1 Introduction

Complex biological systems cannot be composed by a large number of genes acting independently, but rely on the interactions between genes which can be further classified into enhancing and suppressive effects (Dhabhar, 2009). The suppressive effects characterize the situation that, when mutations occur simultaneously in a pair of genes, some important functions will be deactivated which seriously decrease cell viability, whereas the mutation in single gene might not affect the cell viability. A common type of suppressive effect is synthetic lethality (SL; Dobzhansky, 1946), which has been a promising strategy for cancer medicine (Ashworth et al., 1997). Hence, SL is a gold mine of anti-cancer drug targets. High-throughput wet-lab screening methods, including chemical libraries (Simons et al., 2001), pooled RNAi screening (Luo et al., 2009) and CRISPR-based genome editing technology (Du et al., 2017) have been used to find SLs, but they are thwarted by various barriers such as high cost, off-target effects, and batch effects (Liu et al., 2020). Hence, it is compelling to devise efficient computational methods to complement the downsides of the wet-lab screening techniques.

A spectrum of computational methods has been proposed for SL prediction. These methods can be categorized into three classes. The first class involves simulating in silico knockouts using metabolic network models. Folger et al. (2011) proposed to characterize SLs by modeling effects of the single- and double-knockouts of candidate genes in those networks. The second class, referred to as knowledge-oriented methods, is mostly conducted by feature engineering with domain-specific knowledge. To predict SL pairs, these methods identify SL gene pairs. High-throughput wet-lab screening methods, including chemical libraries (Simons et al., 2001), pooled RNAi screening (Luo et al., 2009) and CRISPR-based genome editing technology (Du et al., 2017) have been used to find SLs, but they are thwarted by various barriers such as high cost, off-target effects, and batch effects (Liu et al., 2020). Hence, it is compelling to devise efficient computational methods to complement the downsides of the wet-lab screening techniques.

A spectrum of computational methods has been proposed for SL prediction. These methods can be categorized into three classes. The first class involves simulating in silico knockouts using metabolic network models. Folger et al. (2011) proposed to characterize SLs by modeling effects of the single- and double-knockouts of candidate genes in those networks. The second class, referred to as knowledge-oriented methods, is mostly conducted by feature engineering with domain-specific knowledge. To predict SL pairs, these methods
employ network topology features such as graph centrality (Kranthi et al., 2013), network flow (Zhang et al., 2015), connectivity homology (Jacunski et al., 2015) and features derived from genomic data including somatic copy number alteration (Jerby-Aronn et al., 2014), short hairpin RNA profiles (Jerby-Aronn et al., 2014), and gene expression profiles (Jerby-Aronn et al., 2014). However, the two classes mentioned above rely heavily on the metabolic network models, domain knowledge and genomic data, without fully exploiting the valuable information of known SL pairs.

To exploit the exiting SL data, the third class of methods apply machine learning algorithms, where features are engineered based on domain knowledge and heuristic functions. Paladugu et al. (2008) proposed to train a support vector machine (SVM) for SL prediction where the features were extracted from a protein–protein interaction (PPI) network. MetaSL (Wu et al., 2014) integrated 17 features and weighted outputs of 10 classifiers to predict SLs. Aside from the traditional machine learning methods, graph representation learning approaches have been proposed, which mostly adopt an encoder–decoder paradigm. In this paradigm, an encoder tries to map the nodes into a low-dimensional embedding, whereas a decoder takes the embedding and utilizes it to reconstruct the node structure in the original graph (Hamilton, 2020), thereby recovering missing links. This paradigm can be generalized to matrix factorization (MF)-based methods and graph neural network (GNN)-based methods. Those methods use distinct designs of encoders, but resemble each other in the choices of decoders (mostly taking the form of inner product predictor or its normalized variants). MF methods adopt a MF encoder. SL2MF (Liu et al., 2020) proposed a MF encoder which decomposes the SL matrix, gene ontology (GO) similarity matrix, and PPI matrix to a low-dimensional latent space. GRSMF (Huang et al., 2019) introduced a self-representative MF encoder which focuses on learning a representation matrix from known SL pairs and further integrates the functional similarities among genes derived from GO. Liany et al. (2020) adopted Collective Matrix Factorization (CMF) based methods to integrate data from heterogeneous sources to predict SLs.

The MF-based encoders are just shallow embedding methods, which simply optimize a unique embedding vector for each node, without sharing any parameter between nodes or leveraging node features (Hamilton, 2020). GNN, a state-of-the-art framework for deep learning on graphs, enhances the aforementioned methods by adopting a different embedding strategy. GNN defines a message-passing (MP) process on the original graph, i.e. at each iteration, each node aggregates all the embeddings from its local neighborhood as a message which is combined with its previous embedding to generate a new embedding. Based on GNN, Cai et al. (2020) adopted a novel regularization technique called dual dropout to address the sparsity of SL networks. However, the existing GNN-based methods often regard each SL pair as an independent sample, and make no attempt to take their underlying biological mechanisms into account. However, some shared factors (such as biological processes, pathways, cellular components etc.) might latently invalidate the assumption of independency. For instance, poly (adenosine diphosphate-ribose) polymerase 1 (PARP1) and breast cancer 1 (BRCA1) are a famous SL gene pair, leading to the first clinically approved SL-based cancer drug, PARP inhibitor (Lord and Ashworth, 2017). PARP1 and BRCA1 are both key players in DNA repair process. Meanwhile, ATM and TP53 are another widely known SL pair (Kwok et al., 2016), and ATM is also a key instrument in DNA repair process (Sanders et al., 2020). Here, the DNA repair process might be the common mechanism underlying the two SL pairs.

A subset of existing methods [e.g. SVM, random forests (RFs), SL2MF and GRSMF] have injected some genomic and proteomic data to facilitate the SL prediction, and the results of these studies have underscored the significance of integrating additional information. Meanwhile, GNN-based methods can also encode such information as input features. However, these methods extracted features manually based on domain knowledge and some features might be left out. Therefore, to attain a more comprehensive set of features to improve the performance of SL prediction, we need a new method capable of automatic knowledge integration and feature extraction.

Knowledge graphs (KGs) are a type of multi-relational graph, where nodes and edges have different types. A KG is denoted by $G = (V, E)$, where edges in set $E$ are defined as triplets $e = (h, r, t)$ indicating a particular relationship $r \in R$ between two nodes (Hamilton, 2020). By incorporating a KG into a GNN, one can mitigate the aforementioned independency issue by directly introducing those latent factors as nodes in the graph. Lin et al. (2020) proposed an end-to-end knowledge GNN (KGNN) and achieved good performance in drug–drug interaction prediction.

Here, we propose a novel KGNN-based method for SL prediction, named KG4SL, which utilizes KG MP as a back-end. We approach the independency issue by injecting various factors including biological processes, diseases, compounds etc. that could be pertinent to SL, into our KG. Our model comprises three parts. In the first part, we derive a gene-specific subgraph from the original KG for each gene. In the second part, we conduct MP on the gene-specific subgraph, to automatically associate genes with factors that could be decisive in identifying an SL pair. In the third part, we define a decoder to reconstruct gene–gene similarity in a supervised fashion. To the best of our knowledge, this is the first framework to integrate KG with GNN for SL prediction. We compared our model with 10 state-of-the-art methods for SL prediction, and our model outperformed all the baselines in area under ROC curve (AUC), area under precision-recall curve (AUPR) and F1. Another contribution of our work is that we studied the impact of KG, which suggests that introducing a KG combined with MP process in GNN can significantly improve the performance of SL prediction.

## 2 Materials and methods

In this section, we first introduce the data and the problem of SL prediction. Then, we present the details of the proposed KG4SL model.

### 2.1 Data Collection

SynLethDB (http://synlethdb.sist.shanghaitech.edu.cn/v2/#/; Guo et al., 2016) is a comprehensive database of synthetic lethal gene pairs. Its latest version includes a set of 36,402 human SL pairs, as well as a KG with 11 kinds of entities and 24 kinds of relationships as shown in Table 1. SynLethDB also includes negative SL pairs, i.e. Non-SL and synthetic rescue pairs. However, there are much less known negative SL pairs than known positive ones. To obtain a balance between the positive and negative samples, we randomly pick up unknown pairs as negative pairs so that there are equal numbers of positive and negative SL pairs. Hence, the final SL dataset contains 72,804 gene pairs between 10,004 genes.

The KG, denoted as SynLethKG, includes 24 kinds of relationships between 11 entities. Among 24 kinds of relationships, 16 of them are related to genes directly, e.g. (gene, regulates, gene), (gene, interacts, gene) and (gene, co-varies, gene). And the other 8 relationships are associated with drug and compounds. Besides, 7 out of 11 kinds of entities are directly related to genes, i.e. pathway, cellular component, biological process, molecular function, disease, compound and anatomy. They are in the format of (gene, relationship, entity). These entities can be reached from genes in one hop, whereas the other three kinds of entities (pharmacologic class, side effect)}

| SL data                      | No. of genes | 10 004 |
|-----------------------------|--------------|--------|
| No. of interactions         | 72 804       |
| Density                     | 0.14%        |
| SynLethKG                   | No. of entity types | 11        |
| No. of relationship types   | 24           |
| No. of nodes                | 54 012       |
| No. of edges                | 2 231 921    |
and symptom) can be reached from genes in two hops. After removing isolated nodes, the final graph of SynLethKG contains 34,012 nodes and 2,231,921 edges as shown in Table 1. Tables 2 and 3 show the details about the entities and relationships in SynLethKG. Users can access the SynLethKG through searching the names of the entities that they want to study in SynLethDB.

The 24 relationships in Table 3 describe the features of genes, drugs and compounds. These relationships are collected from Genbank, GO, Drugbank, DrugCentral, PubMed, Bgee, String, LINCS L1000, SIDER4, STARGEO, Uberon and BioGRID. The specific number of each type of relationship and the number of associated nodes are also shown in Table 3. Besides, the types of the entities in SynLethKG and the number of each entity are shown in Table 2.

### 2.2 Problem statement

Formally, the SL data can be modeled as a matrix $S \in (0,1)^{n \times n}$, where $n$ is the number of genes involved in the SL pairs. In this SL matrix $S$, an entry $s_{ij}$ is 1 if there is an SL interaction between gene $i$ and gene $j$, and 0 otherwise. Note that gene pairs with entry value 0 are unknown pairs, some of which could be potential SL pairs not yet discovered. The KG SynLethKG is denoted by $G = (V, E)$, which contains a set of entities $V$ and a set of relationships $E$. Each edge in the KG is defined as a triplet $t = (h, r, t)$, which shows a relationship of type $r$ between head entity $h$ and tail entity $t$, where $h, t \in V$ and $r \in R$.

Given the SL matrix $S$ and the KG $G$, the problem we aim to solve is to predict the SL relationship between gene $e_i$ and gene $e_j$. To achieve this goal, we propose a GNN-based model to learn a scoring function $s_{ij} = F((i, j); W, A, b)$ that estimates how likely gene $i$ and gene $j$ are an SL pair, where $W$, $A$ and $b$ denote the learnable parameters in function $F$.

### 2.3 Overview of KG4SL

The overall framework of KG4SL is laid out in Table 1. KG4SL utilizes a GNN to encode the gene features from KG for SL prediction in three steps. First, we derive a gene-specific weighted subgraph for each SL-related gene from the KG. Specifically, the weight of every edge is defined by a gene-specific relation scoring function to depict the importance of the relation for its target gene. Second, we design an aggregation layer to update the representation for a given gene by aggregating the representations of its neighbors in the gene-specific weighted subgraph. Third, we assign a score for each gene pair computed by the normalized inner product based on their learned representations. Next, we introduce these three steps in details.

#### 2.3.1 Gene-specific weighted subgraph

Given an SL-related gene, we first construct a weighted subgraph from the KG. Identifying relevant nodes and determining the edge weights are two key operations to construct the gene-specific weighted subgraph.

#### 2.3.2 Aggregation of node representations

For any central entity $e$ in the subgraph of gene $e_i$, we aggregate the representations of all its picked neighbors to update its own representation. To show the topological neighborhood structure of entity $e$ in the KG, we compute the weighted average combination of $e$’s neighborhood:

$$ e_{P(e)} = \sum_{e' \in P(e)} \hat{o}_{e, e'} e' $$

where $i$ is the representation of entity $e$, gene $e_i$ and gene $e_j$ are a pair in the SL matrix, and $\hat{o}$ is the normalized gene-relation score by applying a softmax function:

$$ \hat{o}_{e, e'} = \frac{\exp(o_{e, e'}^{c})}{\sum_{e' \in P(e)} \exp(o_{e, e'}^{c})} $$

Assume that $e$ is the central node/entity and $N(e)$ is the set of neighbors of $e$ (i.e. entities directly connected to $e$). In SynLethKG, the size of $N(e)$ varies greatly among the entities. For example, network hubs may have thousands of relations, whereas some nodes are less studied and thus have a limited number of neighbors. In this work, we sample a fixed number of $k$ neighbors for each entity to characterize its local structure and we repeat this process for $H$ hops ($H \geq 1$). In particular, if a node has less than $k$ neighbors, we sample duplicates, i.e. a neighbor may be sampled more than once. The set of sampled $k$ neighbors is denoted as $P(e)$. An example of 2-hop subgraph with neighbor sampling size $k = 4$ in each hop can be seen in Figure 1.

In a gene-specific subgraph, we can assign different weights for edges to describe the importance of the relations. For an SL pair $(e_i, e_j)$, the weight for an edge $r_{e_i e_j}$ in $e_i$‘s subgraph is computed by $o_{e_i e_j}^{c} = g(e_i, r_{e_i e_j})$, where $e$ is one of the entities in the subgraph of $e_i$, and $e' \in P(e)$. Besides, $e_i$ and $r_{e_i e_j}$ are the feature embeddings of gene $e_i$ and relation $r_{e_i e_j}$, respectively. $g$ is an inner product function. Here, $o_{e_i e_j}^{c}$ characterizes the importance of relation $r_{e_i e_j}$ to gene $e_i$. 

### Table 2. Details about the entities in SynLethKG

| Type                  | Sample size |
|-----------------------|-------------|
| Cellular component    | 1670        |
| Gene                  | 67,062      |
| Biological process    | 12,703      |
| Side effect           | 5726        |
| Molecular function    | 3203        |
| Pathway               | 2069        |
| Disease               | 137         |
| Compound              | 2595        |
| Pharmacologic class   | 377         |
| Anatomy               | 402         |
| Symptom               | 453         |

### Table 3. Details about the relationships in SynLethKG

| Type                                      | No. of edges | No. of source nodes | No. of target nodes |
|-------------------------------------------|--------------|---------------------|---------------------|
| (Anatomy, downregulates, gene)            | 31           | 4                   | 24                  |
| (Anatomy, expresses, gene)                | 617,175      | 241                 | 23,881              |
| (Anatomy, upregulates, gene)              | 26           | 5                   | 22                  |
| (Compound, binds, gene)                   | 16,323       | 1922                | 2,306               |
| (Compound, causes, side effect)           | 139,428      | 1079                | 5,702               |
| (Compound, downregulates, gene)           | 21,526       | 747                 | 2,847               |
| (Compound, palliates, disease)            | 384          | 215                 | 50                  |
| (Compound, resembles, compound)           | 6,266        | 1034                | 1,055               |
| (Compound, treats, disease)               | 752          | 385                 | 77                  |
| (Compound, upregulates, gene)             | 19,200       | 721                 | 3,205               |
| (Disease, associates, gene)               | 24,328       | 135                 | 6,672               |
| (Disease, downregulates, gene)            | 7,616        | 44                  | 5,730               |
| (Disease, localizes, anatomy)             | 3,373        | 123                 | 398                 |
| (Disease, presents, symptom)              | 3,401        | 122                 | 427                 |
| (Disease, resembles, disease)             | 404          | 100                 | 98                  |
| (Disease, upregulates, gene)              | 7,730        | 44                  | 5,614               |
| (Gene, covaries, gene)                    | 62,987       | 9174                | 9,706               |
| (Gene, interacts, gene)                   | 148,379      | 9,633               | 14,275              |
| (Gene, participates, biological process)   | 619,712      | 16,608              | 12,703              |
| (Gene, participates, cellular component)   | 97,652       | 11,916              | 16,170              |
| (Gene, participates, molecular function)   | 110,042      | 14,404              | 3,203               |
| (Gene, participates, pathway)             | 57,441       | 11,519              | 2,069               |
| (Gene, regulates, gene)                   | 267,791      | 4649                | 7,105               |
| (Pharmacologic class, includes, compound) | 1205         | 377                 | 837                 |
Fig. 1. The framework of KG4SL. The workflow of KG4SL can be divided into three modules, including gene-specific weighted subgraph module, aggregation module and score computation module. (1) Gene-specific weighted subgraph: First, we construct a weighted subgraph from the KG. (2) Aggregation: Second, for each SL pair, we select the entities and relationships that are directly related to the nodes. Besides, we believe the biological information can flow between nodes through edges. Thus, we also aggregate the information of indirectly connected entities and relationships. Considering the problem of computing power, only two layers of entities and relationships are included. (3) Score computation: Third, the results of aggregation for two genes are used to compute their SL score through inner product. The loss function of KG4SL is composed of two kinds of losses, i.e. the base loss computed based on the truth label and the gene–gene score, and the L2 loss computed using the entity embedding, relation embedding and aggregation weights.

After obtaining the picked neighbors’ representation \( e_{\phi(e)} \) of a central entity in one hop, similar to (Wang et al., 2019a), it integrates the entity representation \( e \) into a single vector to update \( e \):

\[
e(b + 1) = \phi(W(e(b) + e_{\phi(e)}) + b),
\]

where \( W \) and \( b \) are the linear transformation weight and bias, respectively, and \( \phi \) is an activation function such as ReLU. After aggregating neighbors’ information through \( H \) hops, the final feature representation of gene \( e \) is \( e[H] \). \( e \) is obtained in the same way.

### 2.3.3 SL prediction score

Finally, by passing information from two subgraphs of KG, we attain the final representations \( e_i \) and \( e_j \) for the two genes in the SL matrix. The predicted interaction probability between gene \( e_i \) and gene \( e_j \) is calculated by \( \hat{s}_{ij} = \phi(f(e_i, e_j)) \), where \( f \) is the inner product function and \( \phi \) is a sigmoid function, squashing the output to a range between 0 and 1. Furthermore, this link prediction can be viewed as a binary classification problem, by setting the threshold to 0.5. \( \hat{s}_{ij} \) is 1 or 0, which indicates whether an SL relation exists between a candidate pair of genes.

### 2.4 Overall loss and optimization

Two kinds of losses are designed for our model, including base loss and L2 loss. The base loss \( J \) is computed through cross-entropy of the truth label and the predicted label for the edges, represented as follows:

\[
J = \max(\hat{s}_{ij}, 0) - \hat{s}_{ij} \times s_{ij} + \log(1 + \exp(\max(-\hat{s}_{ij}, 0))).
\]

where \( \hat{s}_{ij} \) is the predicted label and \( s_{ij} \) is the truth label for the edge. We also add an L2-regularizer defined as:

\[
\|\Gamma\| = \frac{\|e\| + \|r\| + \|W\|}{2},
\]

where \( \| \cdot \| \) represents the L2 norm for entity embedding, relation embedding and aggregation weights.

The final loss combines the two kinds of loss functions described above as follows:

\[
\min_{W, A, b} \ell = \min_{W, A, b} \sum_{ij} J + b\|\Gamma\|,
\]

where \( A \) is the trainable weighted matrix in which each element represents the gene-relation score and L2 weight \( z \) is a balancing hyperparameter. Here \( z \) was set to 0.0039. The first term corresponds to the part of GNN that learns the linear transformation weight \( W \), gene-relation score weight \( A \) and bias \( b \) simultaneously. The second term added the L2-regularizer. Adam algorithm is used to minimize the final loss and the learning rate is set to 0.002. The framework of KG4SL is outlined in Algorithm 1.

### 3 Results

In this section, we first introduce the state-of-the-art baseline methods and their implementation details, and then we compare our model with the baselines, followed by an analysis of the influence of the KG. The KG4SL model was implemented with Python 3.6 and Tensorflow 1.15.0. We adopt AUC, AUPR and F1 as the evaluation metrics.
We compare KG4SL with the following baselines:

### 3.1.1 Baselines

- **SL2MF** (Liu et al., 2020) integrates gene similarities based on GO biological pathway annotations with SL matrix to predict SL pairs.
- **GRSMF** (Huang et al., 2019) is a graph regularized self-representative MF model which also uses known SL pairs and GO-based gene similarities to predict SL pairs.
- **HOPE** ( Ou et al., 2016) is scalable to preserve high-order proximity of graphs and capable of capturing the asymmetric transitivity.
- **DeepWalk** (Perozzi et al., 2014) is a graph embedding method which uses short random walks (RWs) to learn representations for nodes in graphs.
- **Node2Vec** (Grover and Leskovec, 2016) also learns feature representations for nodes in graphs but adds flexibility in exploring neighborhoods.
- **LINE** (Tang et al., 2015) uses an effective edge-sampling method for model inference and preserves both the first- and second-order proximities by a fine-grained objective function.
- **Convolutional network** (GCN) (Kipf and Welling, 2016) is the most popular GNN architecture, which employs the symmetric-normalized aggregation as well as the self-loop update approach.
- **GraphSAGE** (Hamilton et al., 2017) introduces the idea of generalized neighborhood aggregation.
- **GAT** (Velickovic et al., 2017) introduces the attention mechanisms to GNN.

**Algorithm 1 KG4SL**

**Input:** SL matrix $S$; KG $G(V_a, V_b)$; neighborhood field $P(e_i)$; hyper-parameters $x, d, k, b$ and epoch; **Output:** $\hat{S}$

1: **Initialization:**
2: entity embedding matrix $W_d$
3: relation embedding matrix $W_r$
4: step $\leftarrow 0$
5: while step $< $ epoch do
6:   for $(e_i, e_j) \in S$ do
7:     $G_a([i, j] \rightarrow S)$ -- Gene-specific Subgraph$(e_i)$
8:     $G_a([i, j] \rightarrow S)$ -- Gene-specific Subgraph$(e_j)$
9:   for $m \in [i, j]$ do
10:      $e[0] \leftarrow G_a[m][0]$
11:     for $m \in 1, 2, ..., H$ do
12:      for $e \in G_a[m][b]$ do
13:         $e_p = P(e \rightarrow i - 1) - \frac{\sum e \in P(e)}{\sum e \in P(e)}$
14:         $e[b] \leftarrow e[P(e \rightarrow i - 1) + e_p[b - 1] + e]$; $e[b]$;
15:     end for
16:   end for
17:   $\hat{S}_{ij} \leftarrow \phi(f(e_i, e_j))$
18:   Compute the loss $\ell$;
19:   Compute the predicted probability $\hat{S}$;
20: step $\leftarrow$ step + 1;
21: end while

### 3.1 Performance evaluation

#### 3.1.1 Baselines

We compare KG4SL with the following baselines:

- **SL2MF** (Liu et al., 2020) integrates gene similarities based on GO biological pathway annotations with SL matrix to predict SL pairs.
- **GRSMF** (Huang et al., 2019) is a graph regularized self-representative MF model which also uses known SL pairs and GO-based gene similarities to predict SL pairs.
- **HOPE** (Ou et al., 2016) is scalable to preserve high-order proximity of graphs and capable of capturing the asymmetric transitivity.
- **DeepWalk** (Perozzi et al., 2014) is a graph embedding method which uses short random walks (RWs) to learn representations for nodes in graphs.
- **Node2Vec** (Grover and Leskovec, 2016) also learns feature representations for nodes in graphs but adds flexibility in exploring neighborhoods.
- **LINE** (Tang et al., 2015) uses an effective edge-sampling method for model inference and preserves both the first- and second-order proximities by a fine-grained objective function.
- **Convolutional network** (GCN) (Kipf and Welling, 2016) is the most popular GNN architecture, which employs the symmetric-normalized aggregation as well as the self-loop update approach.
- **GraphSAGE** (Hamilton et al., 2017) introduces the idea of generalized neighborhood aggregation.
- **GAT** (Velickovic et al., 2017) introduces the attention mechanisms to GNN.

**Algorithm 1 KG4SL**

**Input:** SL matrix $S$; KG $G(V_a, V_b)$; neighborhood field $P(e_i)$; hyper-parameters $x, d, k, b$ and epoch; **Output:** $\hat{S}$

1: **Initialization:**
2: entity embedding matrix $W_d$
3: relation embedding matrix $W_r$
4: step $\leftarrow 0$
5: while step $< $ epoch do
6:   for $(e_i, e_j) \in S$ do
7:     $G_a([i, j] \rightarrow S)$ -- Gene-specific Subgraph$(e_i)$
8:     $G_a([i, j] \rightarrow S)$ -- Gene-specific Subgraph$(e_j)$
9:   for $m \in [i, j]$ do
10:      $e[0] \leftarrow G_a[m][0]$
11:     for $m \in 1, 2, ..., H$ do
12:      for $e \in G_a[m][b]$ do
13:         $e_p = P(e \rightarrow i - 1) - \frac{\sum e \in P(e)}{\sum e \in P(e)}$
14:         $e[b] \leftarrow e[P(e \rightarrow i - 1) + e_p[b - 1] + e]$; $e[b]$;
15:     end for
16:   end for
17:   $\hat{S}_{ij} \leftarrow \phi(f(e_i, e_j))$
18:   Compute the loss $\ell$;
19:   Compute the predicted probability $\hat{S}$;
20: step $\leftarrow$ step + 1;
21: end while

**3.1.2 Implementation details**

All the baselines were evaluated at 5-fold cross validation which makes the best use of the available data. We viewed the input graph as an unweighted and undirected graph. For SL2MF and GRSMF, we utilized all the default parameters in the origin papers. For HOPE, the parameter beta was set to 0.02. For RW-based methods, the number of walks to start at each node was set to 5, the length of the RW start at each node was set to 10, the window size was set to 3. For node2vec, $p$ and $q$ were both set to 1, which respectively control how fast the walk explores and how fast the walk leaves the neighborhood of starting. For LINE, the alpha was set to 0.1 and other was set to 2. For DDGCN, we use the same setting of the origin paper. For GCN, GAT and GraphSage, we all used two convolution layers, and the dimensionality of the latent spaces in the first and second layers were chosen to be 5 and 16. The number of training epochs was decided via the early stopping strategy. KG4SL was evaluated using the ratio of training, validation and testing data as 8:1:1. In order to improve the stability of results, we randomly split dataset into 10 pieces and took one of them as the testing set. Early stopping strategy is used to control the number of training epochs.

**3.1.3 Comparison with baselines**

On SynLethKG, KG4SL outperforms all baselines as shown in Table 4. When compared with the second best model GRSMF, KG4SL improves the performance on AUC, AUPR and F1 by 3.11%, 2.16% and 6.4%, respectively.

In general, GNN-based models achieve better performance than shallow embedding methods like MF-based and RW-based models. This may because GNN-based models can learn from the similarity between SLs and enrich the embedding of genes for SL prediction. DDGCN represents the state-of-art model for SL prediction, and it achieves the best performance among the GNN-based baselines. MF-based method GRSMF is second only to KG4SL, which shows that the combination of GO gene similarity information and self-representation matrix decomposition is very effective for SL prediction. The performance of KG4SL is even higher, which shows that learning gene representations from the KG including GO information and other gene features can further improve SL prediction.

**3.2 Model Analysis**

#### 3.2.1 Parameter sensitivity

We present the sensitivity analysis for some key hyperparameters in our KG4SL, including the neighbor sampling size $k$ and the dimension of entity embedding $d$, as shown in Figure 2.

First, we change the number of samples for neighbor $k$ and observe the model performance. KG4SL achieves the best AUC, F1 and AUPR when the neighbor sampling size $k = 64$. When sampling more neighbors with higher value of $k$, the information sampled may become redundant, and thus the model performance slightly decreases when $k = 128$. Next, we also investigate the influence of the dimension of embedding $d$. The KG4SL model already has a good performance when the dimension of embedding is 256. Too large dimension of embedding is a burden on memory and computation. Eventually, we set the neighbor sampling size as 64 and the dimension of embedding as 256 for our KG4SL model.
Table 4. Metrics of KG4SL against baselines in AUC, AUPR and F1

| Categories | Methods     | AUC       | AUPR       | F1       |
|------------|-------------|-----------|------------|----------|
| MF         | SL^2MF      | 0.7811 ± 0.0035 | 0.8635 ± 0.0021 | 0.7464 ± 0.0074 |
|            | GRSMF       | 0.9184 ± 0.0039 | 0.9362 ± 0.0023 | 0.8339 ± 0.0049 |
|            | HOPE        | 0.7776 ± 0.0005 | 0.7410 ± 0.0006 | 0.7089 ± 0.0010 |
| RW         | DeepWalk    | 0.8451 ± 0.0024 | 0.8600 ± 0.0013 | 0.7562 ± 0.0027 |
|            | node2vec    | 0.8362 ± 0.0010 | 0.8523 ± 0.0014 | 0.7503 ± 0.0031 |
|            | LINE        | 0.8233 ± 0.0028 | 0.8327 ± 0.0023 | 0.7380 ± 0.0056 |
| GNN        | GCN         | 0.8329 ± 0.0172 | 0.8727 ± 0.0110 | 0.8508 ± 0.0136 |
|            | GraphSAGE   | 0.8398 ± 0.0291 | 0.8775 ± 0.0188 | 0.8569 ± 0.0236 |
|            | GAT         | 0.7914 ± 0.0182 | 0.8462 ± 0.0103 | 0.8152 ± 0.0129 |
|            | DDGCN       | 0.8491 ± 0.0106 | 0.8998 ± 0.0056 | 0.8154 ± 0.0074 |
|            | KG4SL       | 0.9470 ± 0.0003 | 0.9564 ± 0.0005 | 0.8877 ± 0.0017 |

Note: The best results for each index are in bold.

Fig. 2. Parameter sensitivity analysis with varying $k$ and $d$ for KG4SL. Left: AUC, AUPR, and F1 at different neighbor sampling sizes $k$, ranging from 2 to 128. Right: AUC, AUPR and F1 at different node embedding dimensions $d$, ranging from 8 to 512.

3.2.2 Convergence analysis

With the above parameters set, we observe the convergence of the model. Figure 3 displays the change of loss and three metrics with the increase of epochs. The blue-dotted, red-dashed and green dash-dot lines represent the metrics of training data, validation data and testing data, respectively. The orange line shows the change of loss. We find that loss falls rapidly within the first 10 epochs and begins to converge gradually at the 20th epoch. Under the constraint of the $L_2$-regularizer, loss converges to 0.3111 and the results of the three metrics in the training set, validation set and test set have the similar variation trend which shows that the proposed method can alleviate the problem of overfitting.

3.3 Impact of KG

While automatically integrating the KG into the node feature construction is crucial in our work, we wonder whether the KG is really important for the task of SL prediction. To investigate this problem, we test the SL prediction performance with and without the SynLethKG separately.

Table 5 shows the prediction performance of several machine learning models on SynLethKG, SL graph and the combination of both. KG-based methods intend to learn low-dimensional embeddings of entities and relations in SynLethKG automatically. We take TransE (Bordes et al., 2013), a popular unsupervised KG embedding learning method as an example, which is trained based solely on SynLethKG. We also compare the contributions of a single SL graph, which is called SL-based method, whose performance is exactly that of GCN in Table 4. Then, a combination of the KG- and SL-based method, named ‘TransE + GCN’, is further tested. From the first three rows in Table 5, we can observe that additional information from the KG guides the model to achieve better performance than using KG or SL graph alone. TransE, trained on the KG only, reports an AUC score of 0.5870 and an AUPR of 0.6100, which are the lowest values among the three models. GCN, operating on the SL graph directly, leverages Xavier’s uniform distribution (Glorot and Bengio, 2010) as the initial node features, obtains AUC score of 0.8329 and AUPR score of 0.8727. The models that inspect the KG and SL graph together outperform either of the above. All the evidence supports that the KG information can help with SL prediction.

After that, an ensemble learning method named RF (Breiman, 2001), which also uses the information extracted from KG and SL,
is selected to be compared with 'TransE + GCN'. The difference between RF and 'TransE + GCN' is whether the features are extracted automatically. The features of RF should be carefully selected manually, whereas the gene embeddings are automatically extracted from TransE and fed into GCN to generate SL prediction results. Here, RF uses six features, namely, minTriangles, maxTriangles, minCoefficient, maxCoefficient, sp and sl. For each gene pair, minTriangles and maxTriangles reflect the max and min numbers of triangles that each gene forms. The minCoefficient and maxCoefficient reflect the min and max likelihood that the neighbors of these two genes are connected. sp is a Boolean value that represents whether the two genes are in the same community detected by the label propagation algorithm. sl means whether two genes are in the same community detected by Louvain algorithm (Blondel et al. 2008). The results show that 'TransE + GCN' achieves slightly higher AUC and slightly lower AUPR than RF.

Comparing all these models with KG4SL which is an end-to-end model using the information extracted from KG and SL automatically, KG4SL yields the top AUC of 0.9470 and AUPR of 0.9564. This signifies the benefit of adding a suitable KG in SL prediction.

Furthermore, to qualitatively interpret the above models' learning abilities, we draw the link features extracted from these models. First, we fix the dimension of node features to 256. Next, the features for each node of an SL pair in test data are concatenated together, representing the link embedding between them. Then, the high-dimensional feature vectors are mapped into a 2D space by using visualization technique t-SNE (Van der Maaten and Hinton, 2008). As Figure 4 shows, orange dots denote there is an SL relation between a pair of genes. TransE or GCN alone cannot distinguish the two types of links well as they only capture information from the KG or SL interaction, whereas KG4SL can distinguish the two link embeddings, demonstrating its ability to learn from both the SL network and the KG.
Dobzhansky,T. (1946) Genetics of natural populations. xiii. recombination and variability in populations of drosophila pseudoobscura. Genetics, 31, 269–290.

Du,D. et al. (2017) Genetic interaction mapping in mammalian cells usingcrop interfence. Nat. Methods, 14, 577.

Folger,O. et al. (2011) Predicting selective drug targets in cancer through metabolic networks. Mol. Syst. Biol., 7, 501.

Glorot,X. and Bengio,Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, pp. 249–256. Sardinia, Italy, May 13–15, 2010.

Grover,A. and Leskovec,J. (2016). node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 855–864. San Francisco, CA, USA, August 13–17, 2016.

Guo,J. et al. (2016) SynLethDB: synthetic lethality database toward discovery of selective and sensitive anticancer drug targets. Nucleic Acids Res., 44, D1011–D1017.

Hamilton,W. et al. (2017) Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems, Vol. 30, pp. 1024–1034. Long Beach, CA, USA, December 4–9, 2017.

Hamilton,W.L. (2020) Graph representation learning. In: Synthesis Lectures on Artificial Intelligence and Machine Learning, Vol. 14, pp. 1–159.

Hartwell,L.H. et al. (1997) Integrating genetic approaches into the discovery of anticancer drugs. Science, 278, 1064–1068.

Huang,J. et al. (2020) SL2MF: predicting Synthetic Lethality in Human Cancers of genetic interactions, functions, and pathways in yeast and human cancer. Cancer Inform., 13 (Suppl. 3), 71–80.

Lord,C.J. and Ashworth,A. (2017) Parp inhibitors: synthetic lethality in the clinic. Science, 355, 1132–1158.

Luo,J. et al. (2009) A genome-wide nsi screen identifies multiple synthetic lethal interactions with the ras oncogene. Cell, 137, 835–848.

O’Neil,N.J. et al. (2017) Synthetic lethality and cancer. Nat. Rev. Genet., 18, 613–623.

Ou,M. et al. (2016). Asymmetric transitivity preserving graph embedding. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1105–1114. San Francisco, CA, USA, August 13–17, 2016.

Paladugu,S.R. et al. (2008) Mining protein networks for synthetic genetic interactions. BMC Bioinformatics, 9, 1–14.

Perozzi,B. et al. (2014). Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 701–710. New York, NY, USA, August 24–27, 2014.

Sanders,J.T. et al. (2020) Radiation-induced DNA damage and repair effects on 3d genome organization. Nat. Commun., 11, 1–14.

Simons,A. et al. (2001) Establishment of a chemical synthetic lethality screen in cultured human cells. Genome Res., 11, 266–273.