Relation-Dependent Sampling for Multi-Relational Link Prediction

Arthur Feeney [*1] **Rishabh Gupta** [*1] **Veronika Thost** [2] **Rico Angell** [1] **Gayathri Chandu** [1] **Yash Adhikari** [1] **Tengfei Ma** [2]

1. Introduction

Multi-relational graphs specifically allow for representing different types of relations and are an effective way to model various types of data including social networks (Balakrishnan & T. V., 2020), knowledge graphs (Vrandečić & Krötzsch, 2014), and biomedical networks, such as interactions between molecules (Krogan et al., 2006). Link prediction (Zhang & Chen, 2018) in these graphs is an actively studied problem that is critical for many applications. For example, drug-drug interaction (DDI) prediction is a task important for drug development and repurposing: on the one hand, some diseases are best treated by combinations of drugs (e.g., antiviral drugs are typically administered as cocktails), on the other hand, one has to know about critical side effects between new molecules and existing drugs (in a stricter sense, side effects are consequences of DDIs).

Graph neural networks gained great success in such applications recently. However, learning over the entire graphs is often computationally expensive or even impossible due to their sheer size. Hence, learning approaches have to apply batching and/or sampling (i.e., selecting a subset of nodes and/or relations from the original graph) for being able to process these graphs (Zitnik et al., 2018; Huang et al., 2019).

However, sampling approaches for GNNs have been studied only in a general setting, that is, without focusing on the multi-relational nature of many graphs. Existing works propose approaches to sample random neighborhoods for individual nodes (Hamilton et al., 2017), input for entire layers of the graph neural networks (Chen et al., 2018; Huang et al., 2018), or subgraphs as batches (Zeng et al., 2020). While some of these techniques are adaptive in that they learn to sample based on the problem at hand (vs. randomly) (Huang et al., 2018; Zeng et al., 2020; Xu et al., 2020), relation types are not regarded specifically. This may lead to unintended effects in the distribution of relations in the samples.

In this paper, we study how to do sampling in multi-relational graphs. Specifically, we consider a straightforward extension of the popular relational graph convolutional network (R-GCN) (Schlichtkrull et al., 2018) architecture and propose relation-dependent sampling. A REINFORCE-based (Williams, 1992) approach is used to automatically learn the sampling probability of each edge. To show the effectiveness of our model, in experiments we focus on a typical multi-relational link prediction task, drug-drug-interaction prediction. The results show that our model can learn the right balance: relation-type probabilities that reflect both frequency and importance, and also offer some kind of explanation. In addition, our models outperform state-of-the-art approaches in these prediction tasks.

Lastly, we want to point out a risk of applying standard AI techniques in the biomedical setting, which relates to a special type of graph edges: negative edges. In drug-drug interaction prediction problems (as in general link prediction), it is common to use random negative sampling to increase link prediction performance (Zitnik et al., 2018; Huang et al., 2019; Ma et al., 2019). However, the absence of interaction information in existing datasets does usually not mean that there are no such interactions altogether. To partially overcome this lack of verified negative samples, we suggest to apply information that is readily available but, to the best of our knowledge, has not been considered in this context so far: there are drug mixture products, approved by FDA or similar organizations, whose molecular compounds are hence unlikely to interact in a critical way; moreover, data about drugs that synergize if taken in combination (e.g., by improved efficacy and reduced side effects) provides valuable expert knowledge which can be integrated easily.

Our contributions include: (1) We propose a new approach to *relation-specific sampling* for graph neural networks which yields *state-of-the-art performance* in DDI prediction. (2) We show that relation-specific sampling specifically *benefits imbalanced data*, and show that the learned relation probabilities *support explainability*. (3) We point out two kinds of expert knowledge which can serve as *negative samples for DDI prediction* and experimentally show that our two suggested ways to integrate it improve performance.

[*]Equal contribution [1]University of Massachusetts, Amherst, USA [2]MIT-IBM Watson AI Lab, IBM Research, USA. Correspondence to: Tengfei Ma <tengfei.ma1@ibm.com>, Veronika Thost <veronika.thost@ibm.com>.

Published at the Graph Representation Learning and Beyond (GRL+) Workhop at ICML 2020.

2. Modeling Edge-Type Probabilities

Multi-relational graph neural networks are successful in various tasks, but the computational complexity of the models is too high for large graphs. Motivated by solutions in homogeneous graphs, we propose to utilize sampling techniques to enhance model scalability. We consider to incorporate probabilities for each edge type into the reasoning process over multi-relational graphs.

2.1. Extending the Message Passing

We base our model on R-GCN (Schlichtkrull et al., 2018), one of the most widely used GNN models for multi-relational graphs. A very straightforward idea to incorporate edge-type probabilities in R-GCNs is to assign the probability as an importance weight to each edge type based on the local neighborhood. That is, we consider an additional learnable parameter l_r for each edge type $r \in \mathcal{R}$. During node update, we then calculate a softmax distribution based on the local neighborhood and the new parameters as follows:

$$c_{u,r} = \frac{e^{l_r}}{\sum_{r' \in \mathcal{R}} |\mathcal{N}_{u,r'}| e^{l_{r'}}}$$

Thus we update the node representation as in (Schlichtkrull et al., 2018):

$$h_u^{l+1} = \phi\left(h_u^{(l)} + \sum_{r \in \mathcal{R}} \sum_{v \in N_{u,r}} c_{u,r} W_r^{(l)} h_v^{(l)}\right)$$

Probabilities integrated in this way during the message passing stage offer a certain level of explainability, but no solution in terms of scalability. Therefore we consider incorporating them during the neighbor sampling stage.

2.2. Sampling based on Learned Probabilities

Learning edge probabilities has been explored in different contexts. (Huang et al., 2018) brought in a neural network to learn the optimum probability for importance sampling. (Franceschi et al., 2019) incorporated a Bernoullie distribution for latent graph structures and automatically learned the hyperparameters of the distribution. Following similar ideas, we propose a sampling method based on learned probabilities extending GraphSAGE (Hamilton et al., 2017). The main difference to the latter is that our method *learns* sampling probabilities for each relation type during training. The idea is to learn how to retain a useful neighborhood around an edge without compromising scalability.

More specifically, our model has latent parameters $l \in \mathbb{R}^{|\mathcal{R}|}$ initialized with elements $l_i \sim \mathcal{N}(0, 1)$. For a given target edge, we sample a fixed-sized k-hop neighborhood around that edge iteratively (i.e., for $k = 1, 2, \dots$) to obtain a subgraph. In each hop k, we use the edge-type parameters l to generate a softmax distribution over all the edges in that hop and sample $n_k = s_k \cdot |\mathcal{N}_{k-1}|$ edges from it with replacement; s_k is the number of edges to sample in hop k for each node in hop $k-1$, $\mathcal{N}_{k-1}$ is the set of nodes sampled in hop $k-1$. The sampling probability for an edge type r in the k-th hop can be calculated as follows:

$$p_{r,k} = \frac{e^{l_r}}{\sum_{r' \in \mathcal{R}} |\mathcal{E}_{k,r'}| e^{l_{r'}}}$$

where $\mathcal{E}_{k,r'}$ is the set of edges of relation type r' in hop k. As there are difficulties in backpropagating through the sampled subgraph, we use the score function estimator REINFORCE (Williams, 1992). For a loss function L and sampled subgraph g, to estimate the gradient of the loss with respect to the latent parameters l, we compute $p_l(g)$, the probability of sampling g given l:

$$\nabla_l \mathbb{E}_g[L(g)] = \mathbb{E}_g[L(g) \nabla_l \log p_l(g)]$$

In this way, REINFORCE does not require backpropagation through R-GCN layers or the sampled subgraph if we can compute $\nabla_l \log p_l(g)$. We observe that this can be achieved easily by computing $\log p_l(g)$ as below and backpropagating the gradient; r_i is the edge type of the i^{th} edge:

$$\begin{aligned}\log p_l(g) &= \sum_k \sum_{i=1}^{n_k} \log p_{r_i,k} \\ &= \sum_k \sum_{i=1}^{n_k} \log \frac{e^{l_{r_i}}}{\sum_{r' \in \mathcal{R}} |\mathcal{E}_{k,r'}| e^{l_{r'}}}\end{aligned}$$

3. Additional Evidence for DDI Prediction: Drug Mixtures and Synergies

Past research has shown that random negative sampling and including various types of data (e.g., drug-protein interactions) help to improve DDI prediction (Zitnik et al., 2018). We propose to apply compound pairs contained in marketed drug mixture products, available from DRUGBANK (DS et al., 2017), and knowledge about drugs that synergize, available from DRUGCOMBDB (Liu et al., 2019). For side effect prediction, this data can be considered as verified negative evidence (vs. random, possibly incorrect negative samples). For general DDI prediction, it can provide signals towards positive DDI types. To the best of our knowledge, this has not been studied before. For simplicity, we will call all this data synergistic evidence in the following (i.e., although the compounds from mixtures do not necessarily yield synergistic effects, which means effects that are higher than if the drugs are taken individually).

To incorporate that evidence into our model, we propose two approaches. 1. For side effect prediction, we can use

Table 1: Overview of datasets

	# Nodes	# Edge Types	# Edges
DRUGBANK	1,861	86	192,284
TWOSIDES100	1,918	100	30,979

the evidence as negative samples during the computation of the loss function (i.e., during training and validation). While we thus include a more reliable set of negative samples, the extra information is not incorporated into the model yet. 2. Therefore, we also consider the synergistic evidence as a new edge type in the graph data to allow GNNs to use the extra information when computing node embeddings.

4. Evaluation

We conduct experiments to evaluate the following questions:

Q1 Does the incorporation of edge-type probabilities improve predictions?

Q2 Do learned probabilities offer improvement beyond fixed probabilities such as inverse frequencies?

Q3 Are our two proposals for incorporating drug synergy data effective?

4.1. Datasets and Baselines

We use the DDI prediction dataset **DRUGBANK**[1] (Ryu et al., 2018) and **TWOSIDES100**, a subset of the side effect prediction data from (Tatonetti et al., 2012), see Table 1.

For the synergistic evidence experiments, we extract the **mixture products from DRUGBANK** (DS et al., 2017) and create 18,968 pairs in total from their compounds. Of these pairs, 6,689 drug pairs contain a drug which also occurs in our DRUGBANK dataset. Furthermore, we use the **synergism data from DRUGCOMBDB** (Liu et al., 2019), in total 1,398 drug pairs. Note that in our experiments on the DRUGBANK dataset, we use only the data from DRUGBANK because the DRUGCOMBDB data has poor overlap with that dataset. For the TWOSIDES100 experiments we use both.

We apply common link and DDI prediction baselines: **DistMult** (Yang et al., 2015), a common tensor factorization baseline for link prediction in knowledge graphs. **Multilayer perceptron (MLP)**, a standard feedforward neural network as suggested in (Ryu et al., 2018) for the DRUGBANK dataset. **Message-passing neural network (MPNN)** (Gilmer et al., 2017), a basic GNN architecture that uses message passing to update the node embeddings. **R-GAE** (Schlichtkrull et al., 2018), a relational graph autoencoder that has shown to perform well for DDI prediction (Zitnik et al., 2018). The encoder is an R-GCN which uses edge types for learning node embeddings, and the decoder is a tensor factorization model. We additionally use uniform neighbor sampling similar to (Hamilton et al., 2017) to sample a subgraph around the target edge for message passing. **IFR-GAE** is a baseline in our ablation study. It differs from R-GAE in that it applies a fixed sampling probability for each edge type based on the inverse frequency in the training data. Details about datasets, model configurations, and training can be found in the supplementary material.

[1]Not to be confused with the actual DRUGBANK; note that this name is also used in (Huang et al., 2019).

Table 2: Edge-type prediction results on DRUGBANK and TWOSIDES100

Model	DRUGBANK AUPRC	DRUGBANK AUROC	TWOSIDES100 AUPRC	TWOSIDES100 AUROC
DistMult	61.3	96.5	18.5	50.8
MLP	75.0	98.7	28.6	60.7
MPNN	79.1	99.0	36.5	71.0
R-GAE	80.6	98.9	35.3	67.1
IFR-GAE	81.5	98.9	37.3	71.3
RM-GCN	82.8	99.2	35.7	68.0
RS-GCN	**85.6**	**99.3**	**39.8**	**73.3**
RS-GCN|SyEx	–	–	**40.0**	**73.7**
RS-GCN|SyEd	**87.1**	99.0	39.0	**73.3**

We implemented our proposed approaches on top of R-GAE: we add edge-type probabilities in the message-passing stage in the R-GCN encoder (**RM-GCN**), and similar for the probabilistic sampling based on REINFORCE (**RS-GCN**).

4.2. Results and Discussion

We report our results in Table 2. We observe that GNN-based baselines outperform both MLP and the tensor factorization model. This means that the information about the local neighborhood the GNNs encode is indeed useful. The generally lower numbers of TWOSIDES100 can be explained by the fact that it is less dense that DRUGBANK and it also has a much larger percentage of parallel edges.

A1 OUR EDGE-TYPE PROBABILITIES IMPROVE PREDICTION ACCURACY AND EFFICIENCY

From Table 2, we observe that RM-GCN and RS-GCN perform overall better than all baselines on both datasets. This shows that giving different importance to different edge types based on the local neighborhood during message passing or neighbor sampling is useful in learning node representations. RS-GCN has an added advantage over RM-GCN since it learns to ignore the less informative neighborhood during the sampling stage only and does not need to take less important edges during message passing.

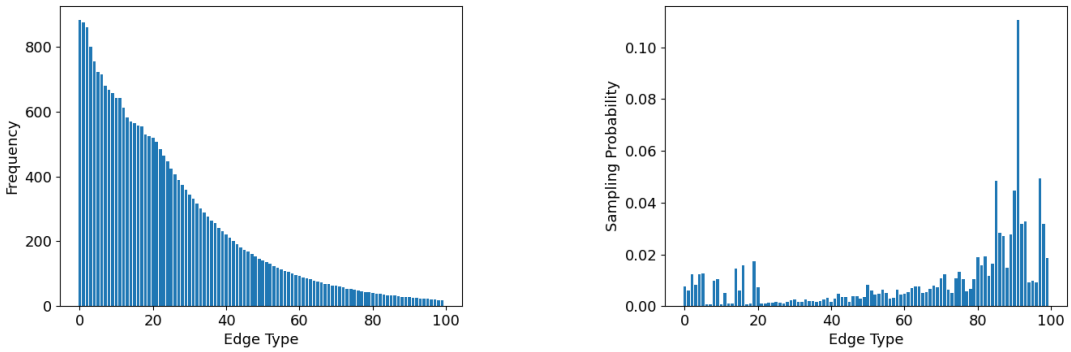

Figure 1: Distribution of edge types vs. learned probabilities for an example neighborhood for TWOSIDES100.

RS-GCN specifically provides an advantage in scalability, this is confirmed by the runtimes: Compared to standard R-GAE (i.e., a version without sampling), we obtain an improvement during both training (**2.47x**) and inference at test time (**1.92x**); recall that RS-GCN applies sampling during training and inference. Accuracy is not compromised.

A2 Learned Probabilities vs. Fixed Ones

To show that it is indeed beneficial to *learn* edge-type probabilities, we compare RS-GCN to IFR-GAE. As it is shown in Table 2, on both datasets, IFR-GAE outperforms the baselines. This is likely due to the fact that both datasets are indeed imbalanced regarding the edge types and shows that edge-type specific sampling is effective for imbalanced data.

Nevertheless, IFR-GAE does not achieve the performance of our models. One of the reasons behind this is that, even though IFR-GAE takes class imbalance into account during sampling, the sampling probabilities are fixed globally and thus irrespective of the local neighborhoods. The sampling probabilities assigned to different edge types in RS-GCN are calculated adaptively based on the distribution of edge types in the local neighborhood of the target edge. Consider Figure 2, which compares the edge type distribution to the assigned probabilities in the 1-hop neighborhood of a specific edge, whose edge type is to be predicted. For demonstration purposes, we chose a neighborhood where all edge types occur exactly once. We observe that for that particular neighborhood, our model learns probabilities that are different from the ones of IFR-GAE. Apart from capturing and resolving the imbalance of edge types, our model also learns to assign high weights to some more commonly occurring edge types based on their importance for the current task. Hence, specific learned probabilities generally reflect importance of edge types better. For DRUGBANK, the probabilities are similar to IFR-GAE, though, which means that the edge types are probably of similar importance for this specific example task. However, since the performance of RS-GCN on DRUGBANK is better than that of IFR-GAE, the learned probabilities must capture some edge-type-specific information beyond frequency. Note that IFR-GAE shows better performance than RM-GCN on TWOSIDES100. Together with RS-GCN outperforming RM-GCN, this confirms that dealing with edge-type probabilities in the context of (sampled) neighborhoods during sampling is generally better than in the model during message passing, where the context is restricted to the neighborhood of the updated node, which was sampled randomly. The globally learned probabilities (i.e., averaged over all sampled neighborhoods) differ from both the inverse frequencies and the probabilities for our example neighborhood, details can be found in the supplementary. Overall, we observed that, for DRUGBANK, class imbalance has most influence on the learned probabilities. The most frequent DDI is assigned the lowest probability, and the least frequent is assigned the highest probability. On the other hand, for the less imbalanced TWOSIDES100, the side effect "traumatic haemorrhage" has almost the same frequency as "stress incontinence" but the former is assigned much higher probability.

A3 Drug Synergy Data Improves DDI Prediction Accuracy Both in the Graph and as Samples

We finally evaluate our suggestion for considering verified data about drug synergies to overcome the lack of verified negative examples. Recall that this is only valid for the TWOSIDES100 data since DRUGBANK contains positive DDIs as well. The second-to-last line in Table 2 (|**SyEx**) confirms that our approach of replacing some of the random negative samples (as used in all other experiments) by synergy data is effective: 1. it decreases the risk attached to using random possibly incorrect negative samples; 2. it yields the same performance as the latter; more specifically, there is even a slight improvement in performance. The last line in the table further shows that synergy data incorporated into the graph as additional edges (|**SyEd**) may be beneficial as well, though the difference is not that pronounced.

References

Balakrishnan, M. and T. V., G. A neural network framework for predicting dynamic variations in heterogeneous social networks. *PLOS ONE*, 15(4):1–21, 04 2020. doi: 10.1371/journal.pone.0231842. URL https://doi.org/10.1371/journal.pone.0231842.

Chen, J., Ma, T., and Xiao, C. FastGCN: Fast learning with graph convolutional networks via importance sampling. In *International Conference on Learning Representations*, 2018.

DS, W., YD, F., AC, G., EJ, L., A, M., JR, G., T, S., Johnson D, L. C., Z, S., N, A., I, I., Y, L., A, M., N, G., A, W., L, C., R, C., D, L., A, P., C, K., and M, W. Drugbank 5.0: a major update to the drugbank database for 2018. 2017.

Franceschi, L., Niepert, M., Pontil, M., and He, X. Learning discrete structures for graph neural networks. In *International Conference on Machine Learning*, 2019.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pp. 1263–1272, 2017.

Hamilton, W., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. In *Conference on Neural Information Processing Systems*, 2017.

Huang, K., Xiao, C., Hoang, T. N., Glass, L. M., and Sun, J. CASTER: predicting drug interactions with chemical substructure representation. *CoRR*, abs/1911.06446, 2019. URL http://arxiv.org/abs/1911.06446.

Huang, W., Zhang, T., Rong, Y., and Huang, J. Adaptive sampling towards fast graph representation learning. In *International Conference on Neural Information Processing Systems*, pp. 4563–4572, 2018.

Krogan, N., Cagney, G., Yu, H., Zhong, G., Guo, X., Ignatchenko, A., Li, J., Pu, S., Datta, N., Tikuisis, A., Punna, T., Peregrín-Alvarez, J., Shales, M., Zhang, X., Davey, M., Robinson, M., Paccanaro, A., Bray, J., Sheung, A., Beattie, B., Richards, D., Canadien, V., Lalev, A., Mena, F., Wong, P., Starostine, A., Canete, M., Vlasblom, J., Wu, S., Orsi, C., Collins, S., Chandran, S., Haw, R., Rilstone, J., Gandi, K., Thompson, N., Musso, G., St Onge, P., Ghanny, S., Lam, M., Butland, G., Altaf-Ul, A., Kanaya, S., Shilatifard, A., O'Shea, E., Weissman, J., Ingles, C., Hughes, T., Parkinson, J., Gerstein, M., Wodak, S., Emili, A., and Greenblatt, J. Global landscape of protein complexes in the yeast saccharomyces cerevisiae. *Nature*, 440(7084):637–643, March 2006. ISSN 0028-0836. doi: 10.1038/nature04670.

Liu, H., Zhang, W., Zou, B., Wang, J., Deng, Y., and Deng, L. DrugCombDB: a comprehensive database of drug combinations toward the discovery of combinatorial therapy. *Nucleic Acids Research*, 48(D1):D871–D881, 10 2019. ISSN 0305-1048.

Ma, T., Shang, J., Xiao, C., and Sun, J. GENN: predicting correlated drug-drug interactions with graph energy neural networks. *CoRR*, abs/1910.02107, 2019. URL http://arxiv.org/abs/1910.02107.

Ryu, J. Y., Kim, H. U., and Lee, S. Y. Deep learning improves prediction of drug–drug and drug–food interactions. *Proceedings of the National Academy of Sciences*, 115(18):E4304–E4311, 2018.

Schlichtkrull, M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I., and Welling, M. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pp. 593–607. Springer, 2018.

Tatonetti, N. P., Ye, P., Daneshjou, R., and Altman, R. B. Data-driven prediction of drug effects and interactions. *Science translational medicine*, 4 125:125ra31, 2012.

Vrandečić, D. and Krötzsch, M. Wikidata: A free collaborative knowledgebase. *Communications of the ACM*, 57: 78–85, 09 2014. doi: 10.1145/2629489.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3–4):229–256, May 1992. ISSN 0885-6125.

Xu, X., Feng, W., Jiang, Y., Xie, X., Sun, Z., and Deng, Z.-H. Dynamically pruned message passing networks for large-scale knowledge graph reasoning. *International Conference on Learning Representations*, 2020.

Yang, B., Yih, W., He, X., Gao, J., and Deng, L. Embedding entities and relations for learning and inference in knowledge bases. *International Conference on Learning Representations*, 2015.

Zeng, H., Zhou, H., Srivastava, A., Kannan, R., and Prasanna, V. Graphsaint: Graph sampling based inductive learning method. *International Conference on Learning Representations*, 2020.

Zhang, M. and Chen, Y. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*, pp. 5165–5175, 2018.

Zitnik, M., Agrawal, M., and Leskovec, J. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466, 2018.

A. Negative Evidence

We extract mixture products from DRUGBANK (DS et al., 2017) and synergism data from DRUGCOMBDB (Liu et al., 2019). When adding negative evidence to a dataset, we only include the negative edges that share an endpoint node with the target dataset. In the experiments on DRUGBANK DDI dataset, we only use the negative evidence from DRUGBANK because the synergism data implies that drug interaction is safe and not that there is no interaction. For TWOSIDES100, therefore, we use both the DRUGBANK and synergism datasets as negative evidence. In order to use the synergism data, we had to match drugs by name, rather than ID. This may result in missing some possible data points.

TWOSIDES100 To create TWOSIDES100, we take a subset of medium-occuring 100 side effects from the original dataset (Tatonetti et al., 2012). We first sort the side effects by their frequency. We start from the 4,000th least common side effect and then consider every 60th side effect up to the 10,000th least common side effect.

B. Model Architectures

When experimenting on DRUGBANK dataset, we always use a hidden dimension of 128. For TWOSIDES100, we use a hidden dimension of 200. The specific architectures were the same for both datasets. For the MLP, we use three layers and apply batch normalization after each layer. For each of the GNN models, we use two graph layers followed by a classifier. For the MPNN, we use an MLP classifier and for the relational models, we use a tensor factorization method called Dedicom, similar to (Zitnik et al., 2018). Additionally for relational graph models, we use 30 bases for basis-decomposition (Schlichtkrull et al., 2018). We use a binary cross entropy loss function for all models. We use ReLU activation function after each layer. Finally, we apply the sigmoid function to each model's output.

C. Training Details

We use a 60/20/20 train/validation/test split for both DRUGBANK and TWOSIDES100. For DRUGBANK, we use the initial node features from (Ryu et al., 2018), where the structural similarity profile (SSP) of each drug is reduced in its dimension (to 50) using PCA and then taken as node embedding. For TWOSIDES100, we generate input features using one-hot encodings. We additionally perform batching on target edges: we create batches of edges in the input graph and sample a subgraph surrounding each of the target edges. We then use the sampled subgraph to make predictions for DDI types of the target edges. The batch size for DRUGBANK is 2000 and is 100 for TWOSIDES100. We train each model for up to 300 epochs and use a patience of 100 epochs: we stop training when the PR-AUC has not increased for 100 epochs. All experiments use the ADAM optimizer with no weight decay and betas of 0.99 and 0.999. We use a learning rate of 0.001 in each experiment.

	non-sampling R-GAE	RS-GCN
Train (per epoch)	245.61	99.05
Test	21.98	11.44

Table 3: Runtime Comparison (in s)

Experiments were run on a single Xeon Gold 6240 CPU with 72 hyper-threaded cores and use versions 1.5.0 of both PyTorch and PyTorch Geometric.

D. Sampling

Our sampling method is based on GraphSage and extracts a subgraph from the k-hop neighborhood surrounding target edges (Hamilton et al., 2017). As our experiments are restricted to networks with two GNN layers, we only sample from the two-hop neighborhood of the target edges. For each method of sampling, we set an upper limit on the number of edges sampled in each hop. The upper limit in the first hop was set to be seven times the batch size. The limit for the second hop was three time the batch size.

We compute logits for each edge based on its attributes. We then apply a softmax function to each neighborhood to get probabilities for sampling the edges. For random sampling, these logits are set to zero; each edge in a neighborhood has the same probability of being sampled. For the inverse frequency sampling, we set the logit to be the log of the inverse of how many times the edge occurs. For the proposed learned sampling method, we have parameters for each edge type that are summed depending on the input edge's types. Each sampling method uses the same underlying implementation. The only difference between them is how we compute the probabilities.

E. Additional Results

Table 3 listed the detailed runtime of our method (RS-GCN) and its non-sampling counterpart, R-GAE. From the table, we can clearly see the speedup of both training and testing from RS-GCN, which demonstrates the efficiency of our sampling method.

In order to show the insights of the learned edge probabilities, we also include additional visualization results of edge frequencies and learned sampling probabilities in Figure 2. As we explained in Section 4.2, different datasets have different distributions for edge probabilities. On DRUGBANK, the learned edge probability is closely proportional to the inverse of the edge frequency; while in TWOSIDES100, the learned edge probability distribution is more complicated.

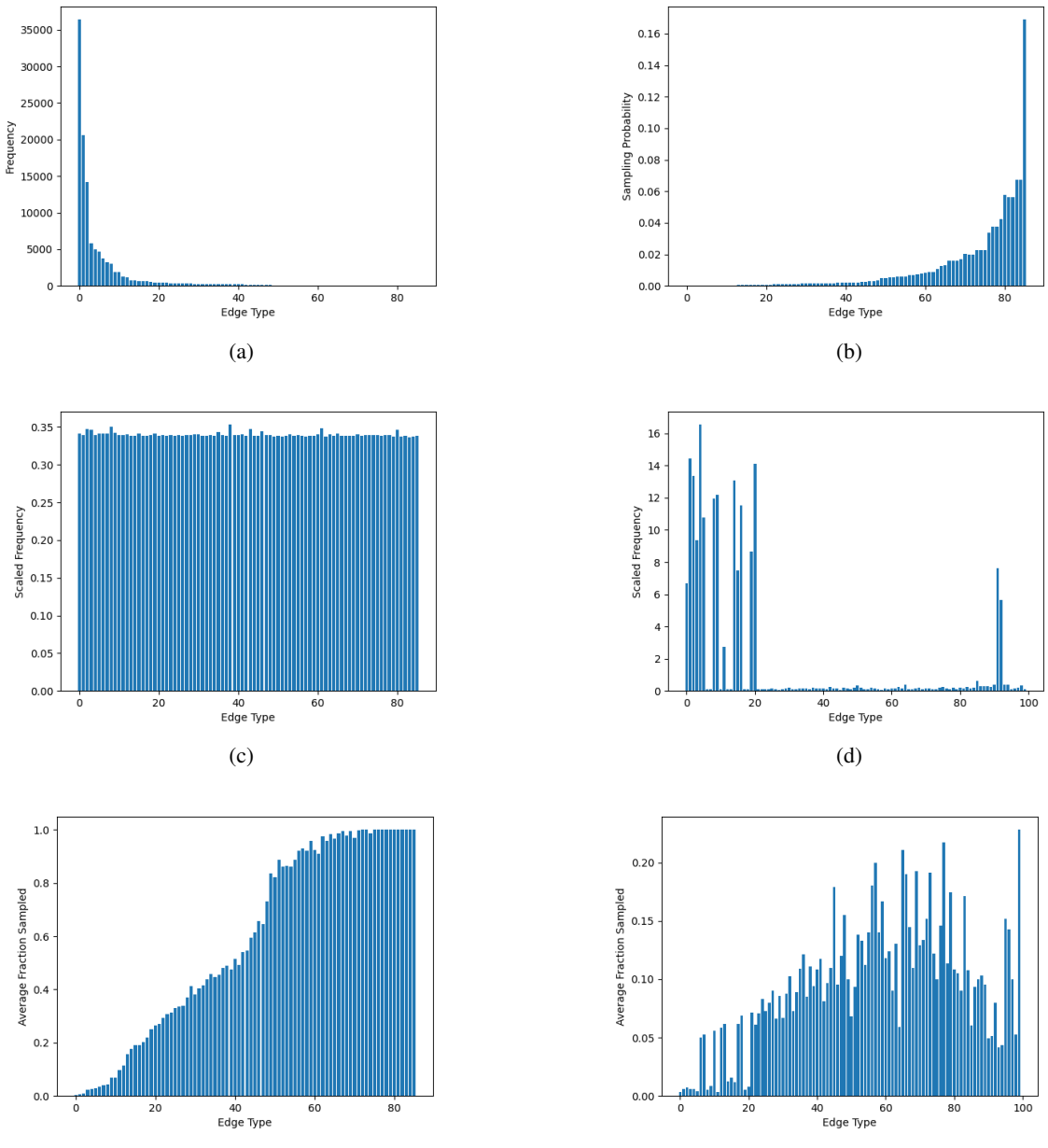

Figure 2: Top: For DRUGBANK, (a) frequency distribution of edge types and (b) probabilities learned for an example neighborhood. Middle: Frequency scaled by the learned probabilities (c) for DRUGBANK and (d) for TWOSIDES100. Bottom: Average fraction of each edge type sampled in the final evaluation (e) for DRUGBANK and (f) TWOSIDES100; the order of edge types (x-axis) in all pictures for a dataset is the same, according to frequency in training data.