Downloaded from https://academic.oup.com/bioinformatics/article-abstract/doi/10.1093/bioinformatics/btaa211/5813330 by Korea national university of transportation user on 21 April 2020

Systems Biology

Dual-Dropout Graph Convolutional Network for Predicting Synthetic Lethality in Human Cancers

Ruichu Cai [1], Xuexin Chen [1], Yuan Fang [2,*], Min Wu [3,*], Yuexing Hao [4]

[1] School of Computer Science, Guangdong University of Technology, 510006, China
[2] School of Information Systems, Singapore Management University, 178902, Singapore.
[3] Institute for Infocomm Research (I^2R), A*STAR, 138632, Singapore and
[4] Rutgers Univeristy New Brunswick, USA.

*To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: Synthetic lethality (SL) is a promising form of gene interaction for cancer therapy, as it is able to identify specific genes to target at cancer cells without disrupting normal cells. As high-throughput wet-lab settings are often costly and face various challenges, computational approaches have become a practical complement. In particular, predicting SLs can be formulated as a link prediction task on a graph of interacting genes. Although matrix factorization techniques have been widely adopted in link prediction, they focus on mapping genes to latent representations in isolation, without aggregating information from neighboring genes. Graph convolutional networks (GCN) can capture such neighborhood dependency in a graph. However, it is still challenging to apply GCN for SL prediction as SL interactions are extremely sparse, which is more likely to cause overfitting.

Results: In this paper, we propose a novel Dual-Dropout GCN (DDGCN) for learning more robust gene representations for SL prediction. We employ both coarse-grained node dropout and fine-grained edge dropout to address the issue that standard dropout in vanilla GCN is often inadequate in reducing overfitting on sparse graphs. In particular, coarse-grained node dropout can efficiently and systematically enforce dropout at the node (gene) level, while fine-grained edge dropout can further fine-tune the dropout at the interaction (edge) level. We further present a theoretical framework to justify our model architecture. Finally, we conduct extensive experiments on human SL datasets and the results demonstrate the superior performance of our model in comparison with state-of-the-art methods.

Availability: DDGCN is implemented in python 3.7, open-source and freely available at `https://github.com/CXX1113/Dual-DropoutGCN`

Contact: name@bio.com

1 Introduction

Cancer is a group of complex diseases often caused by the defects of more than one gene. Therefore, understanding the genetic interactions has become crucial for cancer therapies. Synthetic lethality (SL) is a promising type of genetic interaction between a pair of genes, where only the defects of both genes will significantly impair cell viability, but the defect of a single gene will not affect cell fitness. It therefore becomes feasible to target a non-essential gene in an SL interaction such that the other gene is a cancer-specific defective gene, which would enable the selective killing of cancer cells without harming normal cells (Iglehart and Silver, 2009). SL has thus emerged as a gold mine for anti-cancer drug research (O'Neil *et al.*, 2017). For example, Olaparib and Niraparib as PARP-inhibitors are two drugs based on the well-known SL between genes PARP and BRCA1/BRCA2 (Chan and Giaccia, 2011). They were approved by US FDA for ovarian and breast cancers in 2014 and 2017, respectively. Indeed, SL has been actively studied with high-throughput wet-lab screening methods. Unfortunately, these approaches often face various challenges, such as high cost, off-target effects, and inconsistency

© The Author(s) (2020). Published by Oxford University Press. All rights reserved. For Permissions, please email: journals.permissions@oup.com

across platforms or cell lines. Thus, predicting SL using computational models becomes highly complementary to wet-lab approaches.

In fact, SL interactions can be captured by a graph, where each gene is a node and interacting genes form edges in the graph. Subsequently, predicting novel SLs can be cast as a link prediction task in the graph. In particular, matrix factorization techniques have been successfully applied to address link prediction problems in bioinformatics, such as PPI prediction (Wang *et al.*, 2013) and drug-target interaction prediction (Liu *et al.*, 2016). A more recent work (Liu *et al.*, 2018) employs matrix factorization specifically for SL prediction, based on a comprehensive human SL database called SynLethDB (Guo *et al.*, 2016). However, matrix factorization employs direct encoding, where the encoder is just an embedding lookup (Hamilton *et al.*, 2017). As shown in Fig. 1 (a), gene 3's embedding is simply row 3 in the matrix of latent factors. In other words, such an approach does not leverage valuable information from neighboring genes in the graph.

In contrast, graph neural networks (GNNs) can effectively capture graph structures and model complex dependencies between neighboring nodes in the graph (Xu *et al.*, 2019). An archetype of GNNs is the widely adopted graph convolutional network (GCN) (Kipf and Welling, 2017). As shown in Fig. 1 (b), GCN generates the embedding for gene 3 by aggregating the representations of itself and its neighbors, thus leveraging the neighborhood dependency. The dropout technique (Srivastava *et al.*, 2014) is often applied in GCN to randomly remove some neurons (e.g., dotted units in the embeddings), which aims to prevent overfitting caused by co-adaptation. While standard dropout in vanilla GCN might be adequate in many scenarios, SL prediction presents a challenge due to its extreme sparsity—known SL interactions account for less than 0.1% of all possible pairs. Under such sparse interactions, overfitting is more likely to occur, which motivates us to redesign a more effective and robust dropout mechanism for GCN.

In this paper, we propose a novel Dual-Dropout Graph Convolutional Network (DDGCN) for predicting synthetic lethality. As Fig. 1 (c) illustrates, we propose dual forms of dropout, where the left involves a coarse-grained node dropout, and the right involves a fine-grained edge dropout. In the node dropout (left), we drop some genes (e.g., gene 2) randomly in each training epoch, forcing GCN to learn more robust representations without overfitting to a few genes. However, dropping gene nodes is sometimes too aggressive, since the removal of a node results in the removal of all its incident edges. As shown in the example, removal of gene 2 removes all its interactions with genes 1, 3 and 6. Such node dropout is thus coarse-grained, without the ability to control individual edges. On the other hand, in edge dropout (right), some edges (e.g., between genes 2 and 3) have been randomly dropped, which is fine-grained since not all incident edges of the involved nodes are affected indiscriminately (e.g., the interaction between genes 2 and 6 is still preserved). While coarse-grained node dropout is efficient at systematically reducing co-adaptions at gene levels, fine-grained edge dropout can further fine-tune the dropout on individual interactions. Thus, we integrate both coarse- and fine-grained dropouts into GCN to take the advantage of both.

To materialize the dual dropout, we employ an identity matrix and adjacency matrix simultaneously to feed into a GCN via two paths with shared parameters. The two paths correspond to the coarse-grained node dropout and fine-grained edge dropout, respectively. We further present a theoretical analysis to show that, our proposed architecture of Dual-Dropout GCN or DDGCN, is provably equivalent to dropping nodes and edges jointly on the same graph. We summarize our key contributions in the following.

- To our best knowledge, we are the first to address SL prediction with graph neural networks.
- We propose a novel Dual-Dropout GCN model that integrates both coarse- and fine-grained dropouts to better deal with sparse SL interactions, with a further theoretical analysis of our proposed architecture.
- We conduct extensive experiments on SL prediction, validating the effectiveness of DDGCN.

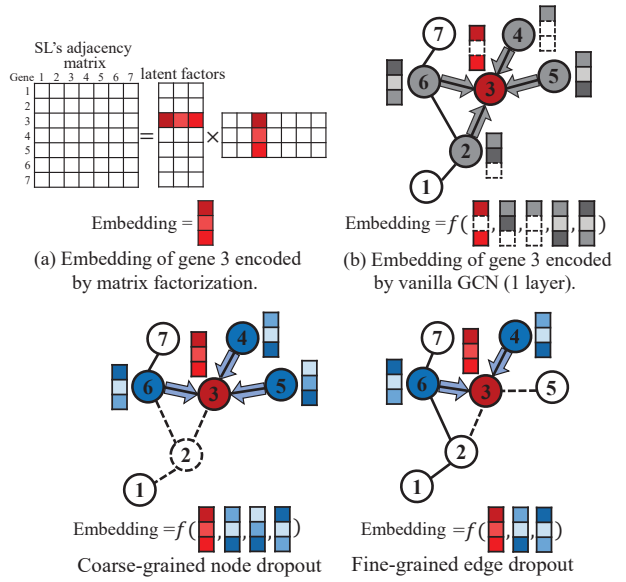


Fig. 1. Different methods to generate gene embeddings.

2 Related Work

In this section, we briefly introduce the existing methods for SL prediction, as well as graph embedding techniques and graph neural networks.

Various SL prediction methods can be divided into two categories, namely, unsupervised methods and supervised methods. Unsupervised methods leverage the prior knowledge or hypotheses for SL prediction. For example, DAISY (Jerby-Arnon *et al.*, 2014) and MiSL (Sinha *et al.*, 2017) predict SL using gene expression and mutation data, based on the knowledge that SL genes are usually co-expressed and seldom co-mutated. Connectivity homology refers to similar network connectivity patterns or features across species, and human gene pairs are predicted as SL interactions if they have connectivity homology to existing yeast SL pairs (Jacunski *et al.*, 2015). A knowledge-driven method proposed in (Zhang *et al.*, 2015) predicts potential SL interactions by simulating the influence of double-gene knockout to cell death in signaling pathways. In (Apaolaza *et al.*, 2017), minimum cut set of genes, whose simultaneous removal directly blocks a particular metabolic task (e.g., biomass reaction), are exploited for SL prediction. Unlike the above unsupervised methods, supervised methods are machine learning models trained on known SL data. For example, DiscoverSL (Das *et al.*, 2018) predicts novel human SL using multi-omic data as features and random forest as classifier. SLant (Benstead-Hume *et al.*, 2019) derives features from PPI networks and Gene Ontology (GO), and then performs in-species and cross-species SL prediction using random forest. In (Li *et al.*, 2019), functional features for genes are first derived from GO and KEGG and random forest is then used for SL prediction. In addition, SL^2MF (Liu *et al.*, 2018) and BLM-NII (Mei *et al.*, 2013) are also supervised methods for SL prediction. In particular, SL^2MF is a logistic matrix factorization (LMF) based method and it

Downloaded from https://academic.oup.com/bioinformatics/article-abstract/doi/10.1093/bioinformatics/btaa211/5813330 by Korea national university of transportation user on 21 April 2020

further incorporates gene-gene similarities to learn more accurate gene embeddings for SL prediction. BLM-NII is a type of bipartite local model originally for drug-target interaction prediction and it was customized in (Liu *et al.*, 2018) to predict SL.

Graph embedding techniques focus on converting the graph data into a low dimensional space for various downstream machine learning tasks (Cai *et al.*, 2018). Matrix factorization techniques with direct encoding as shown in Figure 1 (a), e.g., GraRep (Cao *et al.*, 2015) and HOPE (Ou *et al.*, 2016), have been widely used for graph embedding. Moreover, random-walk based graph embedding techniques, including DeepWalk (Perozzi *et al.*, 2014), node2vec (Grover and Leskovec, 2016) and LINE (Tang *et al.*, 2015), can also be considered as matrix factorization (Qiu *et al.*, 2018). To further model the complex dependencies between neighbors instead of direct encoding, graph neural networks have recently been proposed for graph embedding. In particular, GCN has recently been applied for various bioinformatics tasks, e.g., drug discovery (Sun *et al.*, 2019), miRNA and drug resistence association prediction (Huang *et al.*, 2019), lcnRNA-disease association prediction (Xuan *et al.*, 2019), disease gene prediction (Han *et al.*, 2019), etc. However, so far no work has been done for SL prediction using GCN.

In addition, dropout (Hinton *et al.*, 2012) has been proposed as a form of regularization for fully connected neural network layers. It has proved to be a successful technique in controlling overfitting by randomly omitting some features at each training iteration. Dropout also has many variants. DropConnect has been proposed in (Wan *et al.*, 2013) as a generalization of dropout, in which each connection rather than each output unit can be randomly dropped. Some work indicates that dropout can also be applied to RNN, e.g., (Gal and Ghahramani, 2016), (Semeniuta *et al.*, 2016) and (Krueger *et al.*, 2017) propose to apply dropout to the recurrent connections. In particular, the technique of dropout and its variants are directly applied to neural networks, without a clear interpretation on their effects on graph data. On the other hand, we present a theoretical analysis to explicitly link our proposed dual dropout to the input graph.

3 Methods

In this section, we first introduce the notations and problem statement, then present the details of our proposed DDGCN method, as well as a theoretical analysis of the proposed architecture.

3.1 Preliminary

An SL graph is denoted by $\mathcal{G} = (\mathcal{U}, \mathcal{E})$, with the set of nodes $\mathcal{U}$ to represent genes, and the set of edges $\mathcal{E}$ to represent SL interactions. Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be the adjacency matrix of $\mathcal{G}$, where $n = |\mathcal{U}|$ is the number of genes. Since the SL graph is undirected, the set of all gene pairs is $\mathcal{O} = \{(i,j) \in \mathcal{U} \times \mathcal{U} | i < j\}$. Subsequently, we denote the set of observed SL pairs by $\mathcal{O}^+ = \{(i,j) \in \mathcal{O} | \mathbf{A}_{ij} = 1\}$. The remaining gene pairs $\mathcal{O}^- = \mathcal{O} \setminus \mathcal{O}^+$ are "unknown" pairs, because there is a lack of evidence demonstrating whether these gene pairs form SL interactions.

In this paper, we investigate the problem of SL prediction. Formally, given a set of observed gene pairs $\mathcal{O}^+$ that are known to be SL pairs, the task is to predict, for each unknown gene pair from $\mathcal{O}^-$, if it is in fact an SL pair. In practise, we first estimate the probability that two genes form SL interactions, i.e., learn a decision function $f : \mathcal{O} \to [0, 1]$. Subsequently, we rank every unknown gene pair $o \in \mathcal{O}^-$ by $f(o)$ in descending order, such that the top-ranked gene pairs are likely to be novel SL pairs.

3.2 Overview of DDGCN

The overall framework is illustrated in Fig. 2. The main idea is to jointly drop nodes (genes) and edges (SL interactions) during the graph convolution operation. Specifically, our GCN-based architecture consists of two paths with shared parameters to model the dual forms of dropout: coarse-grained node dropout by employing the identity matrix in (a), and fine-grained edge dropout by employing the adjacency matrix in (b). It can be formally shown that our proposed dropout model is equivalent to dropping nodes and edges simultaneously from the same original graph.

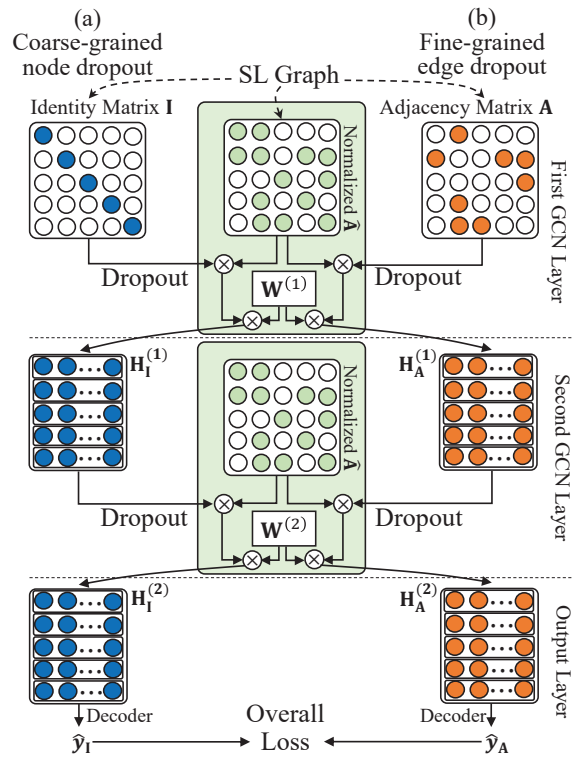


Fig. 2. Architecture of Dual-Dropout GCN (DDGCN).

3.3 Coarse-Grained Node Dropout

Consider an $n \times n$ identity matrix. Each row or column of $\mathbf{I}$ corresponds to a node in the graph. As shown in Fig. 2 (a), the identity matrix implies to a set of five gene nodes, i.e., $|\mathcal{U}| = n = 5$.

We apply dropout to the identity matrix, and subsequently encode the first-layer representation of each gene using graph convolution:

$$\mathbf{H}_{\mathbf{I}}^{(1)} = \text{ReLU}(\hat{\mathbf{A}}(\mathbf{M}_{\mathbf{I}}^{(1)} \odot \mathbf{I})\mathbf{W}^{(1)}). \quad (1)$$

Here $\odot$ is the element-wise multiplication. $\mathbf{M}_{\mathbf{I}}^{(1)} \in \mathbb{R}^{n \times n}$ is the dropout mask in the first layer such that each element follows the Bernoulli distribution *Bernoulli*(p) and p is the dropout rate. $\mathbf{W}^{(1)} \in \mathbb{R}^{n \times d_1}$ is a trainable weight matrix for the first layer ($d_1 \ll n$). $\hat{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\tilde{\mathbf{A}}\mathbf{D}^{-\frac{1}{2}}$ is the normalized graph adjacency matrix following (Kipf and Welling, 2017) where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\mathbf{D} \in \mathbb{R}^{n \times n}$ is a diagonal matrix, in which the diagonal elements are defined as $d_{ii} = \sum_{j=1}^{n} \tilde{\mathbf{A}}_{ij}$.

The first-layer embedding matrix $\mathbf{H}_{\mathbf{I}}^{(1)} \in \mathbb{R}^{n \times d_1}$ can be further fed into more layers of GCN. Without loss of generality, we present two layers only. In particular, the second-layer (output) embedding matrix $\mathbf{H}_{\mathbf{I}}^{(2)} \in \mathbb{R}^{n \times d_2}$ can be encoded as follows ($d_2 \ll n$):

$$\mathbf{H}_{\mathbf{I}}^{(2)} = \hat{\mathbf{A}}(\mathbf{M}_{\mathbf{I}}^{(2)} \odot \mathbf{H}_{\mathbf{I}}^{(1)})\mathbf{W}^{(2)}. \quad (2)$$

Here $\mathbf{M}_{\mathbf{I}}^{(2)} \in \mathbb{R}^{n \times n}$ is the dropout mask in the second layer. $\mathbf{W}^{(2)} \in \mathbb{R}^{n \times d_2}$ is a trainable weight matrix for the second layer ($d_2 \ll n$).

Downloaded from https://academic.oup.com/bioinformatics/article-abstract/doi/10.1093/bioinformatics/btaa211/5813330 by Korea national university of transportation user on 21 April 2020

Intuitively, if the non-zero element in the i^{th} row of $\mathbf{I}$ becomes zero after applying dropout, all elements in the i^{th} row of the product $(\mathbf{M}^{(l)} \odot \tilde{\mathbf{I}})\mathbf{W}^{(l)}$ become zero, which is effectively eliminating gene i's features (embedding) from consideration. In other words, we have dropped node i from the input graph.

3.4 Fine-Grained Edge Dropout

While dropping gene nodes completely can be efficient at preventing co-adaptions at gene levels, it is coarse-grained and can be aggressive sometimes, since removing a node is equivalent to removing all of its incident edges. Thus, we further propose the fine-grained edge dropout to complement node dropout.

Consider the adjacency matrix $\mathbf{A}$, where each element represents an edge in the graph. As shown in Fig. 2 (b), the i^{th} row or column of $\mathbf{A}$ corresponds to the SL interactions between gene i and other genes. As shown in Fig 2 (b), we apply dropout to $\mathbf{A}$ similar to Fig. 2 (a) in the following, for two layers of GCN.

$$\mathbf{H}_{\mathbf{A}}^{(1)} = \text{ReLU}(\hat{\mathbf{A}}(\mathbf{M}_{\mathbf{A}}^{(1)} \odot \mathbf{A})\mathbf{W}^{(1)}), \tag{3}$$

$$\mathbf{H}_{\mathbf{A}}^{(2)} = \hat{\mathbf{A}}(\mathbf{M}_{\mathbf{A}}^{(2)} \odot \mathbf{H}_{\mathbf{A}}^{(1)})\mathbf{W}^{(2)}. \tag{4}$$

Here $\mathbf{M}_{\mathbf{A}}^{(l)}$ is the dropout mask in the l^{th} layer.

Intuitively, we can regard $\hat{\mathbf{A}}\mathbf{A}$ as a 1-and-2-hop neighborhood matrix for the input graph. Note that the 1-hop neighborhood is also covered due to the self-loops in $\hat{\mathbf{A}}$. That is, we are using a slight variant of GCN by aggregating both 1- and 2-hop neighbors in the first layer. If the (i, j) element of $\hat{\mathbf{A}}\mathbf{A}$ becomes zero due to dropout, node j's features (embedding) will not be aggregated into node i's. This mechanism operates with a finer granularity than the node dropout, as node j's features will still be aggregated into other neighbors. In other words, the 1- or 2-hop interaction between genes i and j have been dropped, which we term as edge dropout.

It is also important to note that the GCN parameters $\mathbf{W}^{(l)}$ in each layer are *shared* by the dual dropouts, which is an important design to ensure that both node and edge dropouts are applied jointly on the same graph. To simply put, when the parameters are not shared by the dual paths in Fig. 2 (a) and Fig. 2 (b), we are actually training models on two different graphs. However, our goal is to jointly apply the dual forms of dropout on the same input graph. A formal proof for the necessity of sharing parameters are given in the last part of this section.

3.5 Overall Loss and Optimization

The dual dropout architecture gives us two embedding matrices $\mathbf{H}_{\mathbf{I}}^{(2)}$ and $\mathbf{H}_{\mathbf{A}}^{(2)}$, one from each path in Fig. 2. In each path, for every pair of genes $(i, j) \in \mathcal{O}$, we estimate its interaction confidence as follows:

$$\hat{y}_{\mathbf{I}}(i, j) = \text{Dec}(\mathbf{H}_{\mathbf{I}}^{(2)}), \tag{5}$$

$$\hat{y}_{\mathbf{A}}(i, j) = \text{Dec}(\mathbf{H}_{\mathbf{A}}^{(2)}), \tag{6}$$

where $\text{Dec}(\cdot)$ is the inner-product decoder (Hamilton *et al.*, 2017) defined as

$$\text{Dec}(\mathbf{U}) = \sigma(\mathbf{U}\mathbf{U}^\top) = \frac{1}{1 + \exp(-\mathbf{U}\mathbf{U}^\top)}. \tag{7}$$

Subsequently, we optimize the cross entropy (CE) loss between the above estimations and the observed SL interactions. Let $y_{ij} = 1$ if the gene pair $(i, j) \in \mathcal{O}^+$ is a known SL interaction; otherwise $y_{ij} = 0$.

Algorithm 1: Training of DDGCN

Input: adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ of the SL graph $\mathcal{G}$, dual-dropout coefficient α, the set of known gene pairs $\mathcal{O}^+$ and unknown pairs $\mathcal{O}^-$, maximum training epochs T, learning rate η.
Output: Optimal parameters $\mathbf{W}^{*(l)}$ for each layer.
$\mathbf{W}_0^{(1)}, \mathbf{W}_0^{(2)} \leftarrow$ random initialization;
$t \leftarrow 1$;
repeat
  Compute $\mathbf{H}_{\mathbf{I}}^{(2)}$ with Eq. (2) and $\mathbf{H}_{\mathbf{A}}^{(2)}$ with Eq. (4);
  Compute $L(\mathbf{H}_{\mathbf{I}}^{(2)}, \mathbf{H}_{\mathbf{A}}^{(2)}, \alpha, \mathcal{O}^+, \mathcal{O}^-)$ with Eq. (8);
  $\mathbf{W}_t^{(1)} \leftarrow \mathbf{W}_{t-1}^{(1)} - \eta \frac{\partial L}{\mathbf{W}^{(1)}}$;
  $\mathbf{W}_t^{(2)} \leftarrow \mathbf{W}_{t-1}^{(2)} - \eta \frac{\partial L}{\mathbf{W}^{(2)}}$;
  $t \leftarrow t + 1$;
until $t > T$ *or* L *is converged*;
return $\mathbf{W}_t^{(1)}, \mathbf{W}_t^{(2)}$

Thus, our overall loss function is as follows:

$$L = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \text{CE}(y_{ij}, \hat{y}_{\mathbf{I}}(i, j)) + \alpha \, \text{CE}(y_{ij}, \hat{y}_{\mathbf{A}}(i, j)), \tag{8}$$

where $\alpha > 0$ is a hyper-parameter controlling the trade-off between the two paths and we call it dual-dropout coefficient. In order to better deal with the data skewness, we follow (Liu *et al.*, 2018) to assign higher weights in the cross entropy to gene pairs with known interactions.

The loss function is differentiable and thus can be optimized using gradient-based approaches such as the Adam optimizer. We outline the training process of the proposed DDGCN in Algorithm 1. For the final prediction, we follow (Zhai and Zhang, 2015) to aggregate the two prediction scores $\hat{y}_{\mathbf{I}}(i, j)$ and $\hat{y}_{\mathbf{A}}(i, j)$ by using the geometric mean:

$$\hat{y}(i, j) = \sqrt[1+\alpha]{\hat{y}_{\mathbf{I}}(i, j) \times \hat{y}_{\mathbf{A}}(i, j)^{\alpha}}. \tag{9}$$

The final score $\hat{y}(\cdot, \cdot)$ is used to rank unknown gene pairs, so that the top gene pairs are more likely SL candidates.

3.6 Theoretical Analysis

Lastly, we further provide a theoretical justification for the design of the proposed DDGCN architecture.

First, in the following Theorem 1, we establish that the proposed two forms of dropout are equivalent to removing nodes and edges on the graph.

Theorem 1. *Eq. (1) and Eq. (3) are equivalent to removing nodes and edges randomly from the given graph $\mathcal{G} = (\mathcal{U}, \mathcal{E})$, respectively.*

Proof. First, we show that Eq. (1) is the random removal of nodes from $\mathcal{G}$. We begin with the j^{th} row of the identity matrix $\mathbf{I}$, i.e., $(f_1, f_2, \ldots, f_n)$, where $f_{i=j} = 1$, $f_{i \neq j} = 0$. We rewrite $\mathbf{I}'$ as the matrix $\mathbf{M}_{\mathbf{I}} \odot \mathbf{I}$ after dropout. Suppose $f'_{i=j} = 0$ in $\mathbf{I}'_j$, which means $\mathbf{I}'_j$ as well as the product $\mathbf{I}'_j\mathbf{W}^{(1)}$ are both zero vectors. Meanwhile, $\mathbf{I}'_j\mathbf{W}^{(1)}$ is the feature (embedding) vector of node j in the d_1-dimensional latent space, which will be aggregated to its neighbor by GCN. Given that $\mathbf{I}'_j\mathbf{W}^{(1)} = \mathbf{0}$, node j will not affect any of its neighbors during the aggregation, i.e., all its incident edges are removed from $\mathcal{G}$.

Second, we show the correspondence between Eq. (3) and the removal of edges. Let $\mathbf{A}'$ be $\hat{\mathbf{A}}\mathbf{A}$, which is the 1-and-2-hop neighborhood matrix of the original graph, given self-loops on every node in $\hat{\mathbf{A}}$. Thus, in the first layer, we are aggregating from both 1- and 2-hop neighbors, a variant

Downloaded from https://academic.oup.com/bioinformatics/article-abstract/doi/10.1093/bioinformatics/btaa211/5813330 by Korea national university of transportation user on 21 April 2020

of GCN. Again, the j^{th} row of $\mathbf{W}^{(1)}$ is the feature vector of node j. If the (i, j) element of $\mathbf{A}'$ becomes zero due to applying dropout to $\mathbf{A}$, node j's feature vector will not be aggregated to node i's representation, which is equivalent to removing the 1- or 2-hop interaction (edge) between node i and node j.

While Theorem 1 shows that our proposed operation over the identity matrix and the adjacency matrix are equivalent to the node dropout and edge dropout, it is also important to show that both forms of dropout are jointly applied to the same graph. To achieve this design, it is necessary for the dual paths in Fig. 2 to share parameters, as we establish below.

Theorem 2. *If the parameters $\mathbf{W}^{(l)}$ are not shared by both Eq. 1 and Eq. 3 in the first layer, or by both Eq. 2 and Eq. 4 in the second (or more) layer, then two forms of dropout are applied on two different graphs, respectively.*

Proof. Let us first consider the simpler case of using only one GCN layer. Suppose Eq. 1 and Eq. 3 employ different parameters $\mathbf{W}'^{(1)}$ and $\mathbf{W}''^{(1)}$ such that $\mathbf{W}'^{(1)} \neq \mathbf{W}''^{(1)}$. The feature (embedding) vector for each node is in fact a function of the parameters $\mathbf{W}'^{(1)}$ or $\mathbf{W}''^{(1)}$, and we thus can consider the j^{th} row of $\mathbf{W}'^{(1)}$ or $\mathbf{W}''^{(1)}$ represents the feature vector of node j. If $\mathbf{W}'^{(1)}$ and $\mathbf{W}''^{(1)}$ are different, the nodes on the two paths actually reside in different spaces, resulting in two different graphs even though the structure of these graphs are still the same. Therefore, to jointly apply both forms of dropout on the same graph, two paths with different forms of dropout should share the same parameters at this GCN layer.

Let us further consider the case of using a second or more layers of GCN. Without loss of generality, we present two layers here. In the second layer, assuming no activation function, we have

$$\mathbf{H}^{(2)} = \hat{\mathbf{A}}(\hat{\mathbf{A}}\mathbf{H}^{(0)}\mathbf{W}^{(1)})\mathbf{W}^{(2)},$$

where $\mathbf{H}_0 = \mathbf{I}$ or $\mathbf{A}$, corresponding to Eq. 2 or Eq. 4, respectively. It can be rewritten as

$$\mathbf{H}^{(2)} = (\hat{\mathbf{A}}\hat{\mathbf{A}}\mathbf{H}^{(0)})(\mathbf{W}^{(1)}\mathbf{W}^{(2)}),$$

where $\hat{\mathbf{A}}\hat{\mathbf{A}}\mathbf{H}^{(0)}$ captures graph topology and $\mathbf{W}^{(1)}\mathbf{W}^{(2)}$ can be treated as the feature matrix of nodes. Again, if Eq. 2 and Eq. 4 adopt different parameters $\mathbf{W}'^{(2)} \neq \mathbf{W}''^{(2)}$ in the second layer, they would operate on two different graphs where nodes reside in different feature spaces.

As a summary, the two theorems above provide a theoretical justification for our proposed architecture with two different forms of dropouts and the shared parameters.

4 Results

In this section, we investigate the empirical performance of the proposed DDGCN model, with both quantitative evaluations and qualitative case studies.

4.1 Experimental Setup

4.1.1 Datasets

SynLethDB (Guo *et al.*, 2016) is a comprehensive database for human SL interactions. Note that about 32% of SLs (6,346 pairs) in SynLethDB are predicted from computational methods, among which 5,218 SLs come from DAISY (Jerby-Arnon *et al.*, 2014) and 1,128 from text mining. To isolate the impact of these predicted SLs, we further constructed a reduced dataset, denoted as SynLethDB-NonPred, by removing those predicted SLs from SynLethDB. We test various algorithms on both the original dataset SynLethDB and the reduced dataset SynLethDB-NonPred. The statistics of the two datasets are summarized in Table 1. In particular, density is the ratio of non-zero elements in the adjacency matrix. We observe that both SynLethDB and SynLethDB-NonPred with low density are indeed very sparse as shown in Table 1.

Table 1. Statistics of the datasets used in our experiments.

	SynLethDB	SynLethDB-NonPred
# human genes	6,375	3,877
# SL pairs	19,677	13,331
Average degree	6.17	6.88
Density	0.0968%	0.177%

4.1.2 Baselines and model variants

We consider three categories of methods, namely matrix factorization methods (SL^{2}MF, SL^{2}MF*), graph-based methods (DeepWalk, LINE, VGAE, BLM-NII) and the variants of our proposed method (GCN(I), GCN(A), GCN(A+I)). These methods are summarized as follows.

- **SL^{2}MF**, **SL^{2}MF*** (Liu *et al.*, 2018) both predict the SL pairs based on logistic matrix factorization (LMF). SL^{2}MF* further integrates gene similarities based on GO annotations for learning the gene embeddings.
- **DeepWalk** (Perozzi *et al.*, 2014) is a graph embedding method that combines truncated random walk with skip-gram model.
- **LINE** (Tang *et al.*, 2015) preserves both the 1^{st} order and 2^{nd} order proximities to embed the graph.
- **VGAE** (Kipf and Welling, 2016) is a graph embedding framework based on variational auto-encode (VAE) and leverages GCN to generate the distribution of the latent variables for the nodes.
- **BLM-NII** (Mei *et al.*, 2013) presents a procedure called neighbor-based interaction-profile inferring (NII) within a bipartite local model (BLM) for link prediction in graphs.
- **GCN(I), GCN(A), GCN(A+I)** are variants of our DDGCN without Dual-Dropout. These three variants use identity matrix $\mathbf{I}$, adjacency matrix $\mathbf{A}$ and the sum of $\mathbf{A}$ and $\mathbf{I}$ as input to vanilla GCN (i.e., only one path with a single form of dropout), respectively.

Note that we focus on comparing the methods, which can *automatically* learn the gene embeddings from the known SL data. Meanwhile, supervised methods, e.g., DiscoverSL (Das *et al.*, 2018), Slant (Benstead-Hume *et al.*, 2019) and the method in (Li *et al.*, 2019), *manually* extract gene features from additional data sources, e.g., gene mutation, gene expression, copy number alternation (CNA), protein-protein interactions (PPI), GO, etc. For fair comparison, we thus do not include them in our experiments.

4.1.3 Implementation details

For DDGCN, we used two layers, and adopted the inverted dropout technique with uniform dropout probability $p = 0.5$ in different GCN layers. The dual-dropout coefficient α in Eq. 8 was set to 1. We further set d_1 and d_2, the dimensionality of the latent spaces in the first and second GCN layers, to 512 and 256, respectively. While these are our default settings, we further analyze their impacts on the performance in our sensitivity analysis. In addition, Kaiming initialization was used for weight initialization (He *et al.*, 2015). The learning rate η in the optimization algorithm was set to 0.01, without learning rate decay. We used the Adam optimizer with default parameters. The maximum number of training epochs t was set to 2000 and we would stop optimization if the rate of change in train loss was smaller than the threshold 10^{-5}. In the next section, we will discuss the impacts of the parameters above.

Downloaded from https://academic.oup.com/bioinformatics/article-abstract/doi/10.1093/bioinformatics/btaa211/5813330 by Korea national university of transportation user on 21 April 2020

Table 2. Performance comparison of various SL prediction algorithms under five-fold cross-validation.

	SynLethDB			SynLethDB-NonPred		
Method	AUROC	AUPR	F1	AUROC	AUPR	F1
SL^{2}MF	0.8052±0.0040	0.2652±0.0103	0.4203±0.0055	0.8361±0.0092	0.3600±0.0081	0.4934± 0.0030
SL^{2}MF*	0.8529±0.0048	0.2824±0.0143	0.4363±0.0048	0.8664±0.0076	0.4003±0.0068	0.5625±0.0023
BLM-NII	0.7124±0.0474	0.0011±0.0007	0.0018±0.0012	0.7686±0.0503	0.0004±0.0001	0.0008±0.0002
DeepWalk	0.5260±0.0244	0.0010±0.0011	0.0005±0.0001	0.5011±0.0150	0.0004±0.0003	0.0020±0.0004
LINE	0.7409±0.0049	0.0214±0.0026	0.0842±0.0091	0.7374±0.0041	0.0141±0.0017	0.0004±0.0001
VGAE	0.8601±0.0067	0.2667±0.0331	0.4663±0.0454	**0.8879**±0.0099	0.2228±0.0260	0.4081±0.0530
VGAE(3L)	0.8591±0.0050	0.2421±0.0205	0.4692±0.0179	0.8677±0.0164	0.2113±0.0433	0.4309±0.0817
GCN(I)	0.8399±0.0059	0.3183±0.0180	0.5055±0.0912	0.6943±0.0339	0.2840±0.0051	0.4614±0.0124
GCN(A)	**0.8811**±0.0047	0.2263±0.0292	0.4736±0.0458	0.8633±0.0662	0.3533±0.0930	0.4214±0.0849
GCN(A+I)	0.8799±0.0074	0.2854±0.0376	0.4942±0.0243	0.8705±0.0665	0.2484±0.1503	0.2888±0.1511
DDGCN	0.8782±0.0057	**0.3442**±0.0089	**0.5520**±0.0128	0.8415±0.0092	**0.4565**±0.0274	**0.6565**±0.0130

For the baselines, we tuned their settings empirically. Specifically, we set the importance weight c, λ, α and the embedding dimensionality d in SL^{2}MF and SL^{2}MF* as 50, 0.01, 1.0 and 50, respectively. Moreover, the number of nearest neighbors in GO graph k was set as 100 for SL^{2}MF*. In DeepWalk, the window size ω, the embedding dimensionality d, the number of walks per vertex γ and the walk length t were set as 5, 64, 10 and 40, respectively. For LINE, we fixed the total number of mini-batches T as 1000, the number of negative samples K as 5 and the embedding dimensionality d as 128. For the learning rate in LINE, we set the starting value $\eta_0 = 0.025$ and $\eta_t = \eta_0(1 - t/T)$. We adopted the inner product decoder on node embeddings generated by DeepWalk or LINE, to perform SL prediction. In BLM-NII, we set the combination weight $\alpha = 0.75$, and used the max function to integrate the prediction scores. For VGAE, we used same settings as our model. These settings largely align with the literature.

We employed five-fold cross-validation to evaluate the performance of various methods in two datasets. In particular, the observed SL pairs were randomly partitioned into five non-overlapping subsets with equal size. In each round, a subset of SL interactions were chosen for testing while the remaining four subsets were used as positive examples in model training. As evaluation metrics, we used the area under the ROC curve (AUROC) and the Precision-Recall curve (AUPR), as well as the best F_1 score achievable among all the points on the Precision-Recall curve (Fang *et al.*, 2012).

4.2 Performance Evaluation

We evaluate the empirical performance of DGGCN against both the state-of-the-art baselines and our model variants in Table 2.

4.2.1 Comparison with baselines

On SynLethDB, DDGCN significantly and consistently outperforms all the baselines as shown in Table 2. In particular, SL^{2}MF* represents the state of the art in SL prediction prior to this work, and DDGCN is able to achieve further improvements by 3.0%, 21.9% and 26.5% on the three metrics, respectively. On SynLethDB-NonPred, our DDGCN significantly outperforms all the baselines in terms of AUPR and F1 score, while it achieves a somewhat lower but still competitive AUROC. It is worth noting that on highly skewed problems including SL prediction, it has been established previously (Davis and Goadrich, 2006) that AUPR is a more informative metric than AUROC in assessing the predictive power.

As our SL graph is very sparse, DeepWalk and LINE cannot handle the severe imbalance issue of the SL data and thus achieve low performance for SL prediction as shown in Table 2. In addition, the two-layer VGAE achieves better performance than its three-layer version VGAE(3L) in terms of AUROC and AUPR. This might be due to overfitting with three layers and the over-smoothing effect (Li *et al.*, 2018) in deep convolutional structures.

4.2.2 Comparison with model variants

On both SynLethDB and SynLethDB-NonPred, DDGCN performs better than its variants in terms of AUPR and F1 scores as shown in Table 2, and remains competitive in AUROC. Again, on highly skewed problems, AUPR is a more informative metric than AUROC (Davis and Goadrich, 2006). We can further draw the following two conclusions. First, both the fine- and coarse-grained dropouts are useful, as DGGCN outperforms both GCN(I) and GCN(A). Secondly, our architecture with two paths is able to effectively leverage the dual dropouts, given DGGCN's better performance than GCN(A+I)'s. In GCN(A+I), both forms of dropout are also adopted, but only one path is used without decoupling them, since dropping a node is equivalent to dropping all edges of the node. In contrast, DGGCN decouples the dual dropouts via two paths to further reduce overfitting, and thus obtains superior results.

4.3 Model Analysis

4.3.1 Architecture analysis

As Fig. 2 illustrates, DDGCN consists of two paths for the dual dropouts. While our theoretical analysis already shows the necessity of parameter sharing in the two paths for joint dropouts on the same graph, we further present the empirical evidence supporting this architecture.

Specifically, if l^{th} layer does not share the GCN parameters, we have $\mathbf{W}'^{(l)} \neq \mathbf{W}''^{(l)}$ $(l = 1, 2)$. Therefore, we can have 4 different models according to 4 different scenarios as follows,

- "Not shared": $\mathbf{W}'^{(1)} \neq \mathbf{W}''^{(1)}, \mathbf{W}'^{(2)} \neq \mathbf{W}''^{(2)}$;
- "Share 1st layer only": $\mathbf{W}'^{(1)} = \mathbf{W}''^{(1)}, \mathbf{W}'^{(2)} \neq \mathbf{W}''^{(2)}$;
- "Share 2nd layer only": $\mathbf{W}'^{(1)} \neq \mathbf{W}''^{(1)}, \mathbf{W}'^{(2)} = \mathbf{W}''^{(2)}$;
- "Share all": $\mathbf{W}'^{(1)} = \mathbf{W}''^{(1)}, \mathbf{W}'^{(2)} = \mathbf{W}''^{(2)}$.

As shown in Fig. 3, the proposed "Share all" design in DDGCN attains the highest scores on all metrics. Moreover, sharing the first-layer parameters is crucial, which is intuitive since the first graph convolution operation already captures the most important structural information. Sharing the second layer under the condition of sharing the first layer can further improve the performance.

Downloaded from https://academic.oup.com/bioinformatics/article-abstract/doi/10.1093/bioinformatics/btaa211/5813330 by Korea national university of transportation user on 21 April 2020

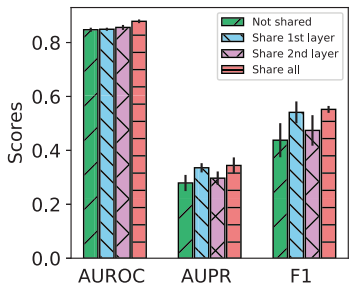
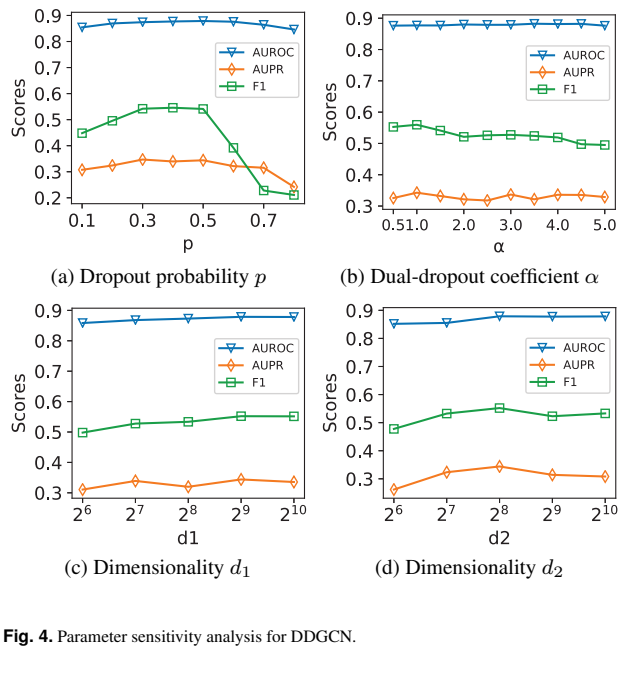


Fig. 3. Impact of parameter sharing in DDGCN on SynLethDB.



(a) Dropout probability p (b) Dual-dropout coefficient α

(c) Dimensionality d_1 (d) Dimensionality d_2

Fig. 4. Parameter sensitivity analysis for DDGCN.

4.3.2 Parameter sensitivity

We present the sensitivity analysis for the parameters in our DDGCN, including the dropout probability p, the dual dropout coefficient α in the loss function, and the dimensionality of the latent spaces d_1, d_2.

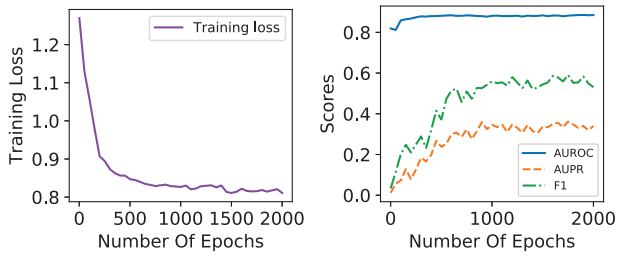
In Fig. 4 (a), the performance of DDGCN gradually increases with larger dropout probability p until before 0.6. Generally, a wide range of values $p \in [0.3, 0.5]$ give stable and optimal results. In Fig. 4 (b), the dual-dropout coefficient α controls the contributions from two paths. Typically, a balanced choice such as $\alpha = 1$ is desirable. Fig. 4 (c) and (d) show the impact of dimensions d_1 and d_2, respectively. We vary one of them and fix the other at its default value. As can be seen, very small dimensions do not perform well, whereas moderately large values are more favorable and stable.

4.3.3 Convergence analysis

As shown in Fig. 5 (a), we find that the training loss falls rapidly within the first 500 epochs, and begins to converge gradually in about 1000 epochs. Fig. 5 (b) shows a similar overall picture, where the three metrics on the test data also begin to converge around 1000 epochs. The similar trends on training loss and test performance shows that the proposed method is able to minimize the overfitting problem for SL prediction.

4.4 Qualitative Case Studies

We further train our DDGCN using all the SL pairs in SynLethDB and predict novel SLs from the unknown pairs. We rank these unknown pairs



(a) Learning curve. (b) Performance on test data.

Fig. 5. Convergence analysis for DDGCN.

Table 3. Top predicted SL pairs with literature support.

#	Rank	Gene 1	Gene 2	PubMed ID
1	61	RAD51	TP53	23728082
2	491	CDK4	TP53	23728082
3	508	RRM1	BCL2	30682083
4	539	USP1	TP53	23284306
5	555	KRAS	BRCA1	24104479
6	578	PIK3CA	BRCA1	26427375
7	592	CHEK1	MTOR	28319113
8	620	MTOR	BRCA2	27438146
9	655	TYMS	PARP1	30682083
10	691	MTOR	MAPK1	17429401

based on their scores predicted by DDGCN, and subsequently search for the top pairs in biomedical literature. In particular, among the top 1000 pairs (out of more than 23 million unknown pairs), we find 28 pairs verified by existing publications. Table 3 shows the first 10 out of these 28 pairs and the last column provides the PubMed ID of the publications that support our prediction.

As shown in Table 3, some pairs have been validated by wet-labs, e.g., row 4 (USP1 and TP53) is validated by RNAi screening (Xie *et al.*, 2012), and row 5 (KRAS and BRCA1) is validated by shRNA screening (Vizeacoumar *et al.*, 2013). Meanwhile, rows 1 and 2 are predicted by network centrality-based method (Kranthi *et al.*, 2013), row 3 is based on text mining (Heinzel *et al.*, 2019), and row 6 is predicted by mutual exclusivity information (Srihari *et al.*, 2015). However, these methods (centrality, mutual exclusivity or text mining) predict SLs from the angles which are orthogonal from GCN. Therefore, we still consider that they provide reasonable supports for our prediction. Taking the first SL pair as an example, RAD51 and TP53 is reported in (Kranthi *et al.*, 2013) (i.e., PubMed ID: 23728082). In fact, TP53 is a well-known tumor suppressor and its mutations are universal across cancers. It also induces cell arrest, apoptosis and DNA repair. Meanwhile, RAD51 is known to be involved in the homologous recombination and DNA repair. It is thus reasonable to predict a SL between them for cancer therapy as they have back-up functions related to DNA repair.

4.5 Discussions

SynLethDB-NonPred dataset achieves a higher data quality by removing all the predicted SLs (6,346 SLs) from SynLethDB, as predicted SLs usually have low quality. We would think data quality is the main reason that our DDGCN can achieve better performance on SynLethDB-NonPred than SynLethDB as shown in Table 2. To further support this claim, we generate another dataset, denoted as SynLethDB-80%, by removing 20% of SLs with the lowest scores in SynLethDB. In SynLethDB-80%, there are 5,294 human genes in the database, with 15,742 known SL pairs. The

Downloaded from https://academic.oup.com/bioinformatics/article-abstract/doi/10.1093/bioinformatics/btaa211/5813330 by Korea national university of transportation user on 21 April 2020

average degree and density of SynLethDB-80% are 5.95 and 0.1124%, respectively.

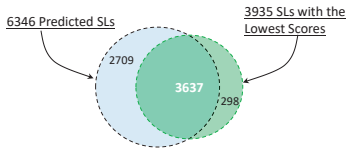


Fig. 6. Predicted SLs vs. 20% SLs with lowest confidence scores.

As mentioned above, we removed 3,935 SLs with lowest scores from SynLethDB to obtain the SynLethDB-80% dataset. Therefore, SynLethDB-80% has a higher quality than SynLethDB. Meanwhile, SynLethDB-NonPred has a higher quality than SynLethDB-80%, as SynLethDB-80% still contains 2,709 predicted SLs as shown in Fig. 6. We then ran both DDGCN and SL^{2}MF* on these 3 datasets and their performance under 5-fold cross validation is shown in Tables 4 and 5. We can observe that the performance of both DDGCN and SL2MF* (i.e., AUPR and F1) increases when we use the SL dataset with better quality. And, these results would support that the computationally predicted SLs in the SynLethDB somehow hurt the predictive power of DDGCN, as well as SL^{2}MF*. Please refer to our supplementary materials for more analysis on model performance against input data quality.

Table 4. Performance of DDGCN on three datasets.

Datasets	AUROC	AUPR	F1
SynLethDB	0.878±0.006	0.344±0.009	0.552±0.013
SynLethDB-80%	0.846±0.010	0.424±0.044	0.641±0.042
SynLethDB-NonPred	0.842±0.009	0.457±0.027	0.657±0.013

Table 5. Performance of SL^{2}MF* on three datasets.

Datasets	AUROC	AUPR	F1
SynLethDB	0.853±0.005	0.282±0.014	0.436±0.005
SynLethDB-80%	0.850±0.005	0.341±0.015	0.509±0.006
SynLethDB-NonPred	0.866±0.008	0.400±0.007	0.563±0.002

Note that our DDGCN predicts novel SL pairs based solely on the known SL pairs without using other data sources for genes. In fact, it is very important to integrate other data sources for the SL prediction task (Das *et al.*, 2018; Liany *et al.*, 2019). It can help to improve the model performance and we have shown that GO data further helps to improve the performance of DDGCN in our supplementary materials. In addition, data integration can enable the prediction of SL pairs for the genes outside the SL graph and provide mechanistic insights for predicted SL relationships. In this work, our focus is to develop a new GCN-based framework with dual dropouts. In the future, we will integrate multi-omics data (e.g., PPI, gene expression, GO, etc.) in a GCN framework to better address the SL prediction task.

5 Conclusion

In this paper, we propose a novel Dual-Dropout Graph Convolutional Network (DDGCN) for predicting synthetic lethality (SL) in human cancers. Specifically, we propose and integrate the dual forms of coarse- and fine-grained dropouts in our approach—the former can efficiently and systematically enforce dropout at gene level, whereas the latter further fine-tune the dropout at interaction level. We further present a theoretical analysis to justify the proposed architecture. The proposed approach DDGCN achieves promising results on a comprehensive human SL database, which not only provides an effective solution to deal with sparse SL interactions, but also fills the gap between graph neural networks and SL prediction.

References

Apaolaza, I., San José-Eneriz, E., Tobalina, L., Miranda, E., Garate, L., Agirre, X., Prósper, F., and Planes, F. J. (2017). An in-silico approach to predict and exploit synthetic lethality in cancer metabolism. *Nature communications*, **8**(1), 1–9.

Benstead-Hume, G., Chen, X., Hopkins, S. R., Lane, K. A., Downs, J. A., and Pearl, F. M. (2019). Predicting synthetic lethal interactions using conserved patterns in protein interaction networks. *PLoS computational biology*, **15**(4), e1006888.

Cai, H., Zheng, V. W., and Chang, K. C.-C. (2018). A comprehensive survey of graph embedding: Problems, techniques, and applications. *TKDE*, **30**(9), 1616–1637.

Cao, S., Lu, W., and Xu, Q. (2015). GraRep: Learning graph representations with global structural information. In *CIKM*, pages 891–900.

Chan, D. A. and Giaccia, A. J. (2011). Harnessing synthetic lethal interactions in anticancer drug discovery. *Nature reviews Drug discovery*, **10**(5), 351.

Das, S., Deng, X., Camphausen, K., and Shankavaram, U. (2018). Discoversl: an r package for multi-omic data driven prediction of synthetic lethality in cancers. *Bioinformatics*, **35**(4), 701–702.

Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and roc curves. In *ICML*, pages 233–240.

Fang, Y., Hsu, B.-J. P., and Chang, K. C.-C. (2012). Confidence-aware graph regularization with heterogeneous pairwise features. In *SIGIR*, pages 951–960.

Gal, Y. and Ghahramani, Z. (2016). A theoretically grounded application of dropout in recurrent neural networks. In *NIPS*, pages 1019–1027.

Grover, A. and Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *KDD*, pages 855–864.

Guo, J., Liu, H., and Zheng, J. (2016). Synlethdb: synthetic lethality database toward discovery of selective and sensitive anticancer drug targets. *Nucleic Acids Research*, **44**(Database Issue), D1011–D1017.

Hamilton, W. L., Ying, R., and Leskovec, J. (2017). Representation learning on graphs: Methods and applications. *IEEE Data Engineering Bulletin*.

Han, P., Yang, P., Zhao, P., Shang, S., Liu, Y., Zhou, J., Gao, X., and Kalnis, P. (2019). GCN-MF: Disease-gene association identification by graph convolutional networks and matrix factorization. In *KDD*, pages 705–713.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, pages 1026–1034.

Heinzel, A., Marhold, M., Mayer, P., Schwarz, M., Tomasich, E., Lukas, A., Krainer, M., and Perco, P. (2019). Synthetic lethality guiding selection of drug combinations in ovarian cancer. *PloS one*, **14**(1), e0210859.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Huang, Y.-a., Hu, P., Chan, K. C., and You, Z.-H. (2019). Graph convolution for predicting associations between mirna and drug resistance. *Bioinformatics*, **btz621**.

Iglehart, J. D. and Silver, D. P. (2009). Synthetic lethality–a new direction in cancer-drug development. *New England Journal of Medicine*, **361**(2), 189.

Jacunski, A., Dixon, S. J., and Tatonetti, N. P. (2015). Connectivity homology enables inter-species network models of synthetic lethality. *PLoS computational biology*, **11**(10), e1004506.

Jerby-Arnon, L., Pfetzer, N., Waldman, Y. Y., Mcgarry, L., James, D., Shanks, E., Seashoreludlow, B., Weinstock, A., Geiger, T., and Clemons, P. A. (2014). Predicting cancer-specific vulnerability via data-driven detection of synthetic lethality. *Cell*, **158**(5), 1199–1209.

Kipf, T. N. and Welling, M. (2016). Variational graph auto-encoders. In *NIPS Workshop on Bayesian Deep Learning Workshop*.

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *ICLR*.

Kranthi, T., Rao, S., and Manimaran, P. (2013). Identification of synthetic lethal pairs in biological systems through network information centrality. *Molecular bioSystems*, **9**(8), 2163–2167.

Krueger, D., Maharaj, T., Kramár, J., Pezeshki, M., Ballas, N., Ke, N. R., Goyal, A., Bengio, Y., Courville, A. C., and Pal, C. J. (2017). Zoneout: Regularizing rnns by randomly preserving hidden activations. In *ICLR*.

Downloaded from https://academic.oup.com/bioinformatics/article-abstract/doi/10.1093/bioinformatics/btaa211/5813330 by Korea national university of transportation user on 21 April 2020

Li, J., Lu, L., Zhang, Y.-H., Liu, M., Chen, L., Huang, T., and Cai, Y.-D. (2019). Identification of synthetic lethality based on a functional network by using machine learning algorithms. *Journal of cellular biochemistry*, **120**(1), 405–416.

Li, Q., Han, Z., and Wu, X.-M. (2018). Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*.

Liany, H., Jeyasekharan, A., and Rajan, V. (2019). Predicting synthetic lethal interactions using heterogeneous data sources. *Bioinformatics*, page btz893.

Liu, Y., Wu, M., Miao, C., Zhao, P., and Li, X.-L. (2016). Neighborhood regularized logistic matrix factorization for drug-target interaction prediction. *PLoS Computational Biology*, **12**(2), e1004760.

Liu, Y., Wu, M., Liu, C., Li, X.-L., and Zheng, J. (2018). Sl2mf: Predicting synthetic lethality in human cancers via logistic matrix factorization. *IEEE/ACM Trans. on Computational Biology and Bioinformatics*, **PP**(99), 1–1.

Mei, J.-P., Kwoh, C.-K., Yang, P., Li, X.-L., and Zheng, J. (2013). Drug-target interaction prediction by learning from local information and neighbors. *Bioinformatics*, **29**(2), 238–245.

O'Neil, N. J., Bailey, M. L., and Hieter, P. (2017). Synthetic lethality and cancer. *Nature Reviews Genetics*, **18**(10), 613–623.

Ou, M., Cui, P., Pei, J., Zhang, Z., and Zhu, W. (2016). Asymmetric transitivity preserving graph embedding. In *KDD*, pages 1105–1114.

Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). Deepwalk: Online learning of social representations. In *KDD*, pages 701–710.

Qiu, J., Dong, Y., Ma, H., Li, J., Wang, K., and Tang, J. (2018). Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *WSDM*, pages 459–467.

Semeniuta, S., Severyn, A., and Barth, E. (2016). Recurrent dropout without memory loss. In *COLING*, pages 1757–1766.

Sinha, S., Thomas, D., Chan, S., Gao, Y., Brunen, D., Torabi, D., Reinisch, A., Hernandez, D., Chan, A., Rankin, E., and Bernards, R. (2017). Systematic discovery of mutation-specific synthetic lethals by mining pan-cancer human primary tumor data. *Nature Communications*, **8**(1), 15580.

Srihari, S., Singla, J., Wong, L., and Ragan, M. A. (2015). Inferring synthetic lethal interactions from mutual exclusivity of genetic events in cancer. *Biology direct*, **10**(1), 57.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, **15**(1), 1929–1958.

Sun, M., Zhao, S., Gilvary, C., Elemento, O., Zhou, J., and Wang, F. (2019). Graph convolutional networks for computational drug development and discovery. *Briefings in bioinformatics*.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. (2015). Line: Large-scale information network embedding. In *WWW*, pages 1067–1077.

Vizeacoumar, F. J., Arnold, R., Vizeacoumar, F. S., Chandrashekhar, M., Buzina, A., Young, J. T., Kwan, J. H., Sayad, A., Mero, P., Lawo, S., *et al.* (2013). A negative genetic interaction map in isogenic cancer cell lines reveals cancer cell vulnerabilities. *Molecular systems biology*, **9**(1).

Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., and Fergus, R. (2013). Regularization of neural networks using dropconnect. In *ICML*, pages 1058–1066.

Wang, H., Huang, H., Ding, C., and Nie, F. (2013). Predicting protein–protein interactions from multimodal biological data sources via nonnegative matrix tri-factorization. *Journal of Computational Biology*, **20**(4), 344–358.

Xie, L., Gazin, C., Park, S. M., Zhu, L. J., Debily, M.-a., Kittler, E. L., Zapp, M. L., Lapointe, D., Gobeil, S., Virbasius, C.-M., *et al.* (2012). A synthetic interaction screen identifies factors selectively required for proliferation and tert transcription in p53-deficient human cancer cells. *PLoS genetics*, **8**(12), e1003151.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2019). How powerful are graph neural networks? In *ICLR*.

Xuan, P., Pan, S., Zhang, T., Liu, Y., and Sun, H. (2019). Graph convolutional network and convolutional neural network based method for predicting lncrna-disease associations. *Cells*, **8**(9), 1012.

Zhai, S. and Zhang, Z. (2015). Dropout training of matrix factorization and autoencoder for link prediction in sparse graphs. In *SDM*, pages 451–459.

Zhang, F., Wu, M., Li, X.-J., Li, X.-L., Kwoh, C. K., and Zheng, J. (2015). Predicting essential genes and synthetic lethality via influence propagation in signaling pathways of cancer cell fates. *Journal of bioinformatics and computational biology*, **13**(03), 1541002.

Downloaded from https://academic.oup.com/bioinformatics/article-abstract/doi/10.1093/bioinformatics/btaa211/5813330 by Korea national university of transportation user on 21 April 2020