Multi-class and Multi-task Strategies for Neural Directed Link Prediction

Claudio Moroni[1,2](✉), Claudio Borile[2], Carolina Mattsson[2], Michele Starnini[2,3], and André Panisson[2]

[1] Dedagroup, Turin, Italy
claudio.moroni@dedagroup.it
[2] CENTAI Institute, Turin, Italy
{claudio.moroni,claudio.borile,carolina.mattsson,michele.starnini, andre.panisson}@centai.eu
[3] Department of Engineering, Universitat Pompeu Fabra, 08018 Barcelona, Spain

Abstract. Link Prediction is a foundational task in Graph Representation Learning, supporting applications like link recommendation, knowledge graph completion, and graph generation. Graph Neural Networks (GNNs) have shown promising results in this domain and are widely used for learning graph representations from data. However, not all GNNs can represent edge direction and not all training strategies support the learning of directed graph representations. For this reason, Neural Directed Link Prediction (NDLP) has been divided into three sub-tasks. Models that perform well in distinguishing uncorrelated samples of positive and negative directed edges (the general DLP task) do not necessarily capture edge directionality and bidirectionality (two additional tasks). While many models can be trained to perform well on any of the sub-tasks, most fail to perform well across them all. In this work, we propose three novel training strategies that adress the three sub-tasks simultaneously and can be applied to any autoencoder-based GNN without motifying its architecture. Our first strategy, the Multi-Class Framework for Neural Directed Link Prediction (MC-NDLP) maps NDLP to a Multi-Class training objective. The second and third strategies adopt a Multi-Task perspective, either with a Multi-Objective (MO-NDLP) or a Scalarized (S-NDLP) approach. We show that these training strategies allow many models to achieve higher or more balanced performance across the three NDLP sub-tasks. The flexibility offered by our proposed training strategies provides a powerful means for improving the capabilities of NDLP models to advance the state of Neural Directed Link Prediction.

Keywords: Graph Machine Learning · Directed Link Prediction

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-662-72243-5_8.

1 Introduction

Graphs are a natural way to represent complex systems. Examples include social networks, financial transaction networks, power grids, and neuronal connectivity [1,16]. These systems can be modeled using different types of graphs, ranging from simple networks to more sophisticated structures like Knowledge Graphs [14], Dynamic Graphs [15], or Bipartite Graphs for recommender systems [24]. Given the widespread presence of graph structures, learning representations from graph data has grown increasingly important with core applications including node classification, link prediction and graph classification.

In this work, we focus on Graph Neural Networks (GNNs) as applied to link prediction [26] on directed graphs. Recently, GNN-based models, such as Graph Autoencoders, have been devised focusing on undirected [5,7,9,20,40] and directed graphs [12,21,31,39,42,43], establishing the field of Neural Link Prediction. These models have several important applications, including completing knowledge graphs [4], serving as baseline for deep graph generation [20,33] and pre-processing transaction networks [25]. However, the recent literature primarily focuses on undirected applications [22,27,37], with few studies mentioning directed cases [3]. Direction can be core to the application itself, in some domains. With citation graphs, for instance, citing and being cited have substantively different meanings. Moreover, incorporating edge direction has been shown to improve learning for node classification across different types of graphs [30]. Even so, the nuances of link prediction on directed graphs are often overlooked and it has been argued that this limits progress in this area [31].

Neural Directed Link Prediction (NDLP) requires a model that is capable of representing edge direction and a training strategy that effectively learns directionality. Not all GNNs can represent edge direction. Graph Autoencoders, for example, often use decoder implementations where probabilities for edges (u, v) and (v, u) are the same by design [31]. We refer to these models as NDLP-incapable. But even NDLP-capable models can fail to learn edge directionality when the training strategy is adapted from link prediction on undirected graphs, where, typically, models are trained and evaluated on random subsets of positive and negative undirected edges [5,9,13,20,35,40,41]. Now, on a sparse directed graph, it is statistically unlikely that a random subset of negative directed edges would include many reverse edges of randomly sampled positive directed edges. This allows models to ignore edge direction without incurring a penalty. Indeed, using uncorrelated samples of positive and negative directed edges to train and evaluate NDLP models [12,21,39,43] can lead to NDLP-incapable models performing deceptively well.

Training and evaluation for NDLP is more complex. Three sub-tasks have been devised in the recent literature to more comprehensively evaluate performance of directed link prediction [31,41,42,44]. The "General DLP" task is the classic adaptation of the approach used in the undirected case to the directed case. This is complemented with two other binary classification sub-tasks designed to test a model's ability to distinguish edge directions. Namely, the "Directional" and "Bidirectional" sub-tasks. Prior work has shown that

NDLP-capable models can learn to perform well on each of the three sub-tasks [31,41,42,44] and that there is a trade-off among them [42]. However, prior approaches do not consider training strategies that can adress these three sub-tasks simultaneously.

Here we propose three learning strategies to improve performance across NDLP sub-tasks, simultaneously. The first strategy, *Multi-Class Directed Link Prediction (MC-NDLP)*, maps directed link prediction to a four-class classification task. This framework distinguishes between unidirectional positives, unidirectional negatives, bidirectional positives, and bidirectional negatives, ensuring balanced contributions to the training loss. The other two approaches recognize the Multi-Task nature of directed link prediction, simultaneously training on simultaneously constructed *General DLP*, *Directional* and *Bidirectional* training sets. Drawing from the literature on Multi-Objective [6] and Scalarization [17] methods, we propose *Multi-Objective Directed Link Prediction (MO-NDLP)* and *Scalarization-based Directed Link Prediction (S-NDLP)* strategies to handle these tasks more effectively. Each of our three training strategies incentivize NDLP-capable models to learn directed representations that perform well across the three sub-tasks. We find that better training strategies can be as useful as better models in advancing the state of Neural Directed Link Prediction.

The remainder of this paper is organized as follows: Sect. 2 covers the background concepts and related work, highlighting their relevance to our approach. In Sect. 3, we detail the proposed multi-class and multi-task strategies. Section 4 outlines the experimental setup, describes the datasets and presents a performance comparison of various models across all strategies and datasets. Finally, Sect. 5 provides concluding remarks.

2 Background and Related Work

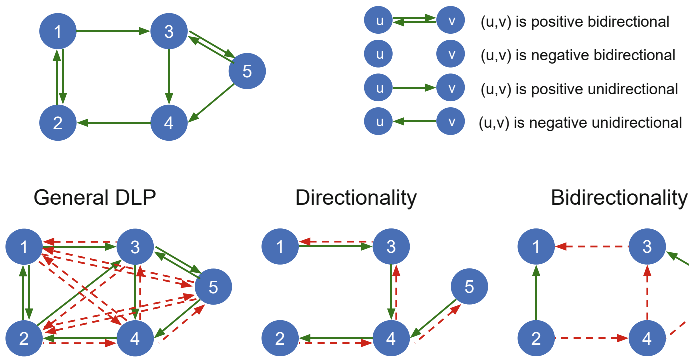
In this section, we introduce the notation and review key concepts and prior research relevant to Neural Directed Link Prediction (NDLP). We briefly discuss foundational work in Graph Neural Networks and their applications in undirected link prediction, then go on to examine approaches for incorporating edge direction, in order to highlight limitations and the need for training strategies that better incentivise models to encode directed representations.

2.1 Notation

Given a directed graph $G = (V, E)$ with $N = |V|$ nodes where $E \subseteq V \times V$, and given $u, v \in V$ we say that:

- (u, v) is *negative bidirectional* $\iff (u, v) \notin E \land (v, u) \notin E$;
- (u, v) is *negative unidirectional* $\iff (u, v) \notin E \land (v, u) \in E$;
- (u, v) is *positive unidirectional* $\iff (u, v) \in E \land (v, u) \notin E$;
- (u, v) is *positive bidirectional* $\iff (u, v) \in E \land (v, u) \in E$;

Moreover, we denote $A \in \{0, 1\}^{N \times N}$ as G's adjacency matrix, and $X \in \mathbb{R}^{N \times F}$ as node features.



(u,v) is positive bidirectional
(u,v) is negative bidirectional
(u,v) is positive unidirectional
(u,v) is negative unidirectional
General DLP
Directionality
Bidirectionality

Fig. 1. From a graph G (top panel), for each task definition, green edges are the positive class and red edges are the negative class. In *General DLP*, any absent directed edge can be selected as negative. For *Directionality* prediction, unidirectional edges are positives, with their inverses as negatives. For *Bidirectionality* prediction, one direction of bidirectional edges are positives, and the reverse of unidirectional edges are negatives.

2.2 Graph Neural Networks

Graph Neural Networks (GNNs) are currently the *de facto* standard approach for learning over graph data, with applications in the pharmaceutical industry, material science [28], time series [18], and anti-money laundering [2,8,19,36].

Message Passing Neural Networks (MPNNs) [10,11,34,38] constitute the most general framework for GNNs, and elaborate successive hidden representations for each node v by aggregating messages from its neighbors. Given $z_v^{(k)}$ as the k-th layer embedding of node v (with $z_v^{(0)}$ equalling v's feature vector x_v), GNNs compute the next layer embedding $z_v^{(k+1)}$ through the following relation:

$$z_v^{(k+1)} = f^{(k)}\left(m_s^k(z_v^{(k)}), A^{(k)}\left(\left\{m_n^{(k)}(z_u^{(k)}) | u \in N(v)\right\}\right)\right) \tag{1}$$

where $f^{(k)}, A^{(k)}, m_s^k$ and $m_n^{(k)}$ are, respectively, the layer-specific update function, the aggregation function, the self-information and message functions. $N(v)$ indicates the neighborhood of node v.

If we consider K layers in total, the last embedding of each node $z_v \equiv z_v^{(K)}$ can be used for downstream tasks such as node classification, link prediction, and graph classification. For link prediction, a *decoder* is a function that takes as input the representation of two nodes, z_u, z_v and outputs a normalized score representing the probability of an edge (u, v) being present. For the undirected case, the decoder can be the scalar product followed by a sigmoid function, $DEC(z_u, z_v) = \sigma(z_u \cdot z_v)$, or a neural network such as a Multi-Layer Perceptron (MLP). The scalar product, being symmetric in u and v, is an example of a decoder that leaves a model unable to represent directionality in its predictions.

2.3 Neural Directed Link Prediction

This work considers three sub-tasks for NDLP, as presented in [31]. Figure 1 shows a representation of the positive and negative classes that define the *General Directed Link Prediction (General DLP)* task, the *Biased Directional Negative Samples Link Prediction (Directional)* task, and the *Bidirectionality Prediction (Bidirectional)* task. These classes are the basis for selecting the samples of (directed) edges used in the most relevant recent literature for testing and evaluation of NDLP-capable models performing directed link prediction.

We focus on NDLP-capable models that use Graph Autoencoder techniques. [21] develops an extension of the Weisfeiler-Lehman kernel [32] which is then used as a basis to define a source/target-like [31] graph autoencoder for directed graphs; however, the model is only tested on the *General DLP* sub-task. [42] defines a GCN for directed graphs where the aggregation is performed by a complex, hermitian laplacian. This model is tested across all three tasks, although for each of them, a different parameter set is inferred. To the best of our knowledge, [31] devises the earliest GNN-based autoecoders for NDLP, and is among the first to highlight the intrinsic differences between the directed and undirected cases; Their work is also responsible for one of the earliest usage of the *General DLP, Directional* and *Bidirectional* sub-tasks in a neural setting. Similarly to [42], the authors of [31] do not find one parameter set for all three tasks for each model. [41] endows cluster information in node embeddings, but, similarly to [31], trains and tests over the *Directional* and *Bidirectional* tasks using two different training graphs. [44] produces unsupervised source/target node embeddings by adversarially training a neural pair of generator and discriminator on the graph topology; the final model performs simultaneously well on the *General DLP* and *Directional* task, but it is not evaluated on the *Bidirectional* task.

Although we will experiment with different models, it is not our aim to sponsor any model in particular. Rather, we propose learning strategies that can used with *any* NDLP-capable model to encourage encoding directionality and strike a better balance among the performances on sub-tasks of NDLP. Therefore our work is much more in the spirit of [23], a representation learning framework that argues in favor of masking compared to full-graph training. Unfortunately, [23] has been developed and tested on undirected graphs only, so its extension to directed graphs could be a future research avenue.

3 Strategies for Neural Directed Link Prediction

In the previous section, we discussed that NDLP has recently been described as three distinct sub-tasks: General DLP, Directional, and Bidirectional. Simultaneously addressing these requires more than the classic approach to training for link prediction. In this section, we formalize Multi-Class and Multi-Task training strategies that are in principle applicable to every NDLP-capable encoder-based MPNN and encourage such models to learn to encode directionality.

3.1 Multi-class Strategies for Neural Directed Link Prediction

As anticipated in Sect. 1, we consider here that the reason why prior works are generally not able to infer a single model that performs well on all three sub-tasks could be related to an unaddressed imbalance between unidirectional and bidirectional edges' contributions to the training loss. We note that this imbalance should be dealt with without compromising the reweighting between positive and negative edges. Therefore we propose to simultaneously balance positives vs negatives and unidirectional vs bidirectional edges using the following Multi-Class Neural Directed Link Prediction (*MC-NDLP*) strategy.

Given a GNN model that computes d_K-dimensional embeddings z_v, $\forall v \in V$, we may compute logits for each of the four classes listed in Sect. 2.1 by applying an MLP to the concatenation of the embeddings. The MLP must take $2d_K$ input dimensions and output 4 logits, and can be arbitrarily deep:

$$[\hat{l}^{nb}_{uv}, \hat{l}^{nu}_{uv}, \hat{l}^{pu}_{uv}, \hat{l}^{pb}_{uv}] = \mathrm{MLP}(z_u || z_v), \tag{2}$$

where $\hat{l}^{m}_{uv}, m \in \{nb, nu, pu, pb\}$ denote the model's output logits for the edge (u, v) being negative bidirectional, negative unidirectional, positive unidirectional, or positive bidirectional as defined in Sect. 2.1.

Notably, *MC-NDLP* is also compatible with any graph autoencoder that makes use of specific decoders which output only one logit $\hat{l}_{uv}$, that is, the model output for the presence of a directed edge (u, v) [21,31]. We can turn the standard binary classification task for NDLP into a 4-class classification task by transforming the output logit into a probability via e.g., a sigmoid $\hat{p}_{uv} = \sigma(\hat{l}_{uv})$ and defining:

$$\begin{aligned} \hat{p}^{nb}_{uv}, \hat{p}^{nu}_{uv}, \hat{p}^{pu}_{uv}, \hat{p}^{pb}_{uv} t = [&(1 - \hat{p}_{uv})(1 - \hat{p}_{vu}), \\ &(1 - \hat{p}_{uv})\hat{p}_{vu}, \\ &\hat{p}_{uv}(1 - \hat{p}_{vu}), \\ &\hat{p}_{uv}\hat{p}_{vu}]. \end{aligned} \tag{3}$$

We note that Eq. 3 assumes statistical independence between $\hat{p}_{uv}$ and $\hat{p}_{vu}$, which are both conditioned on A and X. In fact, most autoencoders model $\hat{p}_{uv} = \hat{P}(e_{uv}|A, X)$ where $e_{uv} = 1 \iff (u, v) \in E$ otherwise it equals 0 [20], and Eq. 3 naturally extends these univariate autoencoders. Please refer to Appendix for details.

We can then define a weighted Multi-Class cross-entropy loss function:

$$\mathcal{L}_{MC\text{-}NDLP}(\Theta) = -\sum_{c \in C} \sum_{uv \in T} w_{y_{uv}} \mathbb{I}(y_{uv} = c) \log(\hat{p}^{y_{uv}}_{uv}), \tag{4}$$

where $C = \{nb, nu, pu, pb\}$, T is the *General DLP* training set (see Sect. 3.3), $\mathbb{I}$ is the indicator function and $y_{uv} \in C$ is the ground-truth class of the edge (u, v), and $\hat{p}^{y_{uv}}_{uv}$ is the model's output probability of edge (u, v) belonging to the ground-truth class y_{uv}. The class weight is defined as

$$w_{y_{uv}} = \frac{n_x}{n_{y_{uv}}},$$

where n_x represents the number of samples in class x, where x is the most numerous class (usually nb), and $n_{y_{uv}}$ is the number of samples in class y_{uv}. As discussed in Sect. 1, this class reweighting mitigates the statistical imbalance between all four classes defined in Sect. 2.1.

3.2 Multi-task Strategies for Neural Directed Link Prediction

Multi-Task Learning (MTL) refers to scenarios where more than one objective function must be simultaneously optimized. It is more challenging compared to single-task learning, due to the various objectives having no a priori relative importance and generally competing against each other. To simultaneously exploit the *General DLP, Directional* and *Bidirectional* training sources of information, we devise a multi-task objective over the three sub-tasks, defined by the binary cross-entropy loss functions on *General DLP* $\mathcal{L}_G$, *Directional* $\mathcal{L}_D$ and *Bidirectional* $\mathcal{L}_B$.

Multi-Task learning is usually carried out in two ways:

– *Scalarization*: it prescribes to sum and weight the losses to reduce the optimization to the single-objective case:

$$\mathcal{L} = \alpha_G\mathcal{L}_G + \alpha_D\mathcal{L}_D + \alpha_B\mathcal{L}_B$$

The coefficients $\alpha_i,\ i = G, D, B$, can be either learned or heuristically set. Despite its simplicity, heuristic implementations of Scalarization have been proven to achieve competitive performance in real use cases [17]. In this work, we set them to the validation losses (normalized between 0 and 1) of the previous epoch, to favor generalization. We name this approach *S-NDLP*;

– *Multi-Objective*: it consists in finding a parameter update rule that ensures that all losses are diminished (or left unchanged) at each optimization step. Given a model f_Θ and the objectives $\{\mathcal{L}_i(\Theta)\}_{i=1}^{L}$ to be simultaneously optimized, we say that Θ_1 *dominates* $\Theta_2 \iff \mathcal{L}_i(\Theta_1) \leq \mathcal{L}_i(\Theta_2) \quad \forall i \in 1, ..., L$. Therefore, a solution Θ^* is *Pareto-optimal* if it is not dominated by any other solution. The set of non-dominated solutions is called the Pareto set $\mathcal{P}$. While many Gradient-based Multi-Objective optimization algorithms with guaranteed convergence on the Pareto set have been developed, we focus for simplicity on one of the first, MGDA [6]. This algorithm is based on the observation that given the gradients associated with the individual losses, the opposite of their shortest convex linear combination points in the direction where all losses either remain constant or diminish. We name this approach *MO-NDLP*.

3.3 Simultaneous Splits

Train, validation, and test sets are constructed from the positive and negative classes associated to each of the three NDLP sub-tasks (see Fig. 1). In [31], edges are randomly sampled from the positive classes to construct the validation and test sets separately for each sub-task. Since models are separately trained on each sub-task, in that work, the training sets can be defined as what remains when the positive samples for that sub-task are reserved. In our case, the models must be trained over the graph that remains when *all* the reserved edges are removed. In order to preserve as many edges as possible for training, sampling is done together. Specifically, for each dataset, a random 10% and 5% of all unidirectional edges are reserved for testing and validation, respectively. Moreover, a random 30% and 15% of (one direction of) all bidirectional edges are reserved for testing and validation, respectively.

Validation and test sets for the three sub-tasks are constructed using the reserved edges. All the sampled edges (unidirectional and bidirectional) are used as positives for the *General DLP* task, complemented with an equal number of randomly sampled absent directed edges as negatives. The sampled unidirectional edges are used as positives for the *Directional* task, complemented with their reverses as negatives. The randomly sampled bidirectional edges are used for the *Bidirectional* task, complemented with an equal number of the reverses of unidirectional edges randomly sampled from the remaining graph as negatives. To perform well on this task, the model must distinguish between directed edges whose reverse edge exists from those whose reverse edge does not.

Multi-class learning requires a single training set. As in the *General DLP* task of [31], and in the classic formulation of directed link prediction, the training set is constructed out of all remaining directed edges and all the absent edges in the incomplete training graph. In our case, these edges are sorted into the four classes defined in Sect. 2.1 for multi-class classification as described in Sect. 3.1.

Multi-task learning requires training sets for each sub-task. Here, we take the opportunity to construct our own versions of the training sets for the *Directional* and *Bidirectional* tasks. This is done to give the training set similar edge statistics as their respective validation and test sets. In particular: for the *Directional* task, the training set is composed of all the remaining unidirectional edges (as positives) and their reverses (as negatives). This is similar to the training set for the *General DLP* task, but with the bidirectional edges removed. Finally, for the *Bidirectional* task, one direction of all the remaining bidirectional edges are used, as positives, together with an equal amount of the reverses of unidirectional edges randomly sampled from the remaining graph, as negatives.

These modifications to the sampling described in [31] make simultaneous training possible while ensuring no overlap between train and test data.

4 Experiments

In this section, we evaluate the effectiveness of our proposed strategies through comparative experiments using well-known datasets and NDLP models. We aim to demonstrate the performance improvements of our approaches across multiple tasks and models, highlighting their ability to handle the challenges of edge directionality and directionality. All code and results are publicly available at https://github.com/ClaudMor/MTMC-NDLP.

4.1 Datasets

Our experiments are conducted on three publicly available datasets, each of which is a directed graph. As in [31], we consider two small citation networks (Cora and CiteSeer) and a larger hyperlink network (Google).

Table 1. Statistics of the datasets used.

Dataset	Nodes $\lvert V\rvert$	Edges $\lvert E\rvert$	Edges (undirected)	Reciprocity	Density $\lvert E\rvert/\lvert V\rvert^2$	Clustering
Cora	2,708	5,429	5,278	0.056	0.000741	0.131
CiteSeer	3,327	4,732	4,676	0.024	0.000428	0.074
Google	15,763	171,206	149,456	0.254	0.000689	0.343

Table 1 gives the key network statistics. The Cora and CiteSeer graphs have few bidirectional edges. For the Google graph, however, a randomly sampled directed edge will be part of a bidirectional edge around 25% of the time.

Despite the difference in size, the graph density is similar across the datasets. The local density is higher for the Google dataset, as measured by the average (directed) clustering coefficient. Edge weights and/or node attributes are not considered. In all experiments, we use one-hot encoding of the node IDs as node features [31], except for the MAGNET model for which we used in- and out-degree as prescribed by the authors [42]. While employing node IDs means the models are transductive, all the strategies can be extended to inductive settings by using other node features as appropriate; an avenue for future development.

4.2 Models

We evaluate our proposed training strategies using several NDLP-capable models from the literature, all of which follow the Graph Autoencoder paradigm. These include the Gravity-Inspired Graph Autoencoder (Gr-GAE) [31], Source/Target Graph Autoencoder (ST-GAE) [31], the DiGAE Directed Graph Auto-Encoder from [21], MAGNET [42] and our custom MLP-GAE, which uses a decoder based on concatenating the encoder outputs followed by a multilayer perceptron. We also consider a recently introduced undirected model, MPLP [7], that achieves the current state-of-the-art in undirected link prediction computing approximations of graph heuristics, following the idea introduced in [40]. Altough the original implementation is not natively NDLP-capable, we introduce a modification of the decoder (dMPLP) that allows for NDLP and achieves competitive results. Each model is tested under various experimental conditions to evaluate its performance within our proposed framework. The standard graph autoencoder (GAE) [20] is included to provide a baseline NDLP-incapable model. Further details on model implementations and settings are provided in Appendix.

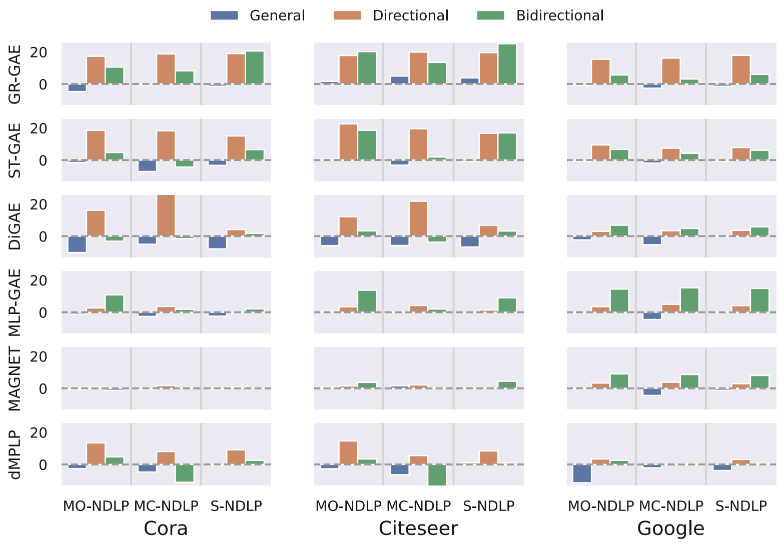


Fig. 2. Performance difference of each proposed strategy compared to the baseline strategy, measured in ROC-AUC (x100). Each bar represents the change in ROC-AUC - either an increase or decrease - when applying one of the proposed strategies to a specific sub-task, NDLP model, and dataset, relative to the same model's baseline performance. Scores are averaged over 5 runs. Error bars are omitted for visual clarity.

4.3 Experimental Settings

For the three tasks described in Fig. 1, under the sampling defined in Sect. 3.3, we measure ROC-AUC (Receiver Operating Characteristic - Area Under the Curve) to evaluate a model's ability to distinguish between classes, while AUPRC (Area Under Precision-Recall Curve) evaluates precision across different recall levels. We train the models according to the strategies defined in Sect. 3, as well as a *Baseline* strategy. Namely, the model is trained on the *General DLP* training set with rebalancing of positive and negative edges' contributions to training loss (Binary Cross Entropy). For each novel training strategy, we perform early stopping on the sum of ROC-AUC and AUPRC metrics over the *General DLP, Directional* and *Bidirectional* validation sets. Missing self-loops are always inserted for message passing, and they are also used as positive supervision samples in both *MO-NDLP* and *S-NDLP*, while they are treated as bidirectional negative supervision samples in *MC-NDLP*.

4.4 Results

The performance results are summarized in Table 2 (Cora dataset), Table 3 (CiteSeer dataset), Table 4 (Google dataset) and in Fig. 2. Performances are averaged over 5 random splits, keeping the same seed for all models. All ROC-AUC and AUPRC values are scaled by 100 for compactness and clearer visualization. In

Table 2. ROC-AUC and AUPRC test scores of various models on Cora Dataset, trained with the *Baseline* strategy and our proposed strategies.

MODEL	STRATEGY	GENERAL		DIRECTIONAL		BIDIRECTIONAL	
		ROC-AUC	AUPRC	ROC-AUC	AUPRC	ROC-AUC	AUPRC
GAE	BASELINE	84.6 ± 0.4	88.6 ± 0.3	50.0 ± 0.0	50.0 ± 0.0	62.4 ± 3.0	64.0 ± 3.1
GR-GAE	BASELINE	**89.2 ± 0.4**	**92.4 ± 0.2**	63.4 ± 2.5	61.5 ± 2.7	69.1 ± 3.1	66.5 ± 3.3
	MO-NDLP	84.5 ± 1.1	86.3 ± 1.1	80.6 ± 0.7	80.2 ± 0.9	79.6 ± 4.3	84.6 ± 3.5
	MC-NDLP	88.6 ± 0.4	90.0 ± 0.4	82.1 ± 0.5	**81.8 ± 0.7**	77.3 ± 2.2	76.3 ± 1.7
	S-NDLP	87.8 ± 0.6	89.5 ± 0.5	**82.3 ± 0.5**	81.6 ± 0.4	**89.6 ± 1.6**	**92.4 ± 1.1**
ST-GAE	BASELINE	**87.8 ± 0.7**	**90.1 ± 0.5**	60.8 ± 0.5	64.5 ± 0.6	74.6 ± 1.8	74.1 ± 2.2
	MO-NDLP	86.3 ± 0.5	86.2 ± 0.4	**79.3 ± 1.0**	80.0 ± 0.9	79.3 ± 0.5	79.5 ± 1.9
	MC-NDLP	80.7 ± 2.0	80.1 ± 2.1	79.0 ± 2.3	**81.6 ± 1.9**	70.3 ± 3.0	68.1 ± 2.1
	S-NDLP	84.5 ± 0.4	84.9 ± 0.7	75.8 ± 1.0	78.4 ± 0.9	**81.1 ± 0.9**	**80.4 ± 1.6**
DiGAE	BASELINE	**80.4 ± 1.1**	**85.3 ± 0.8**	57.5 ± 1.3	63.0 ± 1.4	70.4 ± 2.2	68.6 ± 1.2
	MO-NDLP	70.2 ± 3.8	72.6 ± 3.6	73.6 ± 5.4	76.0 ± 4.2	67.3 ± 4.6	69.6 ± 4.1
	MC-NDLP	75.4 ± 0.9	77.4 ± 1.0	**84.3 ± 0.6**	**85.4 ± 0.8**	68.9 ± 1.5	69.3 ± 1.1
	S-NDLP	72.5 ± 4.0	77.4 ± 4.4	61.6 ± 1.3	69.2 ± 1.4	**72.1 ± 5.6**	**74.4 ± 5.7**
MLP-GAE	BASELINE	**77.1 ± 0.9**	**78.2 ± 0.6**	90.7 ± 0.6	90.7 ± 0.6	69.9 ± 3.2	69.7 ± 3.7
	MO-NDLP	76.0 ± 0.8	76.4 ± 0.7	93.4 ± 0.6	93.5 ± 0.6	**80.7 ± 1.6**	**79.2 ± 2.4**
	MC-NDLP	74.5 ± 0.7	75.6 ± 0.7	**94.3 ± 0.6**	**94.4 ± 0.5**	71.7 ± 2.4	65.7 ± 1.8
	S-NDLP	74.7 ± 1.0	74.9 ± 0.9	90.5 ± 0.7	90.0 ± 0.9	72.0 ± 2.6	70.5 ± 2.9
MAGNET	BASELINE	**75.2 ± 1.4**	**77.8 ± 1.0**	90.4 ± 0.9	89.8 ± 0.8	**71.9 ± 2.3**	**70.4 ± 2.8**
	MO-NDLP	74.4 ± 1.4	77.4 ± 1.1	91.3 ± 1.0	90.9 ± 1.0	70.6 ± 2.7	68.6 ± 2.7
	MC-NDLP	74.4 ± 1.0	77.4 ± 1.0	**92.1 ± 0.7**	**91.6 ± 0.7**	71.8 ± 2.6	70.0 ± 2.6
	S-NDLP	74.6 ± 1.3	77.5 ± 1.1	91.0 ± 1.0	90.4 ± 1.0	71.8 ± 2.8	70.2 ± 2.9
dMPLP	BASELINE	**86.1 ± 0.5**	**88.0 ± 0.9**	75.7 ± 2.2	76.8 ± 1.6	81.1 ± 3.6	82.2 ± 5.3
	MO-NDLP	83.5 ± 0.6	85.1 ± 0.6	**89.1 ± 1.7**	**89.0 ± 2.1**	**85.8 ± 3.3**	**89.3 ± 2.5**
	MC-NDLP	81.4 ± 1.7	82.0 ± 1.5	83.7 ± 4.2	83.7 ± 3.6	70.0 ± 4.3	71.5 ± 3.9
	S-NDLP	85.6 ± 1.0	86.9 ± 1.0	84.8 ± 2.7	86.3 ± 2.3	83.6 ± 4.7	87.0 ± 4.3

bold we highlight the best training strategy for each metric/model/task combination, while the underlined scores indicate the best training strategy across all models.

For comparison, we evaluated an NDLP-incapable graph autoencoder (GAE) model trained under the baseline strategy. While GAE performs deceptively well on the *General DLP* task, it fails to capture edge directionality, as expected. This is reflected in its random performance on the *Directional* task, with a ROC-AUC of 0.5. This limitation arises from the inner product decoder used by GAE [20], which always assigns the same probability to edges (u, v) and (v, u). Results from this experiment are reported in the first rows of Tables 2, 3 and 4.

Our proposed strategies consistently improved performance on the *Directional* and on the *Bidirectional* tasks across all datasets and models, only slightly compromising (at times even benefiting) *General DLP* performance [42], with a

Table 3. ROC-AUC and AUPRC test scores of various models on CiteSeer Dataset, trained with the *Baseline* strategy and our proposed strategies.

MODEL	STRATEGY	GENERAL		DIRECTIONAL		BIDIRECTIONAL	
		ROC-AUC	AUPRC	ROC-AUC	AUPRC	ROC-AUC	AUPRC
GAE	BASELINE	78.6 ± 0.7	84.1 ± 0.6	50.0 ± 0.0	50.0 ± 0.0	56.2 ± 3.8	59.3 ± 1.9
GR-GAE	BASELINE	77.0 ± 0.7	84.3 ± 0.6	55.7 ± 2.3	58.2 ± 3.2	72.5 ± 3.7	71.3 ± 4.4
	MO-NDLP	78.6 ± 0.8	82.6 ± 1.3	73.4 ± 1.6	76.8 ± 1.2	92.6 ± 2.1	94.6 ± 1.5
	MC-NDLP	**81.9 ± 0.8**	**85.1 ± 0.5**	**75.5 ± 0.7**	**78.9 ± 0.6**	85.9 ± 2.8	85.1 ± 3.1
	S-NDLP	80.8 ± 0.9	84.4 ± 0.7	75.2 ± 1.0	78.2 ± 0.9	**97.5 ± 1.1**	**98.0 ± 0.7**
ST-GAE	BASELINE	80.9 ± 0.8	**85.2 ± 0.7**	56.0 ± 0.3	61.1 ± 0.5	72.0 ± 4.5	73.0 ± 3.7
	MO-NDLP	**81.4 ± 1.5**	82.6 ± 2.0	**78.5 ± 2.3**	**80.1 ± 1.8**	**90.5 ± 4.4**	**92.2 ± 4.1**
	MC-NDLP	77.8 ± 1.8	79.3 ± 2.4	75.5 ± 4.0	79.5 ± 2.8	73.9 ± 5.1	75.1 ± 4.9
	S-NDLP	80.0 ± 1.3	82.0 ± 1.4	72.6 ± 1.6	77.4 ± 1.1	88.9 ± 4.3	90.3 ± 3.9
DiGAE	BASELINE	**78.5 ± 0.9**	**83.5 ± 0.8**	56.6 ± 1.0	65.2 ± 1.5	62.3 ± 3.3	65.8 ± 3.8
	MO-NDLP	72.6 ± 5.0	74.9 ± 5.2	68.7 ± 3.4	71.7 ± 4.1	**65.6 ± 4.5**	70.8 ± 5.2
	MC-NDLP	72.7 ± 1.3	74.6 ± 0.9	**78.3 ± 2.9**	**80.1 ± 1.8**	58.6 ± 3.8	61.6 ± 3.3
	S-NDLP	71.8 ± 3.9	75.4 ± 4.6	63.3 ± 1.4	69.7 ± 2.2	65.5 ± 5.1	**71.5 ± 5.8**
MLP-GAE	BASELINE	73.3 ± 0.8	**76.1 ± 0.7**	88.4 ± 0.7	89.8 ± 0.6	76.5 ± 1.1	76.5 ± 2.6
	MO-NDLP	**74.0 ± 0.9**	75.2 ± 1.0	91.8 ± 0.5	92.2 ± 0.5	**90.2 ± 0.9**	**90.0 ± 1.4**
	MC-NDLP	73.7 ± 0.8	74.3 ± 0.9	**92.6 ± 0.5**	**92.9 ± 0.4**	78.5 ± 1.1	73.6 ± 2.4
	S-NDLP	73.3 ± 0.7	74.8 ± 0.9	89.8 ± 0.3	90.1 ± 0.3	85.5 ± 2.5	85.1 ± 2.4
MAGNET	BASELINE	71.6 ± 0.7	74.9 ± 0.8	89.5 ± 0.6	89.9 ± 0.6	70.9 ± 6.1	68.9 ± 6.6
	MO-NDLP	72.3 ± 0.6	74.7 ± 0.6	91.0 ± 0.6	91.2 ± 0.5	74.6 ± 7.1	73.4 ± 7.7
	MC-NDLP	**73.2 ± 0.9**	**75.2 ± 0.9**	**91.6 ± 0.6**	**91.7 ± 0.6**	71.1 ± 7.5	69.7 ± 7.5
	S-NDLP	71.3 ± 0.8	74.6 ± 0.8	89.6 ± 0.6	90.0 ± 0.5	**75.3 ± 6.2**	**73.5 ± 7.0**
dMPLP	BASELINE	83.9 ± 0.9	86.8 ± 0.9	72.3 ± 1.5	73.8 ± 1.7	84.3 ± 7.0	86.2 ± 6.0
	MO-NDLP	81.2 ± 1.6	83.7 ± 1.7	**86.9 ± 2.1**	**87.6 ± 1.8**	**87.7 ± 4.6**	**90.8 ± 2.9**
	MC-NDLP	77.5 ± 1.9	80.8 ± 1.9	77.8 ± 3.3	77.0 ± 2.7	63.7 ± 8.6	65.5 ± 8.8
	S-NDLP	**84.9 ± 1.7**	**87.2 ± 1.8**	80.7 ± 1.4	82.9 ± 1.0	85.2 ± 7.1	88.8 ± 5.0

few exceptions. For instance, MAGNET showed similar performance on Cora and CiteSeer, regardless of the training strategy, while it achieved significant improvement in the *Bidirectional* task on the Google dataset when trained using our strategies. This highlights that even though some models, like MAGNET, show limited gains on specific datasets, the overall benefits might be more pronounced in larger datasets like Google. dMPLP shows good results with a balanced performance across all tasks for the *S-NDLP* and *MO-NDLP* strategies, while *MC-NDLP* seems less effective especially for the *Bidirectional* task. Reflecting the representational power of the original undirected model, dMPLP achieves the best performances in the *General* task both for Citeseer and Google datasets.

For other models like DiGAE, we observed a trade-off: its performance on the *Directional* task improved, but often at the expense of lower *General* task scores. Notably, on the Google dataset, especially with the *S-NDLP* strategy,

Table 4. ROC-AUC and AUPRC test scores of various models on Google Dataset, trained with the *Baseline* strategy and our proposed strategies.

MODEL	STRATEGY	GENERAL		DIRECTIONAL		BIDIRECTIONAL	
		ROC-AUC	AUPRC	ROC-AUC	AUPRC	ROC-AUC	AUPRC
GAE	BASELINE	93.5 ± 0.2	94.9 ± 0.2	50.0 ± 0.0	50.0 ± 0.0	54.8 ± 0.8	53.6 ± 1.4
GR-GAE	BASELINE	**98.3 ± 0.1**	**98.9 ± 0.0**	76.5 ± 0.8	69.1 ± 0.9	92.0 ± 0.2	91.9 ± 0.2
	MO-NDLP	97.4 ± 0.1	98.1 ± 0.1	91.9 ± 0.2	90.3 ± 0.4	97.6 ± 0.1	97.6 ± 0.1
	MC-NDLP	95.7 ± 0.1	95.7 ± 0.1	92.6 ± 0.2	92.8 ± 0.1	95.1 ± 0.1	94.2 ± 0.1
	S-NDLP	96.9 ± 0.1	97.7 ± 0.0	**94.3 ± 0.1**	**94.7 ± 0.1**	**98.0 ± 0.0**	**98.5 ± 0.0**
ST-GAE	BASELINE	**98.4 ± 0.1**	**98.7 ± 0.0**	87.2 ± 0.2	86.2 ± 0.1	92.2 ± 0.3	89.6 ± 0.4
	MO-NDLP	97.6 ± 0.2	97.4 ± 0.3	**96.6 ± 0.1**	**96.8 ± 0.1**	**98.8 ± 0.1**	**98.6 ± 0.1**
	MC-NDLP	96.6 ± 0.1	96.4 ± 0.2	94.6 ± 0.1	96.1 ± 0.1	96.4 ± 0.1	95.9 ± 0.1
	S-NDLP	97.6 ± 0.0	97.5 ± 0.1	95.0 ± 0.1	96.0 ± 0.1	98.3 ± 0.1	96.8 ± 0.1
DiGAE	BASELINE	**97.0 ± 0.1**	**97.8 ± 0.1**	92.9 ± 0.2	94.5 ± 0.2	90.9 ± 0.3	87.7 ± 0.4
	MO-NDLP	94.7 ± 0.1	95.7 ± 0.2	95.9 ± 0.1	96.9 ± 0.1	**97.7 ± 0.1**	**97.9 ± 0.1**
	MC-NDLP	91.7 ± 0.6	92.4 ± 0.5	96.3 ± 0.2	**97.2 ± 0.1**	95.7 ± 0.3	95.5 ± 0.4
	S-NDLP	96.8 ± 0.1	97.3 ± 0.1	**96.5 ± 0.1**	97.0 ± 0.1	96.7 ± 0.2	96.3 ± 0.3
MLP-GAE	BASELINE	90.8 ± 0.1	91.6 ± 0.0	93.5 ± 0.1	94.4 ± 0.1	81.2 ± 0.2	77.8 ± 0.4
	MO-NDLP	90.4 ± 0.1	91.0 ± 0.1	97.0 ± 0.0	97.3 ± 0.0	95.6 ± 0.1	95.1 ± 0.1
	MC-NDLP	86.3 ± 0.1	87.9 ± 0.1	**98.4 ± 0.1**	**98.5 ± 0.1**	**96.4 ± 0.2**	95.4 ± 0.1
	S-NDLP	**91.2 ± 0.1**	**91.9 ± 0.1**	97.6 ± 0.0	97.8 ± 0.0	96.0 ± 0.1	**95.5 ± 0.1**
MAGNET	BASELINE	**89.1 ± 0.1**	**90.1 ± 0.0**	93.8 ± 0.6	94.3 ± 0.4	83.9 ± 2.0	77.7 ± 2.3
	MO-NDLP	88.5 ± 0.2	89.8 ± 0.1	97.1 ± 0.1	97.2 ± 0.1	**92.9 ± 0.1**	**91.1 ± 0.3**
	MC-NDLP	84.7 ± 0.7	86.2 ± 0.4	**97.6 ± 0.0**	**97.3 ± 0.1**	92.5 ± 0.2	88.9 ± 0.3
	S-NDLP	87.9 ± 0.3	89.4 ± 0.2	96.7 ± 0.1	96.8 ± 0.1	91.9 ± 0.5	88.3 ± 0.9
dMPLP	BASELINE	**98.7 ± 0.1**	**98.9 ± 0.2**	93.6 ± 0.9	92.3 ± 2.2	95.6 ± 0.7	93.4 ± 0.8
	MO-NDLP	87.3 ± 2.8	85.1 ± 2.9	**97.0 ± 0.2**	**97.1 ± 0.3**	**98.1 ± 0.6**	**97.7 ± 0.8**
	MC-NDLP	96.6 ± 0.6	97.6 ± 0.4	93.6 ± 0.4	93.7 ± 0.5	96.1 ± 0.4	96.0 ± 0.5
	S-NDLP	94.9 ± 1.0	95.0 ± 1.0	96.7 ± 0.3	96.8 ± 0.2	96.3 ± 0.8	94.5 ± 1.6

DiGAE maintained its *General* task performance while delivering modest gains in *Directional* and *Bidirectional* tasks. Both MLP-GAE and MAGNET performed well on the *Directional* task but struggled on the *General* task, where their scores were systematically lower than those of the NDLP-incapable baseline GAE. DiGAE also struggled with the *General* task, surpassing GAE's baseline performance only on the Google dataset.

Selecting the right model-strategy combination depends on how much one is willing to sacrifice *General* task performance for improvements in *Directional* and *Bidirectional* tasks. Interestingly, this trade-off is not always necessary. For example, with the CiteSeer dataset, Gravity-GAE with *MC-NDLP* achieved the best *General* task performance while significantly improving *Directional* and *Bidirectional* scores. However, the optimal combination of model and strategy varies by dataset. For the Cora dataset, Gravity-AE with *S-NDLP* offers a balanced solution, delivering strong *Directional* and *Bidirectional* performance with only a slight reduction in *General* task scores. On the CiteSeer dataset, ST-GAE with *MO-NDLP* provides a good balance, offering competitive *General* task performance alongside noticeable gains in *Directional* and *Bidirectional* tasks. Similarly, for the Google dataset, ST-GAE with *MO-NDLP* proves to be an excellent choice, delivering significant improvements in *Directional* and *Bidirectional* tasks with minimal sacrifice in performance on the *General* task.

5 Conclusions

In this paper, we introduced and evaluated new training strategies to improve performance on Neural Directed Link Prediction tasks, addressing the limitations of current models in learning edge directionality. By extending existing models to handle multiple sub-tasks simultaneously, we demonstrated that the proposed strategies – Multi-Class (MC-DLP), Scalarization-based (S-DLP), and Multi-Objective (MO-DLP) Directed Link Prediction – consistently improve performance on both *Directional* and *Bidirectional* tasks, although at times with a trade-off in *General DLP* task performance.

While no single approach universally outperforms across all settings, the flexibility offered by our proposed training strategies provides a powerful means for improving NDLP model capabilities, and adopting any of the strategies is likely to yield meaningful benefits over the models trained without our optimization strategies. The proposed strategies do not require any modifications in the original models' architecture and are thus applicable to most MPNN models and versatile. The three proposed strategies affect the computational complexity of training differently. While *MC-NDLP* has a minimal impact, both *S-NDLP* and *MO-NDLP* require the computation of three losses, increasing the total memory needed. Nevertheless, also these two strategies can be effectively trained by parallizing the computation of the losses and implementing batching.

Future work can focus on refining these strategies to minimize trade-offs, particularly for applications that demand robust handling of directed graphs and directed link prediction. Our training strategies for learning edge directionality might also be usefully combined with approaches that allow GNNs to better represent edge directionality. Many alternative encodings [4, 5] and labeling tricks [1, 2, 3] have been proposed to enhance the expressiveness of GNNs, also for performing DLP, and it would be interesting to explore a wider range of augmented models. Simultaneous training across the three facets of DLP enables more concise comparative studies on the ability of models and various enhancements to provide balanced performance across these facets. Also, an interesting area for future exploration is knowledge graphs (KG), which could greatly benefit from our methods. Since KG-oriented tasks often employ specialized losses with margin terms [4] and involve complex query answering rather than basic link prediction [29], studying how enhanced directionality learning impacts KG performance would be a valuable direction.

Acknowledgements. M. S. acknowledges support from Grants No. RYC2022-037932-I and CNS2023-144156 funded by MCIN/AEI/10.13039/501100011033 and the European Union NextGenerationEU/PRTR.

References

1. Albert, R., Barabási, A.L.: Statistical mechanics of complex networks. Rev. Mod. Phys. **74**(1), 47 (2002)
2. Altman, E., Blanuša, J., Von Niederhäusern, L., Egressy, B., Anghel, A., Atasu, K.: Realistic synthetic financial transactions for anti-money laundering models. In: Advances in Neural Information Processing Systems, vol. 36 (2024)
3. Arrar, D., Kamel, N., Lakhfif, A.: A comprehensive survey of link prediction methods. J. Supercomput. (2023)
4. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in Neural Information Processing Systems, vol. 26 (2013)
5. Cai, L., Ji, S.: A multi-scale approach for graph link prediction. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 04, pp. 3308–3315 (2020)
6. Désidéri, J.A.: Multiple-gradient descent algorithm (MGDA) for multiobjective optimization. C. R. Math. **350**(5), 313–318 (2012)
7. Dong, K., Guo, Z., Chawla, N.: Pure message passing can estimate common neighbor for link prediction. In: Advances in Neural Information Processing Systems, vol. 37, pp. 73000–73035 (2024)
8. Egressy, B., Von Niederhäusern, L., Blanuša, J., Altman, E., Wattenhofer, R., Atasu, K.: Provably powerful graph neural networks for directed multigraphs. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, pp. 11838–11846 (2024)
9. Fu, X., Zhang, J., Meng, Z., King, I.: MAGNN: metapath aggregated graph neural network for heterogeneous graph embedding. In: WWW 2020, pp. 2331–2341. Association for Computing Machinery, New York (2020)
10. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: Proceedings of the 34th International Conference on Machine Learning, ICML 2017, vol. 70, pp. 1263–1272 (2017)
11. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
12. He, C., Zeng, J., Li, Y., Liu, S., Liu, L., Xiao, C.: Two-stream signed directed graph convolutional network for link prediction. Phys. A: Stat. Mech. Appl. **605**, 128036 (2022)
13. He, C., Cheng, J., Fei, X., Weng, Y., Zheng, Y., Tang, Y.: Community preserving adaptive graph convolutional networks for link prediction in attributed networks. Knowl.-Based Syst. **272**, 110589 (2023)
14. Hogan, A., et al.: Knowledge graphs. ACM Comput. Surv. (CSUR) **54**(4), 1–37 (2021)
15. Holme, P., Saramäki, J.: Temporal Network Theory. Springer (2019)
16. Hu, W., et al.: Open graph benchmark: datasets for machine learning on graphs. In: Advances in Neural Information Processing Systems, vol. 33, pp. 22118–22133 (2020)

17. Hu, Y., Xian, R., Wu, Q., Fan, Q., Yin, L., Zhao, H.: Revisiting scalarization in multi-task learning: a theoretical perspective. In: Advances in Neural Information Processing Systems, vol. 36 (2024)
18. Jin, M., et al.: A survey on graph neural networks for time series: forecasting, classification, imputation, and anomaly detection. IEEE Trans. Pattern Anal. Mach. Intell. (2024)
19. Johannessen, F., Jullum, M.: Finding money launderers using heterogeneous graph neural networks. arXiv preprint arXiv:2307.13499 (2023)
20. Kipf, T.N., Welling, M.: Variational graph auto-encoders. arXiv preprint arXiv:1611.07308 (2016)
21. Kollias, G., Kalantzis, V., Id'e, T., Lozano, A.C., Abe, N.: Directed graph auto-encoders. In: AAAI Conference on Artificial Intelligence (2022)
22. Kumar, A., Singh, S.S., Singh, K., Biswas, B.: Link prediction techniques, applications, and performance: a survey. Phy. A: Stat. Mech. Appl. **553**, 124289 (2020)
23. Li, J., et al.: What's behind the mask: understanding masked graph modeling for graph autoencoders. In: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 1268–1279 (2023)
24. Li, X., Chen, H.: Recommendation as link prediction: a graph kernel-based machine learning approach. In: Proceedings of the 9th ACM/IEEE-CS Joint Conference on Digital Libraries, JCDL 2009, pp. 213–216. ACM, New York (2009)
25. Lin, D., Wu, J., Xuan, Q., Tse, C.K.: Ethereum transaction tracking: inferring evolution of transaction networks via link prediction. Phys. A: Stat. Mech. Appl. **600**, 127504 (2022)
26. Lü, L., Zhou, T.: Link prediction in complex networks: a survey. Phys. A: Stat. Mech. Appl. **390**(6), 1150–1170 (2011)
27. Qin, M., Yeung, D.Y.: Temporal link prediction: a unified framework, taxonomy, and review. ACM Comput. Surv. **56**(4) (2023)
28. Reiser, P., et al.: Graph neural networks for materials science and chemistry. Commun. Mater. **3**(1) (2022)
29. Ren, H., Hu, W., Leskovec, J.: Query2box: reasoning over knowledge graphs in vector space using box embeddings. arXiv preprint arXiv:2002.05969 (2020)
30. Rossi, E., Charpentier, B., Giovanni, F.D., Frasca, F., Günnemann, S., Bronstein, M.M.: Edge directionality improves learning on heterophilic graphs. In: Proceedings of the Second Learning on Graphs Conference, pp. 25:1–25:27. PMLR (2024). iSSN: 2640-3498
31. Salha, G., Limnios, S., Hennequin, R., Tran, V.A., Vazirgiannis, M.: Gravity-inspired graph autoencoders for directed link prediction. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, pp. 589–598 (2019)
32. Shervashidze, N., Schweitzer, P., Leeuwen, E.J., Mehlhorn, K., Borgwardt, K.M.: Weisfeiler-Lehman graph kernels. J. Mach. Learn. Res. **12**(77), 2539–2561 (2011)
33. Simonovsky, M., Komodakis, N.: GraphVAE: towards generation of small graphs using variational autoencoders. In: Kůrková, V., Manolopoulos, Y., Hammer, B., Iliadis, L., Maglogiannis, I. (eds.) ICANN 2018. LNCS, vol. 11139, pp. 412–422. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01418-6_41
34. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: International Conference on Learning Representations (2018)
35. Wang, J., Liang, J., Yao, K., Liang, J., Wang, D.: Graph convolutional autoencoders with co-learning of graph structure and node attributes. Pattern Recogn. **121**, 108215 (2022)

36. Weber, M., et al.: Anti-money laundering in bitcoin: experimenting with graph convolutional networks for financial forensics. arXiv preprint arXiv:1908.02591 (2019)
37. Wu, H., Song, C., Ge, Y., Ge, T.: Link prediction on complex networks: an experimental survey. Data Sci. Eng. **7**(3), 253–278 (2022)
38. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: International Conference on Learning Representations (2019)
39. Yi, T., Zhang, S., Bu, Z., Du, J., Fang, C.: Link prediction based on higher-order structure extraction and autoencoder learning in directed networks. Knowl.-Based Syst. **241**, 108241 (2022)
40. Zhang, M., Chen, Y.: Link prediction based on graph neural networks. In: Advances in Neural Information Processing Systems, vol. 31 (2018)
41. Zhang, S., Zhang, W., Bu, Z., Zhang, X.: ClusterLP: a novel cluster-aware link prediction model in undirected and directed graphs. Int. J. Approximate Reasoning **172**, 109216 (2024)
42. Zhang, X., He, Y., Brugnone, N., Perlmutter, M., Hirn, M.: MagNet: a neural network for directed graphs. In: Advances in Neural Information Processing Systems (2021)
43. Zhang, Y., Tan, Y., Jian, S., Wu, Q., Li, K.: DGLP: incorporating orientation information for enhanced link prediction in directed graphs. In: ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6565–6569. IEEE (2024)
44. Zhu, S., Li, J., Peng, H., Wang, S., He, L.: Adversarial directed graph embedding. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 5, pp. 4741–4748 (2021)