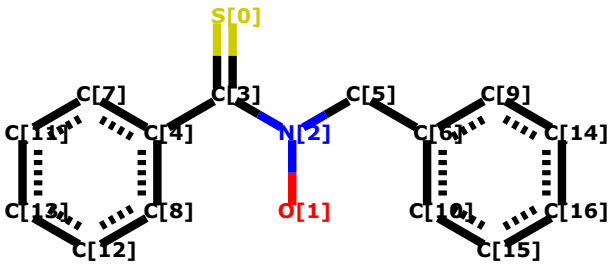


Fig. 1 A drug molecule (PubChem SID = 502937) represented as an attributed graph. This graph has 17 nodes numbered from 0 to 16 corresponding to 17 atoms. Nodes are characterized by atom types (sulfur, oxygen, nitrogen, carbon) and edges are specified by bond types (single, double, aromatic)

In our current work, we only use the atomic number information. Thus, $\boldsymbol{v}_i$ is the embedded vector with respect to that atom type. Similar to nodes, each edge (i, j) is also associated with an edge type e_{ij} (for molecules, it is a bond type).

3.3 Message passing graph neural network

Let $\boldsymbol{x}_i$ be the state of node i and $\mathcal{N}(i) = \{j \mid (i, j) \in \mathcal{E}\}$ denote the neighborhood of node i. In a message passing graph neural network (Gilmer et al. 2017; Pham et al. 2017; Scarselli et al. 2009), a node uses information from its neighbors to update its own state as follows:

$$\boldsymbol{x}_i^t = f\left(\boldsymbol{x}_i^{t-1}, \left\{\left(\boldsymbol{x}_j^{t-1}, e_{ij}\right)\right\}_{j \in \mathcal{N}(i)}\right) \tag{1}$$

where t denotes the update step; and $f(\cdot)$ is a non-linear function (e.g., a multi-layer perceptron (MLP)). At $t = 0$, we set $\boldsymbol{x}_i^0 = \boldsymbol{v}_i$.

Equation (1) is generic for most graph neural network models. In practice, it can be divided into two steps: *message aggregation* and *state update*. In the *message aggregation* step, we combine multiple messages sent to node i into a single message vector $\boldsymbol{m}_i$:

$$\boldsymbol{m}_i^t = g^{\mathrm{a}}\left(\boldsymbol{x}_i^{t-1}, \left\{\left(\boldsymbol{x}_j^{t-1}, e_{ij}\right)\right\}_{j\in\mathcal{N}(i)}\right) \tag{2}$$

where $g^{\mathrm{a}}(\cdot)$ can be an attention (Bahdanau et al. 2014; Xu et al. 2015) or a pooling architecture. For example, the message aggregated using mean pooling has the following formula:

$$\boldsymbol{m}_i^t = \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)} W_{e_{ij}}\boldsymbol{x}_j^{t-1} \tag{3}$$

where $|\mathcal{N}(i)|$ is the number of neighbor nodes of node i; $W_{\boldsymbol{e}}$ is a projection matrix which corresponds to the edge type $\boldsymbol{e}_{ij}$. Despite of simplicity, Eq. (3) has shown to be able to encode graph structures in several message passing models (Gilmer et al. 2017; Pham et al. 2017; Schlichtkrull et al. 2017).

During the *state update* step, the node state is updated as follows:

$$\boldsymbol{x}_i^t = g^{\mathrm{u}}\left(\boldsymbol{x}_i^{t-1}, \boldsymbol{m}_i^t\right) \tag{4}$$

where $g^{\mathrm{u}}(\cdot)$ can be any type of deep neural networks such as MLP (Kipf and Welling 2016a; Hamilton et al. 2017a), RNN (Scarselli et al. 2009), GRU (Li et al. 2016) or Highway Network (Pham et al. 2017). In our model, we use Highway Network (Srivastava et al. 2015) for $g^{\mathrm{u}}(\cdot)$ as it has been shown to be effective for long range dependencies thanks to its skip-connection and gating mechanism. As a result, Eq. (4) now becomes:

$$\boldsymbol{x}_i^t = \left(1 - \boldsymbol{\eta}_i^t\right) \odot \boldsymbol{x}_i^{t-1} + \boldsymbol{\eta}_i^t \odot \hat{\boldsymbol{x}}_i^t \tag{5}$$

where $\boldsymbol{\eta}_i^t \in (\mathbf{0}, \mathbf{1})$ and $\hat{\boldsymbol{x}_i}^t$ are the gate vector and the non-linear candidate vector of node i at time t, respectively; $\odot$ is the element-wise product. The formulas of $\boldsymbol{\eta}^t$ and $\hat{\boldsymbol{x}}^t$ are provided below:

$$\boldsymbol{\eta}_i^t = \mathrm{sigmoid}\left(W_\eta \boldsymbol{x}_i^{t-1} + U_\eta \boldsymbol{m}_i^t\right) \tag{6}$$

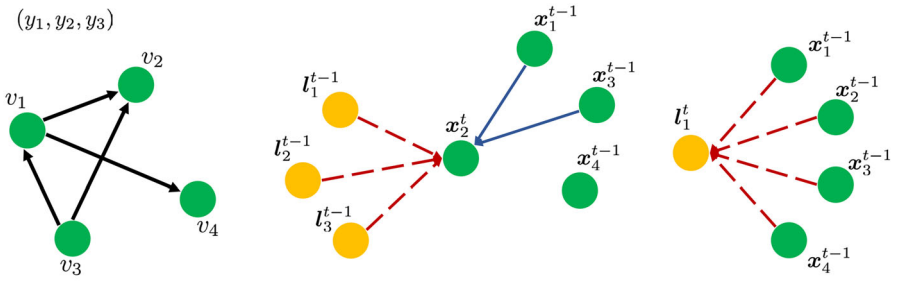
$$\hat{\boldsymbol{x}}_i^t = \mathrm{relu}\left(W_x \boldsymbol{x}_i^{t-1} + U_x \boldsymbol{m}_i^t\right) \tag{7}$$

where W_η, $W_x U_\eta$, U_x are parameters which can be different or shared among layers. During experiments, we observed that models with parameter sharing run faster but still provide comparable results. Hence, we applied this sharing scheme to our model. We abstract Eqs. (4-7) into:

$$\boldsymbol{x}_i^t = \mathrm{Highway}\left(\boldsymbol{x}_i^{t-1}, \boldsymbol{m}_i^t\right) \tag{8}$$

After T steps of message passing, $\boldsymbol{x}_i^T$ would capture the graph substructure centered at node i with radius T. The graph summary vector (also called graph representation vector) $\boldsymbol{x}_{\mathcal{G}}$ is the combination of the state vector of all nodes in the graph at step T. In the simplest form, $\boldsymbol{x}_{\mathcal{G}}$ is the average of $\left\{\boldsymbol{x}_i^T\right\}_{i\in\mathcal{V}}$, as follows:

$$\boldsymbol{x}_{\mathcal{G}} = \frac{1}{|\mathcal{V}|}\sum_{i\in\mathcal{V}} \boldsymbol{x}_i^T$$



(a) A input graph with 4 nodes and 3 labels **(b)** Input node update **(c)** Label node update

Fig. 2 Message passing in the joint graph of input nodes and label nodes. In (**b**, **c**), dash red link indicates message passing with attention while blue solid link indicates message passing with mean pooling (Color figure online)

4 Method

In this section, we present our main contribution—**G**raph **A**ttention model for **M**ulti-**L**abel learning (GAML).

4.1 Multilabel classification over graphs

We generalize the definition of multilabel classification in Sect. 3.1 to the situation in which inputs are graphs by considering a problem of learning a function $f : X_{\mathcal{G}} \mapsto \mathcal{P}_{\mathcal{C}}$ where $X_{\mathcal{G}}$ is the space of input graph representation vectors. We argue that in order to perform well on this task, two types of relation must be captured: those *within the label set* and those *between the label set and input subgraphs*.

For an input graph $\mathcal{G}$, we consider all the C classes as auxiliary nodes (called *label nodes*) alongside $|\mathcal{V}|$ existing nodes of the input graph $\mathcal{G}$. Each label node c connects to all input nodes and has the initial state $\boldsymbol{l}_c^0 \in \mathbb{R}^{d_l}$ which is the embedding of class c to a vector space. On the other hand, each input node i also connects to all label nodes. It results in a joint graph of $C + |\mathcal{V}|$ nodes, which naturally lends itself to the message passing scheme in the graph neural network presented in Sect. 3.3. The idea is that by iteratively updating the states of input and label nodes using message passing, complex label-label and label-substructure dependencies emerge. See Fig. 2 for an illustration.

4.1.1 Input node update

Since an input node i connects to its neighbor nodes $j \in \mathcal{N}(i)$ and all the label nodes $c \in \overline{1, C}$, the message passing update of the input node i at step t is formulated as follows:

$$\boldsymbol{x}_i^t = f\left(\boldsymbol{x}_i^{t-1}, \left\{\left(\boldsymbol{x}_j^{t-1}, \boldsymbol{e}_{ij}\right)\right\}_{j\in\mathcal{N}(i)}, \left\{\boldsymbol{l}_c^{t-1}\right\}_{c\in\overline{1,C}}\right) \tag{9}$$

Note that Eq. (9) is derived from Eq. (1) with the introduction of new arguments $\left\{\boldsymbol{l}_c^{t-1}\right\}_{c\in\overline{1,C}}$.

There are two types of message sent to the input node i. One contains structure information from neighbor input nodes and the other contains label-related information from label nodes.

Because these messages have different meanings, they should be aggregated into separate message vectors. In case of neighbor input nodes, we use mean pooling to combine them as similar to Eq. (3):

$$\boldsymbol{\mu}_i^t = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} W_{\boldsymbol{e}_{ij}} \boldsymbol{x}_j^{t-1}$$

However, mean pooling may not be ideal to aggregate labels since it equalizes the importance of each class towards the input node i. To overcome this issue, we use the attention mechanism (Bahdanau et al. 2014; Xu et al. 2015) to compute a weighted sum of all the label nodes as follows:

$$\boldsymbol{m}_i^t = \sum_{c=1}^{C} a_{ic}^t \boldsymbol{l}_c^{t-1} \tag{10}$$

where $a_{ic}^t > 0$, $\sum_{c=1}^{C} a_{ic}^t = 1$ is the attention coefficient from the input node i to a label node c at time t, computed as:

$$s_{ic}^t = \boldsymbol{u}_s^\intercal \tanh\left(W_s \boldsymbol{x}_i^{t-1} + U_s \boldsymbol{l}_c^{t-1} + \boldsymbol{b}_s\right) \tag{11}$$

$$a_{ic}^t = \frac{\exp\left(s_{ic}^t\right)}{\sum_{c'=1}^{C} \exp\left(s_{ic'}^t\right)} \tag{12}$$

The set of all unnormalized attention scores s_{ic}^t in Eq. (11) forms a matrix $S^t \in \mathbb{R}^{|\mathcal{V}| \times C}$, which we will reuse later.

For generality, Eqs. (10–12) are written in a more compact form:

$$\boldsymbol{m}_i^t = \text{Attention}\left(\boldsymbol{x}_i^{t-1}, \{\boldsymbol{l}_c^{t-1}\}_{c \in \overline{1,C}}\right) \tag{13}$$

We call the attention in Eq. (13) *input-to-label* attention.

In the state update phase, the new state $\boldsymbol{x}_i^t$ of the input node i is computed as:

$$\boldsymbol{x}_i^t = \text{Highway}\left(\boldsymbol{x}_i^{t-1}, \left[\boldsymbol{\mu}_i^t, \boldsymbol{m}_i^t\right]\right)$$

where $[\cdot]$ denotes vector concatenation and Highway(.) is defined in Eq. (8).

4.1.2 Label node update

By connecting to every input node, a label node c can receive information about various substructures in the graph $\mathcal{G}$ through multiple steps of message passing. Among these substructures, only a few are related to the class c. Therefore, we use the attention mechanism to extract the most useful substructures for predicting class c and store them in the message vector as follows:

$$\boldsymbol{m}_c^t = \text{Attention}\left(\boldsymbol{l}_c^{t-1}, \left\{\boldsymbol{x}_i^{t-1}\right\}_{i \in \overline{1,|\mathcal{V}|}}\right) \tag{14}$$

where Attention(.) is similar to the function defined in Eq. (13) with the role of input nodes and label nodes swapped. We denote this function *label-to-input* attention. The unnormalized score matrix S^t from Eq. (11) is reused here to save computation and improve consistency. However, the attention coefficients are be normalized over rows instead of columns of S^t, i.e.,

$$a_{ci}^t = \frac{\exp\left(s_{ic}^t\right)}{\sum_{i=1}^{|\mathcal{V}|} \exp\left(s_{ic}^t\right)}$$

Finally, we compute the new state of the label node c using a different Highway Network as:

$$\boldsymbol{l}_c^t = \text{Highway}\left(\boldsymbol{l}_c^{t-1}, \boldsymbol{m}_c^t\right)$$

4.1.3 A priori label dependencies

When explicit label dependencies are available, a label graph can be formed in the same way as the input graph. Messages between label nodes is aggregated using mean-pooling as in Eq. (3):

$$\boldsymbol{\mu}_c^t = \frac{1}{|\mathcal{N}(c)|} \sum_{f \in \mathcal{N}(c)} W_{\boldsymbol{e}_{cf}} \boldsymbol{l}_f^{t-1}$$

The state of the label node c is updated as:

$$\boldsymbol{l}_c^t = \text{Highway}\left(\boldsymbol{l}_c^{t-1}, \left[\boldsymbol{m}_c^t, \boldsymbol{\mu}_c^t\right]\right)$$

4.1.4 Vector input as a special case

In many traditional multilabel classification problems, the input is represented as vector instead of graph. This can be seen as a special case of our model where the input graph $\mathcal{G}$ collapses into a single node $\boldsymbol{x}$. With this observation, the state update of the input node at step t is:

$$\begin{aligned}\boldsymbol{m}^t &= \text{Attention}\left(\boldsymbol{x}^{t-1}, \left\{\boldsymbol{l}_c^{t-1}\right\}_{c \in \overline{1,C}}\right)\\ \boldsymbol{x}^t &= \text{Highway}\left(\boldsymbol{x}^{t-1}, \boldsymbol{m}^t\right)\end{aligned}$$

The state of a label node c is updated as:

$$\boldsymbol{l}_c^t = \text{Highway}\left(\boldsymbol{l}^{t-1}, \boldsymbol{x}^{t-1}\right)$$

4.2 Learning

After T steps of message passing, we pass each class-specific final state vector $\boldsymbol{l}_c^T$ to a multi-layer perceptron (MLP) with sigmoid activation on top to predict the present of class c:

$$o_c = \text{MLP}\left(\boldsymbol{l}_c^T\right)$$

Here the value of o_c is in (0, 1). The MLPs for all classes share the same parameters. For learning, we use a binary cross-entropy loss function which is defined as:

$$\mathcal{L} = \mathbb{E}_{\text{train}}\left(\sum_{c=1}^{C} y_c \log o_c + (1 - y_c)\log(1 - o_c)\right)$$

where $\mathbb{E}_{\text{train}}$ denotes the mean over all training data.

4.3 Scale to big graphs and many labels

When the number of nodes in the input graph ($|\mathcal{V}|$) and the number of classes (C) are large, it becomes expensive to calculated the unnormalized score matrix $S^t \in \mathbb{R}^{|\mathcal{V}|\times C}$ in Eq. (11) for all steps $t = 1, 2, ..., T$. To handle this problem, we propose a new attention technique called *hierarchical attention*. At each layer, we define K ($K \ll \min\{|\mathcal{V}|,\ C\}$) intermediate *attentional factors* between input nodes and label nodes. The input-label attentions are broken down into two steps as follows:

- For *label-to-input* attention, we do *factor-to-input* attention then *label-to-factor* attention.
- For *input-to-label* attention, we do *factor-to-label* attention then *input-to-factor* attention.

Label-to-input message aggregation. More concretely, the *label-to-input* message aggregation in Eq. (10) is replaced by:

$$\boldsymbol{m}_i^t = \sum_{k=1}^{K} a_{ik}^t \boldsymbol{\lambda}_k^{t-1}; \quad \text{for} \quad \boldsymbol{\lambda}_k^{t-1} = \sum_{c=1}^{C} b_{ck}^t \boldsymbol{l}_c^{t-1}$$

where $\boldsymbol{\lambda}_k^{t-1}$ is the kth intermediate factor ($k \in \overline{1, K}$) that aggregates all label nodes; $\boldsymbol{m}_i^t$ is the message to the input node i; a_{ik}^t is factor-to-input attention probability (i.e., $\sum_{k=1}^{K} a_{ik}^t = 1$); and b_{ck}^t is label-to-factor attention probability (i.e., $\sum_{c=1}^{C} b_{ck}^t = 1$).

To compute a_{ik}^t and b_{ck}^t we define two score matrices $S_1^t = \left[s_{1;ik}^t\right] \in \mathbb{R}^{|\mathcal{V}|\times K}$ and $S_2^t = \left[s_{2;ik}^t\right] \in \mathbb{R}^{C\times K}$ as follows:

$$s_{1;ik}^t = \boldsymbol{u}_1^\intercal \tanh(W_1 \boldsymbol{x}_i^{t-1} + \boldsymbol{z}_k^{t-1}) \tag{15}$$

$$\text{and} \quad s_{2;ck}^t = \boldsymbol{u}_2^\intercal \tanh(W_2 \boldsymbol{l}_c^{t-1} + \boldsymbol{z}_k^{t-1}) \tag{16}$$

where $\boldsymbol{u}_1, \boldsymbol{u}_2 \in \mathbb{R}^{d_z}$, $W_1 \in \mathbb{R}^{d_x\times d_z}$, $W_2 \in \mathbb{R}^{d_l\times d_z}$ and $\boldsymbol{z}_k^t \in \mathbb{R}^{d_z}$, ($k = \overline{1, K}$) are parameters. Then factor-to-input attention probability and label-to-factor attention probability are computed as:

$$a_{ik}^t = \frac{\exp(s_{1;ik'}^t)}{\sum_{k'=1}^{K} \exp(s_{1;ik'}^t)}; \quad b_{ck}^t = \frac{\exp(s_{2;c'k}^t)}{\sum_{c'=1}^{C} \exp(s_{2;c'k}^t)}$$

Input-to-label message aggregation. Likewise the two-step *input-to-label* message aggregation is computed as:

$$\boldsymbol{m}_c^t = \sum_{k=1}^{K} \alpha_{ck}^t \boldsymbol{\chi}_k^{t-1}; \quad \text{for} \quad \boldsymbol{\chi}_k^{t-1} = \sum_{i=1}^{|\mathcal{V}|} \beta_{ik}^t \boldsymbol{x}_i^{t-1}$$

where $\boldsymbol{\chi}_k^{t-1}$ is the kth intermediate factor ($k \in \overline{1, K}$) that aggregates all input nodes; $\boldsymbol{m}_c^t$ is the message to the label node c; α_{ck}^t is factor-to-label attention probability (i.e., $\sum_k \alpha_{ck}^t = 1$); and β_{ik}^t is input-to-factor attention probability (i.e., $\sum_i \beta_{ik}^t = 1$). The attention probabilities are respectively computed as:

$$\alpha_{ik}^t = \frac{\exp(s_{1;ik}^t)}{\sum_{i'=1}^{|\mathcal{V}|} \exp(s_{1;i'k}^t)}; \quad \beta_{ck}^t = \frac{\exp(s_{2;ck}^t)}{\sum_{k'=1}^{K} \exp(s_{2;ck'}^t)}$$

where the scores $s_{1;ik}^t$ and $s_{2;ck}^t$ are computed using Eqs. (15,16).

It is clear that with this decomposition strategy, the number of computation steps reduces from $\mathcal{O}(|\mathcal{V}|C)$ to $\mathcal{O}((|\mathcal{V}| + C)\,K)$ for $K \ll \min\{|\mathcal{V}|,\ C\}$.

4.4 Detecting higher-order relation

Higher-order label-label relation The iterative message passing scheme spreads information to distant nodes. Two labels can indirectly interact with each other after two step of updates: a label sends messages to input nodes which then redistribute the information back to other labels. This brings about higher-order label correlation.
Multi-resolution substructure-label relation Likewise, after t steps, an input node accumulates information from other nodes within t hops. Because t varies from 1 to T, our model can detect label-specific substructures with multiple resolutions via the label-to-input attention. This capability is discussed in detail in Sect. 5.1.8.

5 Experiments

We present empirical results on two comprehensive sets of experiments: one on graph-structured input (Sect. 5.1) and the other on traditional unstructured input (Sect. 5.2).

5.1 Multilabel classification with graph-structured input

Our experiments focus on biochemical databases of potential drugs. A drug is a moderate-sized molecule with desirable bioactivities treated as labels. In the molecular graph of a drug, nodes represent atoms and edges represent bond types.

5.1.1 Datasets

We use two real-world biochemical datasets:

- *9cancers* For this dataset, the goal is to predict drug activity against nine types of cancer (see Table 2). The activity is binary indicating whether there is a response, i.e., the drug reduces or prevents tumor growth. We first download nine separate datasets for each cancer type from PubChem.[1] Then, we search for drug molecules that appear in all datasets, which results in about 22 thousand molecules in total. Among them, there are 3,356 molecules active for at least one type of cancer. We select all the active molecules and 10,000 fully inactive molecules to create the final dataset for experiment.
- *50proteins* This dataset is about drug-protein binding prediction. Again, drugs are treated as input graphs while proteins are labels. We obtain the raw version from BindingDB.[2] In this dataset, the number of unique proteins (also called targets) is 595 and the number of unique drugs (or ligands) is 55,781. We select top 50 proteins that are bound by most ligands to construct our experimental dataset. There are 36,349 ligands in total with the average number of proteins to which one ligand binds is 1.35.

We divide each dataset into train/valid/test sets with the proportions of 0.6/0.2/0.2, respectively. The detailed statistics are shown in Table 3 and the number of label occurrences is shown in Fig. 3. The labels in *50proteins* are sparse as each ligand links to at most 10 proteins (but the majority of ligands bind to only 1 or 2 proteins). Meanwhile, the labels in *9cancers* are denser with nearly a thousand of drugs positive to all cancers.

[1] https://pubchem.ncbi.nlm.nih.gov/.

[2] http://www.bindingdb.org/bind/index.jsp.

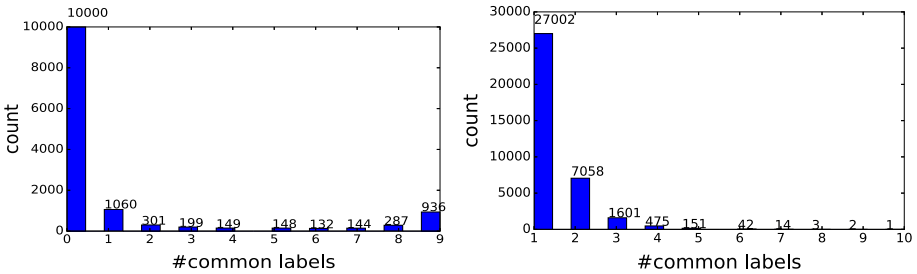
Table 2 Assay ID and name of nine cancers in *9cancers* dataset extracted from PubChem

Assay ID	Cancer type	%Positive
1	Lung	12.28
33	Melanoma	9.97
41	Prostate	11.77
47	Central nervous system	12.22
81	Colon	14.50
83	Breast	16.22
109	Ovarian	12.76
123	Leukemia	18.91
145	Renal	12.03

%Positive denotes the average percentage of positive examples for each cancer type over the total number of 13,356 molecules

Table 3 Statistics of all multilabel datasets with graph-structured inputs

Dataset	#labels	avg. #nodes	avg. #edges	#node types	#edge types
9cancers	9	27.68	29.95	43	4
50proteins	50	25.31	27.49	14	4



(a) 9cancers **(b)** 50proteins

Fig. 3 Histogram of the number of common labels that each instance associates to in *9cancers* and *50proteins*

5.1.2 Baselines

For comparison, we employ the following data representations and associated multilabel classifiers:

Molecular fingerprint The first set of baselines works on *molecular fingerprints*. A molecular fingerprint is a binary vector whose each element is associated with a particular type of substructures in the molecular graph. We use the well-known Morgan algorithm from RDKit[3] to generate multiple fingerprints with an increasing radius from 1 to 5 to account for fine-grained levels of substructures. Then, these fingerprints are concatenated to form a final feature vector. For each radius, we set the length of the fingerprint hash vector to 100. This results in the final feature vector of size 500. We evaluate two models running on top of this vector representation:

[3] http://www.rdkit.org/

– The first model is a SVM with RBF kernels set as a base classifier for Binary Relevance algorithm (Tsoumakas and Katakis 2007). We denote this model as fp + SVM + BR.
– The second model is a Highway Network (HWN) (Srivastava et al. 2015) followed by a fully connected neural network with sigmoid activation function. All highway layers share parameters. We denote the combination of fingerprint and HWN as fp + HWN. In this model, the dependencies among classes are implicitly captured through the intermediate hidden layers.

String representation SMILES is one of the most popular string representation of molecules which encapsulates the graph structure into its grammar. We consider SMILES as a sequence of characters and model it using a GRU (Cho et al. 2014). When reaching the end of the sequence, the last state of the GRU is fed to a 2-layer MLP that outputs prediction for all labels. This SMILES+GRU combination has been recently proven to be highly effective in drug evaluation and design (Segler et al. 2017).
Graph representation The last set of baselines handle graph-structured input directly. We select two representative models: Weisfeiler-Lehman Graph Kernels (WLs) (Shervashidze et al. 2011) for graph kernel based methods and Column Networks (CLNs) (Pham et al. 2017) for graph neural network based methods.

– WL is an unsupervised graph2vec model that maps a graph into a characteristic representation vector. Each element of this vector is the count of a specific rooted subgraph (or tree) in the graph. Because the length of the graph representation vector is equal to the vocabulary size of the trees which is very big, in practice, the similarity (kernel) matrix for every pair of graphs is used instead. We precompute the kernel matrix for both training and testing data using the Weisfeiler-Lehman algorithm. The maximum height of the trees is chosen to be 3. For *9cancers*, it results in about 49 thousand different tree structures for the entire graph dataset. Meanwhile, the total number of graphs is only about 13 thousands. Therefore, increasing height more than 3 will add very little information about graph similarity as the proportion of matching substructures approach zero. The kernel matrix for training graphs is used as input to a SVM wrapped by Binary Relevance (WL+SVM+BR) for multilabel classification.
– CLN, on the other hand, is a supervised graph message passing neural network. We use the same model as in Pham et al. (2017) with a mean pooling layer on top message passing layers to compute the graph representation vector. This vector is then fed to a 2-layer MLP to predict all labels. Different from our GAML, a CLN only captures the relations between the labels and the subgraphs at the topmost layer rather than at every layer.

The hyper-parameters of fp + HWN and SMILES + GRU are obtained through validation. Meanwhile, the hyper-parameters of CLN are set similar to the optimal hyper-parameters of our model (see below).

5.1.3 Model setting

In our model, the sizes of the node and edge embedding are both set to 50. We perform grid search for other hyper-parameters with the label embedding size in {10, 30, 50, 70, 100}, the number of factors in {1, 5, 10, 15, 20}, and the number of message passing layers in {2, 4, 6, 8, 10}. Dropout is set for every graph input node with the rate of 0.3. We do not use dropout for label nodes as it results in low F1 score although it makes the model less overfitting. In addition, we set the batch size to 60 and 100 for *9cancers* and *50proteins*,

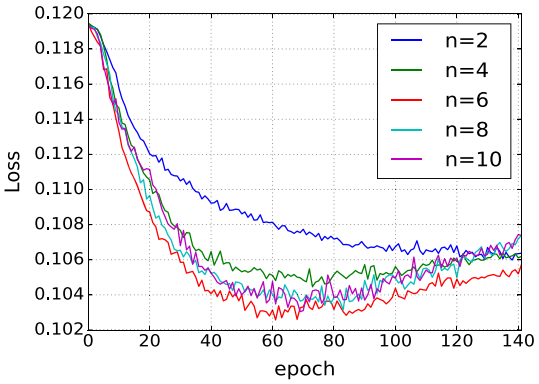
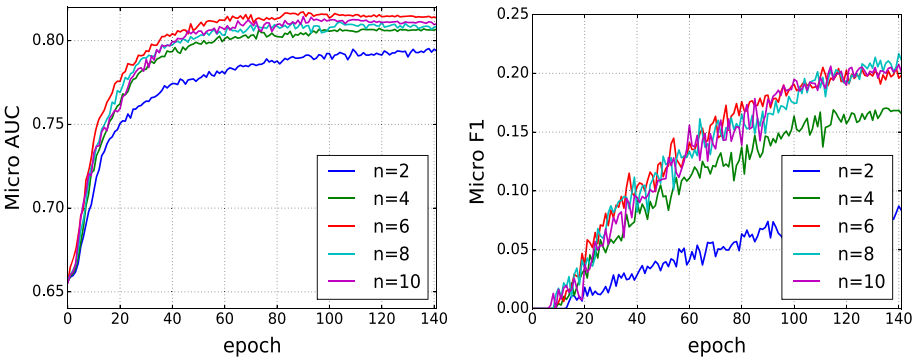


Fig. 4 Learning curves on *50proteins* with different number of message passing layers $n \in \{2, 4, 6, 8, 10\}$. Best viewed in color



(a) Micro AUC **(b)** Micro F1

Fig. 5 Micro AUC (**a**) and micro F1 (**b**) on 50proteins with different number of message passing layers $n \in \{2, 4, 6, 8, 10\}$. Best viewed in color

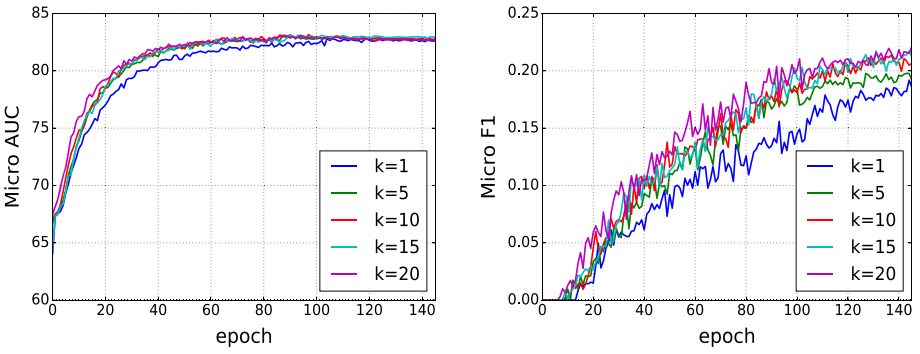
respectively. We use Adam optimizer (Kingma and Jimmy 2014) with an initial learning rate of 0.001. During training, the learning rate will be reduced by half if the validation loss does not improve after 20 consecutive epochs. We train our model for a maximum of 300 epochs and may stop early after decaying the learning rate 4 times.

5.1.4 Evaluation metrics

We use popular metrics for multilabel classification which are micro, macro (sometimes called per label) F1 and micro, macro AUC. While micro F1 favors labels with many examples due to its global averaging, macro F1 treats all labels equally regardless of their sample size, hence, is a good indication of the model performance on small labels.

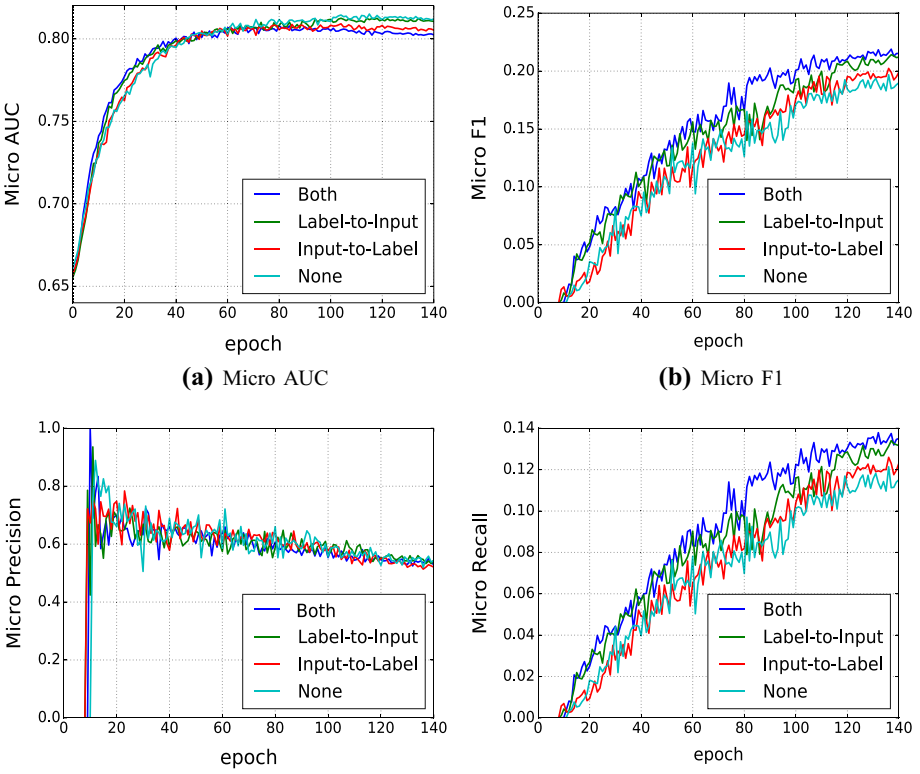
5.1.5 Parameter sensitivity

To have a deep understanding of how GAML works for graph structured input, we investigate the contribution of different hyper-parameters including: the number of message passing layers (Figs. 4, 5), the number of attention factors (Fig. 6), and the type of attention (Fig. 7). We report results for *50proteins*, but similar results are also observed for *9cancers*.



(a) Micro AUC **(b)** Micro F1

Fig. 6 Micro AUC (**a**) and micro F1 (**b**) on 50proteins with different number of factors $k \in \{1, 5, 10, 15, 20\}$. Best viewed in color



(a) Micro AUC **(b)** Micro F1

(c) Micro Precision **(d)** Micro Recall

Fig. 7 Results on *50proteins* with different type of attentions. *Label-to-Input* refers to unidirectional attention from label to input nodes; *Input-to-Label* refers to attention in the reverse direction; *Both* refers to bidirectional attention. Best viewed in color

From Fig. 5, it is seen that when the number of layers n is small, e.g. $n = 2$, the model performs sub-optimally. Increasing the number of layers usually improves the results. We hypothesize that at higher level, input nodes receive a wider range of structural information through message passing. However, when $n \geq 6$, the improvement rate becomes steady and the model is more likely to overfit (see Fig. 5c). We believe there are two reasons for this situation: (i) the structure information from distant nodes is much less important than that from close neighbors; and (ii) the structure information at every node becomes more global and indistinguishable, causing difficulty for the model to detect meaningful substructures during prediction.

Another factor that affects the model performance is the type of attention. Generally, using attention provides better micro F1 score than not using it. However, the input-to-label attention seems to be redundant and causes misleading to the model. We observed that when the input-to-label attention is available, the model often has higher loss and lower micro AUC (see Fig. 7a, c). Meanwhile, the label-to-input attention is important as it helps the label nodes focus on particular substructures of the input graph to give accurate prediction. One interesting thing to note here is that the improvement of micro F1 by using attention mainly comes from micro Recall (as can be seen from Fig. 7b, d, e) and since the denominator in the micro Recall formula is constant (which is equal to the number of positive examples in the dataset), the number of true positives actually increases.

GAML performs worst in term of both micro AUC and micro F1 when the number of attention factors k is 1 which is equivalent to collapsing all the neighbor nodes into one aggregating vector. For other values from 10 to 20, the results are quite comparable, which suggests that a small value of k is usually sufficient.

5.1.6 Performance results

Table 4 shows the classification results for graph structured input. GAML consistently beats all baselines on all evaluation metrics. In particular, our model achieves about 2–3% higher F1 and about 0.25–1% higher AUC than the second best method (CLN) on both datasets. We believe this improvement comes from the fact that our model can associate labels with useful multi-resolution substructures of the input graph through attention mechanism while CLN does not have this capability. Furthermore, it is also clear that the models learning directly on graphs such as WL + BR or CLN usually provide better results than those learning on strings or vectors. For example, CLN achieves roughly 2% improvement in term of micro and macro F1 compared to its vector counterpart fp + HWN. Whereas, WL+BR produces about 2–4% higher macro and micro AUC than fp+SVM+BR.

5.1.7 External knowledge of label dependencies

A priori label dependencies are known to improve model performance as they bring structural constraints to the output space (McCallum and Pereira 2001; Tsochantaridis et al. 2004). We consider the setting where label dependencies form a graph. The multilabeling becomes node classification in the label graph *conditioned on the input graph*. We investigate the case of *50proteins* where the labels are sparse. We compute the protein-protein interaction (PPI) scores by using Human Integrated Protein-Protein Interaction rEference (HIPPIE) (Alanis-Lobato et al. 2016). HIPPIE provides a normalized scoring scheme that integrates multiple PPI sources (Schaefer et al. 2012), hence, is reliable. The PPI scores have already been normalized in the range of [0, 1]. We add an edge between two proteins if their interaction

Table 4 The performance in the multi-label classification with graph-structured input (m-X: micro average of X; M-X: macro average)

Dataset	Metrics	Fingerprint		SMILES	Molecular Graph		
		SVM	HWN	GRU	WL+SVM	CLN	GAML
9cancers	m-AUC	81.94	85.95	83.29	86.06	88.35	**88.78**
	M-AUC	81.37	85.85	82.74	85.74	88.23	**88.50**
	m-F1	50.63	57.44	55.97	54.55	59.48	**62.03***
	M-F1	50.71	57.29	55.99	54.54	59.50	**62.14***
50proteins	m-AUC	79.85	77.46	79.11	81.62	82.08	**82.82**
	M-AUC	74.77	73.78	75.25	77.60	78.36	**79.35***
	m-F1	17.21	16.37	16.08	17.04	18.37	**20.47***
	M-F1	18.40	15.87	14.96	18.66	17.72	**19.83***

SVM and HWN work on fingerprint representation; GRU works on string representation of molecule known as SMILES; WL + BR and CLN work directly on graph representation. Bold indicates better values
$^*p < 0.05$

Table 5 Results on incorporating external knowledge of label dependencies

Model	*50proteins*			
	m-AUC	M-AUC	m-F1	M-F1
GAML	**82.82**	**79.35**	20.47	19.83
GAML + PPI	82.61	79.29	**21.15**	**20.28**

Bold indicates better values

score is larger than a predefined threshold (which set to 0.5 in our experiment). Since the interaction scores are asymmetric, the edges are directed. Table 5 reports results of our model when external label dependencies are introduced. The results are improved on F1 measures but not on the AUC scores suggesting that the external label constraints may help balance recall and precision when labels are sparse.

5.1.8 Attention visualization

In Fig. 8, we show the label-to-input attention scores at different message passing layers when our model runs on *9cancers* to see how our model matches labels to substructures of the input graph. At the first layer, the label nodes often attend to many input nodes. The reason is that input nodes at this level only contain information about their types. In addition, the attended input nodes are usually special atoms like Oxygen (8) or Nitrogen (7) instead of the common Carbon (6). However, the attention becomes more focused when going up to higher layer since the structure information at each input node has been updated via message passing. Sometimes, new substructures emerge and the model may switch its attention to these substructures if it finds them to be more appropriate.

From the label-to-input attention matrices in Fig. 8, we can map back to the molecule graph to detect meaningful substructures toward labels. In Fig. 9, we can observe the shift in the model attention with respect to the evolution of structures across layers. In particular, at layer 2, the model focuses most on the O-N substructure. However, at layer 3, the model changes its attention to N[6], N[5] and C[11] instead of Os. The reason is that the model becomes more interested in the appearance of two adjacent Ns in an aromatic group, which

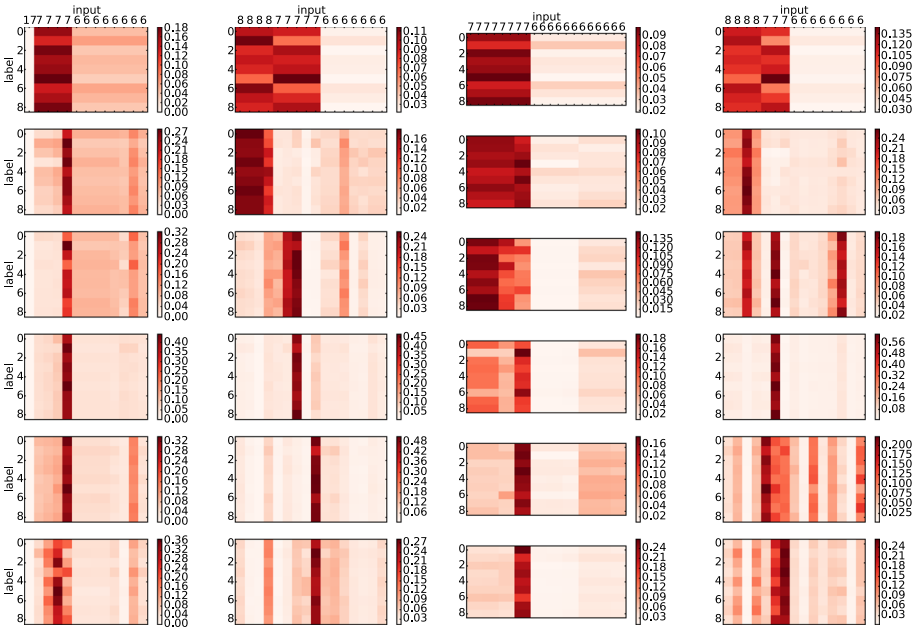


Fig. 8 Normalized label-to-input attention probability at 6 layers of GAML over 4 different molecular graphs sampled from *9cancers*. Darker color refers to higher probability. Columns correspond to input graphs and rows correspond to layers with the first layer drawn on top then the second layer and so on. Each tick in the x-axis is labeled with the atomic number of the corresponding node in the input graph (6: Carbon, 7: Nitrogen, 8: Oxygen, 17: Chlorine). Best view in color (Color figure online)

cannot be captured within two hops by starting at O. Therefore, an *attention shift* is performed by the model.

Note that although the attention shift looks disruptive in Fig. 8 as the model is highly attentive (due to well training), it is actually smooth under graphical view in Fig. 9 since the substructures rooted at N[6], N[5] and C[11] all contains the substructure O-N from the previous layer. This suggests a human-like concept transferring mechanism through attention where the old concept is not totally discarded but still exists as part of the new concept with less focusing from the brain. From layer 3 to layer 6, the model performs one more small attention shift (from N[6] to N[8]). We hypothesize the model does that to keep itself attended to the left ring only (instead of both the left and the right rings). This is reasonable because when N[8] receives more redundant information about the right ring, its attention score reduces from 0.48 (the 5th row) to 0.27 (the 6th row). The strong focus of the model on a particular substructure is also well demonstrated in Fig. 9. As we can see in the last row, although C[10] (at the last column) contains information about the whole molecular graph, its attention score is still significantly smaller than of the substructure rooted at N[8].

To discovery typical rooted substructures at a particular depth for a group of classes, we select a node with the best attention score averaged over the present classes for every molecular graph in the training data. Then, we perform clustering on the representation vector of these nodes to find similar substructures. Figure 10 shows an example of such common substructures shared by different molecules that is typical to all classes in *9cancers*.

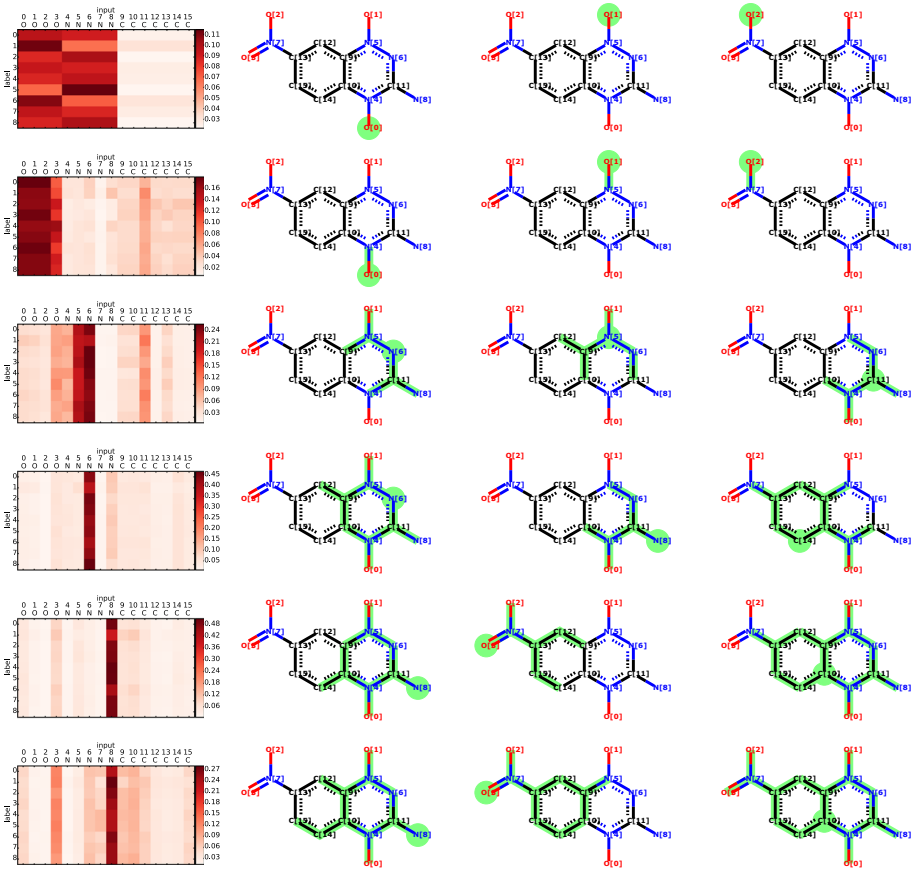


Fig. 9 Attention visualization on substructures of a molecule with PubChem SID of 491286. This molecule is the second example in Fig. 8. Each row specifies the top 3 substructures with the highest attention score (sorted in descending order from left to right) at the corresponding layer. For each substructure at layer k, the root atom as well as its neighbor atoms and bonds up to k hops are highlighted in green. Each atom is displayed with its atomic number and its index number (in square brackets) in the molecule. Best view in color (Color figure online)

5.2 Multilabel classification with unstructured input

We now test whether our proposed method can work on the traditional setting where the input is a vector.

5.2.1 Datasets

Four datasets are used in this experiments: *media_mill*, *bookmarks*, *Corel5k* and *NUS-WIDE* (see Table 6 for statistics). The former two belong to the text categorization domain where each instance is a document represented as binary bag-of-words. Meanwhile, the latter two belong to the image classification domain where each image is represented as a real-value feature vector. For all datasets, we follow the predefined train/test split so that our results can be comparable to others.

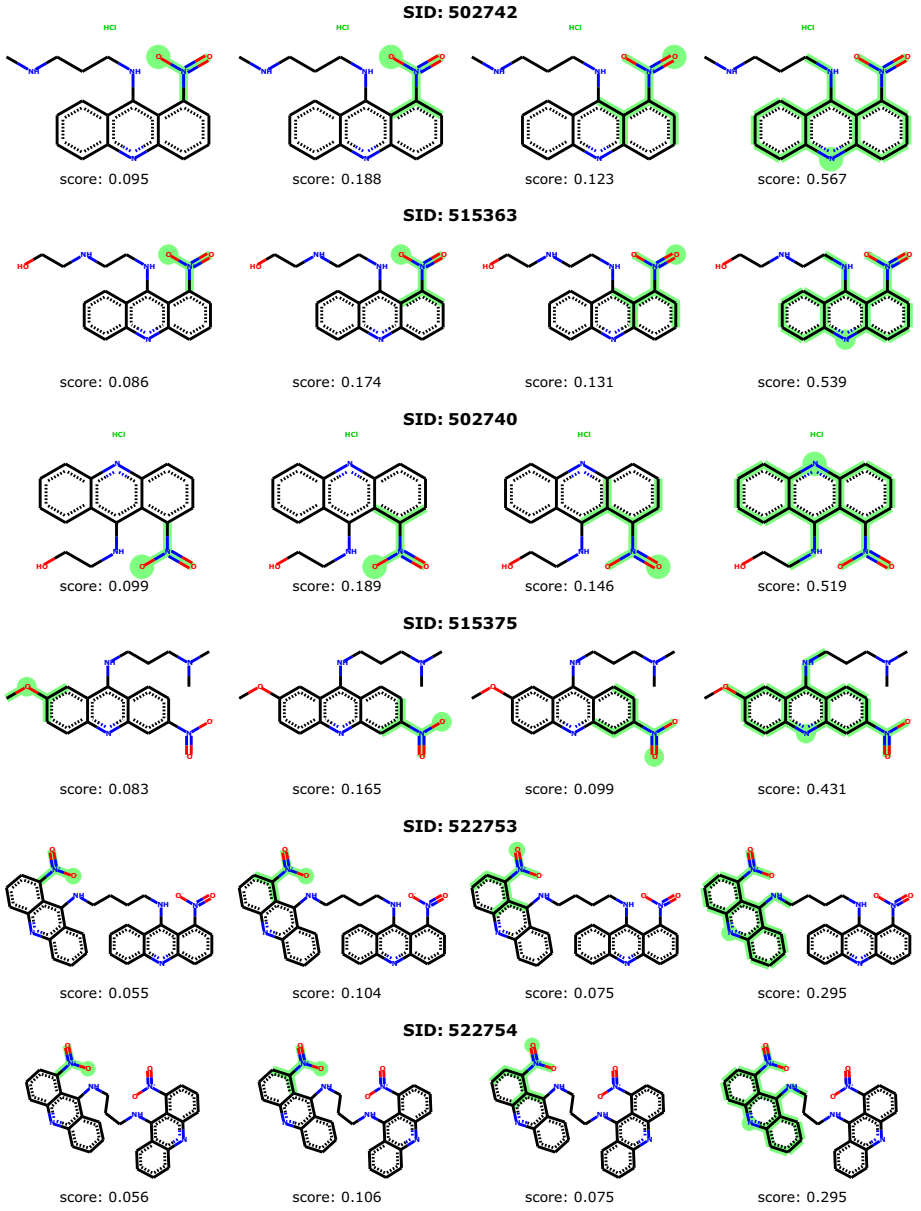


Fig. 10 Common substructures shared by some molecules that are typical to all classes in *9cancers*. Pictures from left to right show the evolution of the rooted substructures with depth from 2 to 5. Along the rows, the molecules are sorted by the average attention scores computed at the topmost layer. Best view in color (Color figure online)

Table 6 Statistics of all multilabel datasets with unstructured input

Dataset	#labels	#features	#total	#train	#test
media_mill	101	120	43,907	30,993	12,914
bookmarks	208	2150	87,856	70,285	17,571
Corel5k	374	499	5,000	4,500	500
NUS-WIDE	81	128	269,648	161,789	107,859

5.2.2 Baselines

For comparison, we consider the following methods:

- State-of-the-art classical methods for multilabel classification (MLC) evaluated in Madjarov et al. (2012), which are representative for broader classes of algorithms. They are RAkEL (Tsoumakas and Vlahavas 2007) for ensemble methods, ML-kNN (Zhang and Zhou 2007) for algorithm adaptation methods, HOMER (Tsoumakas et al. 2008) for label power set methods and Calibrated Label Ranking (Fürnkranz et al. 2008) for pairwise ranking. Most of these methods are implemented in well-known multilabel machine learning systems, such as Tsoumakas et al. (2011) and Read et al. (2016) with careful hyper-parameters tuning by the authors. Thus, their result are strong and reliable. For presentation compactness, we only report the best results in Madjarov et al. (2012).
- Collective multilabel classification with CRF (CML) (Ghamrawi and McCallum 2005). This model can learn pairwise relations among labels via CRF, hence, should be selected as baseline for comparison. We use the Java implementation of CML released on Github[4] and search for the optimal values of "train.gaussianVariance" in {0.01, 0.03, 0.1, 0.3, 1, 3, 10}. However, we can only test this model on *media_mill* and *NUS-WIDE* since the other two datasets are not accepted by the implementation.
- A Highway Network (HWN) (Srivastava et al. 2015), similar to what described in Sect. 5.1.

5.2.3 Model setting

The label embedding size is set to 50 for *NUS-WIDE* and *media_mill*, 75 for *bookmarks* and 30 for *Corel5k*. We project input vector to a low dimensional space by using a single layer neural network with ReLU activation before feeding it to GAML. The size of the projected vector is 55 for *NUS-WIDE*, 75 for *media_mill*, 110 for *bookmarks* and 50 for *Corel5k*. For all datasets, the number of message passing layers is set to 6. In training, the batch size for *NUS-WIDE* is 500 while for the other datasets, it is 100. We use k-fold cross validation where k is 9 for *Corel5k* and 5 for other dataset. The optimizer is Adam with an initial learning rate of 0.001. We reduce the learning rate by half if the valid loss does not decrease after 5 consecutive epochs for *Corel5k* and 20 for other datasets. The maximum number of epochs is 300 and the early stopping condition is 4 times of the learning rate decay.

5.2.4 Results

The classification performance of all the methods is presented in Table 7. The deep networks (HWN and GAML) outperform traditional methods and CML on most datasets except for

[4] https://github.com/cheng-li/pyramid/wiki/CRF.

Table 7 The performance in the multi-label classification with unstructured input (m-X: micro average of X, M-X: macro average of X)

Dataset	Metrics	Best in Madjarov et al. (2012)	CML (Ghamrawi and McCallum 2005)	HWN	GAML
media_mill	m-F1	56.3	45.78	56.73	**57.53**
	M-F1	11.3	3.95	13.50	**14.17**
bookmarks	m-F1	26.8	–	32.51	**33.33**
	M-F1	11.9	–	20.43	**21.68**
Corel5k	m-F1	**29.3**	–	15.28	22.13
	M-F1	**4.2**	–	1.83	3.82
NUS-WIDE	m-F1	–	30.42	38.50	**39.83**
	M-F1	–	3.84	9.31	**11.38**

Missing results are because NUS-WIDE was not tested in Madjarov et al. (2012) and CML implementation did not work on *bookmarks* and *Corel5k*. Bold indicates better values

Corel5k – the smallest dataset. Especially, on *bookmarks*, our model improves the micro F1 and macro F1 over the best traditional methods by about 7% and 10%, respectively. In the case of *Corel5k*, the best traditional method is CLR (see Madjarov et al. 2012), a ranking-based method that uses SVM as a base classifier. SVM appears to be more robust than deep networks on small datasets like *Corel5k*. Compared to HWN, GAML achieves better results on all datasets. This supports our model's strength in learning relations between labels and the input at multiple levels of abstraction.

Remark on running time

Kernel methods such as SVM and WL + SVM are not very scalable against data size due to the quadratic storage and cubic running time for inversion of the kernel matrix. Training these models may take tens of hours on a single CPU with a moderate data size of 30K. Deep learning methods such as HWN, GRU, CLN and GAML do not suffer from the same limitation and they are trained on GPUs, hence run much faster. HWN runs significantly faster than GAML since it only deals with flat vectors instead of structured graphs. CLN is lightly faster than GAML (about 10%) since it shares similar message passing complexity.

6 Discussion

We introduced GAML, a new graph neural network to tackle an open problem of multi-label learning over graph structured data. The key insight is to realize that label nodes and input nodes can be put into a joint graph to model the multi-way relations among labels and subgraphs. This is achieved through a message passing scheme that exchanges information between connected nodes across multiple steps and an attention mechanism that enables selective flowing of information between label nodes and input nodes. Our model is highly flexible and scalable. We evaluated GAML using an extensive set of experiments on both graph structured and unstructured inputs. Our results clearly demonstrate the efficacy of the proposed model.

This work opens up a wide room for the future at both applied and theoretical fronts. GAML is directly applicable to many other domains. One example is shopping basket recommendation, where users play the role of labels (with or without profile), and item basket modeled as input graph of items. Alternatively, items recommendation to user group works in a similar way, where the user group forms a social graph, and items play the role of labels. At the modeling front, a next step is to extend GAML from label node classification to full graph prediction, where edges are also predicted. Additionally, the current setting is open for auxiliary tasks, e.g., the input graph is node-labeled.

References

Alanis-Lobato, G., Andrade-Navarro, M. A., & Schaefer, M. H. (2016). HIPPIE v2. 0: Enhancing meaningfulness and reliability of protein–protein interaction networks. *Nucleic Acids Research*, gkw985.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. ArXiv preprint arXiv:1409.0473.

Bi, W., & Kwok, J. T. (2011). Multi-label classification on tree and dAG-structured hierarchies. In *Proceedings of the international conference on machine learning*, pp. 17–24.

Bruna, J., Zaremba, W., Szlam, A., & LeCun, Y. (2013). *Spectral networks and locally connected networks on graphs*. ArXiv preprint arXiv:1312.6203.

Chen, M., Lin, Z., & Cho, K. (2018). *Graph convolutional networks for classification with a structured label space*. ArXiv preprint arXiv:1710.04908.

Chen, S.-F., Chen, Y.-C., Yeh, C.-K., & Wang, Y.-C. F. (2017). *Order-free RNN with visual attention for multi-label classification*. ArXiv preprint arXiv:1707.05495.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., et al. (2014). *Learning phrase representations using RNN encoder-decoder for statistical machine translation*. arXiv preprint arXiv:1406.1078.

Defferrard, M. l., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of advances in neural information processing systems*, pp. 3844–3852.

Dembczynski, K., Cheng, W., & Hüllermeier, E. (2010). Bayes optimal multilabel classification via probabilistic classifier chains. In *Proceedings of the international conference on machine learning*, Vol. 10, pp. 279–286.

Deng, J., Ding, N., Jia, Y., Frome, A., Murphy, K., Bengio, S., et al. (2014). Large-scale object classification using label relation graphs. In *Proceedings of the European conference on computer vision* (pp. 48–64). Springer.

Fout, A., Byrd, J., Shariat, B., & Ben-Hur, A. (2017). Protein interface prediction using graph convolutional networks. In *Proceedings of advances in neural information processing systems*, pp. 6533–6542.

Fürnkranz, J., Hüllermeier, E., Mencía, E. L., & Brinker, K. (2008). Multilabel classification via calibrated label ranking. *Machine Learning*, *73*(2), 133–153.

Ghamrawi, N., & McCallum, A. (2005). Collective multi-label classification. In *Proceedings of the international conference on information and knowledge management* (pp. 195–200). ACM.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *Proceedings of the international conference on machine learning*.

Gong, Y., Jia, Y., Leung, T., Toshev, A., & Ioffe, S. (2013). *Deep convolutional ranking for multilabel image annotation*. ArXiv preprint arXiv:1312.4894.

Grover, A., & Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the international conference on knowledge discovery and data mining* (pp. 855–864). ACM.

Guo, Y., & Gu, S. (2011). Multi-label classification using conditional dependency networks. In *Proceedings of the international joint conference on artificial intelligence*, Vol. 22, p. 1300.

Hamilton, W., Ying, Z., & Leskovec, J. (2017a). Inductive representation learning on large graphs. In *Proceedings of advances in neural information processing systems*, pp. 1025–1035.

Hamilton, W. L., Ying, R., & Leskovec, J. (2017b). *Representation learning on graphs: Methods and applications*. ArXiv preprint arXiv:1709.05584.

Jin, W., Barzilay, R., & Jaakkola, T. (2018). *Junction tree variational autoencoder for molecular graph generation*. ArXiv preprint arXiv:1802.04364.

Kingma, D., & Ba, J. (2014). *Adam: A method for stochastic optimization*. ArXiv preprint arXiv:1412.6980.

Kipf, T. N., & Welling, M. (2016a). *Semi-supervised classification with graph convolutional networks*. ArXiv preprint arXiv:1609.02907.
Kipf, T. N., & Welling, M. (2016b). *Variational graph auto-encoders*. ArXiv preprint arXiv:1611.07308.
Kong, X., & Philip, S. Y. (2012). gMLC: A multi-label feature selection framework for graph classification. *Knowledge and Information Systems*, *31*(2), 281–305.
Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the international conference on machine learning*, pp. 282–289.
Li, X., & Guo, Y. (2015). Multi-label classification with feature-aware non-linear label space transformation. In *Proceedings of the international joint conference on artificial intelligence*, Vol. 2015, pp. 3635–3642.
Li, Y., Tarlow, D., Brockschmidt, M., & Zemel, R. (2016). Gated graph sequence neural networks. In *Proceedings of the international conference on learning representations*.
Li, Y., Vinyals, O., Dyer, C., Pascanu, R., & Battaglia, P. (2018). *Learning deep generative models of graphs*. ArXiv preprint arXiv:1803.03324.
Madjarov, G., Kocev, D., Gjorgjevikj, D., & Džeroski, S. (2012). An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, *45*(9), 3084–3104.
Mullard, A. (2014). New drugs cost US [dollar] 2.6 billion to develop. *Nature Reviews Drug Discovery*, *13*(12), 877–877.
Narayanan, A., Chandramohan, M., Venkatesan, R., Chen, L., Liu, Y., & Jaiswal, S. (2016). *graph2vec: Learning distributed representations of graphs*. ArXiv preprint arXiv:1707.05005.
Niepert, M., Ahmed, M., & Kutzkov, K. (2016). Learning convolutional neural networks for graphs. In *Proceedings of the international conference on machine learning*.
Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the international conference on knowledge discovery and data mining* (pp. 701–710). ACM.
Pham, T., Tran, T., & Venkatesh, S. (2017). *One size fits many: Column bundle for multi-x learning*. ArXiv preprint arXiv:1702.07021.
Pham, T., Tran, T., & Venkatesh, S. (2018). *Graph memory networks for molecular activity prediction*. ArXiv preprint arXiv:1801.02622.
Pham, T., Tran, T., Phung, D., & Venkatesh, S. (2017). Column networks for collective classification. In *Proceedings of AAAI conference on artificial intelligence*.
Read, J., Reutemann, P., Pfahringer, B., & Holmes, G. (2016). Meka: A multi-label/multi-target extension to Weka. *Journal of Machine Learning Research*, *17*(1), 667–671.
Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2009). The graph neural network model. *IEEE Transactions on Neural Networks*, *20*(1), 61–80.
Schaefer, M. H., Fontaine, J.-F., Vinayagam, A., Porras, P., Wanker, E. E., & Andrade-Navarro, M. A. (2012). HIPPIE: Integrating protein interaction networks with experiment based quality scores. *PLoS ONE*, *7*(2), e31826.
Schlichtkrull, M., Kipf, T. N., Bloem, P., van den Berg, R., Titov, I., & Welling, M. (2017). *Modeling relational data with graph convolutional networks*. ArXiv preprint arXiv:1703.06103.
Segler, M. H. S., Kogej, T., Tyrchan, C., & Waller, M. P. (2017). Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS Central Science*, *4*, 120–131.
Shervashidze, N., Schweitzer, P., van Leeuwen, E. J., Mehlhorn, K., & Borgwardt, K. M. (2011). Weisfeiler-Lehman graph kernels. *Journal of Machine Learning Research*, *12*(Sep), 2539–2561.
Simonovsky, M., & Komodakis, N. (2018). *Graphvae: Towards generation of small graphs using variational autoencoders*. ArXiv preprint arXiv:1802.03480.
Srivastava, R. K., Greff, K., & Schmidhuber, J. (2015). Training very deep networks. In *Proceedings of advances in neural information processing systems*, pp. 2377–2385.
Sun, L., Ji, S., & Ye, J. (2011). Canonical correlation analysis for multilabel classification: A least-squares formulation, extensions, and analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *33*(1), 194–200.
Takigawa, I., & Mamitsuka, H. (2017). Generalized sparse learning of linear models over the complete subgraph feature set. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(3), 617–624.
Teney, D., Liu, L., & van den Hengel, A. (2017). Graph-structured representations for visual question answering. In *Proceedings of IEEE conference on computer vision and pattern recognition*.
Tsochantaridis, I., Hofmann, T., Joachims, T., & Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the international conference on machine learning*, pp. 823–830.
Tsoumakas, G., & Vlahavas, I. (2007). Random k-labelsets: An ensemble method for multilabel classification. In *Proceedings of the European conference on machine learning*, pp. 406–417. Springer.

Tsoumakas, G., Katakis, I., & Vlahavas, I. (2008). Effective and efficient multilabel classification in domains with large number of labels. In *Proceedings of ECML/PKDD workshop on mining multidimensional data* (Vol. 21, pp. 53–59). sn.

Tsoumakas, G., & Katakis, I. (2007). Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, *3*(3), 1–13.

Tsoumakas, G., Spyromitros-Xioufis, E., Vilcek, J., & Vlahavas, I. (2011). Mulan: A java library for multi-label learning. *Journal of Machine Learning Research*, *12*(Jul), 2411–2414.

Wang, H., Wang, J., Wang, J., Zhao, M., Zhang, W., Zhang, F., et al. (2017). *Graphgan: Graph representation learning with generative adversarial nets*. ArXiv preprint arXiv:1711.08267.

Wang, J., Yang, Y., Mao, J., Huang, Z., Huang, C., & Xu, W. (2016). CNN-RNN: A unified framework for multi-label image classification. In *Proceedings of IEEE conference on computer vision and pattern recognition*, pp. 2285–2294. IEEE.

Wei, Y., Xia, W., Lin, M., Huang, J., Ni, B., Dong, J., et al. (2016). HCP: A flexible CNN framework for multi-label image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *38*(9), 1901–1907.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., et al. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the international conference on machine learning*, pp. 2048–2057.

Yanardag, P., & Vishwanathan, S. V. N. (2015). Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1365–1374). ACM.

Yeh, C.-K., Wu, W.-C., Ko, W.-J., & Wang, Y.-C. F. (2017). Learning deep latent space for multi-label classification. In *Proceedings of AAAI conference on artificial intelligence*, pp. 2838–2844.

You, J., Ying, R., Ren, X., Hamilton, W. L., & Leskovec, J. (2018). *Graphrnn: A deep generative model for graphs*. ArXiv preprint arXiv:1802.08773.

Zhang, D., Yin, J., Zhu, X., & Zhang, C. (2017). *Network representation learning: A survey*. ArXiv preprint arXiv:1801.05852.

Zhang, M.-L., & Zhou, Z.-H. (2007). ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, *40*(7), 2038–2048.