Flaky Performances when Pre-Training on Relational Databases with a Plan for Future Characterization Efforts

Anonymous Authors[1]

Abstract

We explore the downstream task performances for graph neural network (GNN) self-supervised learning (SSL) methods trained on subgraphs extracted from relational databases (RDBs). Intuitively, this joint use of SSL and GNNs allows us to leverage more of the available data, which could translate to better results. However, while we observe positive transfer in some cases, others showed systematic performance degradation, including some spectacular ones. We hypothesize a mechanism that could explain this behaviour and draft the plan for future work testing it by characterize how much relevant information different strategies can (theoretically and/or empirically) extract from (synthetic and/or real) RDBs.

1. Introduction

The success story of large language models (Devlin et al., 2018; Brown et al., 2020) hinges on self-supervised learning (SSL): however small is your task-specific dataset, your model may now be pre-trained on a large chunk of humankind's text records, with startling transfer performances on downstream tasks. Deep neural networks (DNNs) in other domains have similarly benefited from different SSL techniques, including some based on image augmentations (Chen et al., 2020; Caron et al., 2020; Chen & He, 2021).

Yet tasks often thought as "easier"—like predicting entries from relational database (RDB) tables—are still addressed by practitioners with fully-supervised, non-deep machine learning (ML) models. Typically, one first "flattens" the RDB to a single table: this lossy pre-processing step enables the use of ML models accepting "tabular data" (Borisov et al., 2021), a domain that has recently been called "the last *unconquered castle* for deep learning" (Kadra et al., 2021).

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

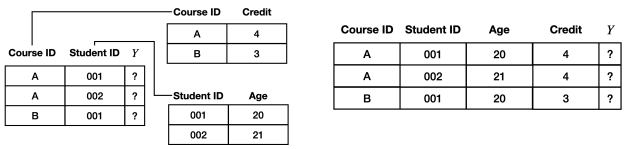
Preliminary work. Under review by the First Workshop of Pre-training: Perspectives, Pitfalls, and Paths Forward at ICML 2022. Do not distribute.

An attack angle toward this conquest is that DNNs need not restrict themselves to tabular inputs: they may leverage more of the original RDB's structure, an hypothesis supported graph neural networks (GNNs) work (Cvitkovic, 2019; 2020). However, publications on deep graph-based models for RDB data remain very rare, a situation perhaps partly due to the lack of appropriate tooling (Vasiloglou et al., 2021), but also to an embarrassing fact: even with access to extra structural information, systematically beating gradient boosted decision trees (GBDT) on RDBs flattened with deep feature synthesis (DFS) (Kanter & Veeramachaneni, 2015) remains a challenge (Cvitkovic, 2019; 2020).

SSL presents another conquest opportunity for DNNs (Arik & Pfister, 2021; Huang et al., 2020; Yoon et al., 2020; Somepalli et al., 2021; Bahri et al., 2022): business ML tasks often have access to a vast number of unlabeled RDB entries in addition to the few labeled ones. Leveraging such untapped data should only do better, right? Wrong. In our experiments, pre-training on unlabeled data (followed by fine tuning) often performed *worse* than a fully-supervised version of the same model, with few cases doing slightly better. In fact, some seemingly "reasonable" choices of SSL strategy gave *systematically worse linear probing performances than a randomly initialized (untrained) instance of the same model*. The use of InfoNode, introduced in Section 3, has shown limited improvements in some cases.

Taking a step back, we consider what kind of underlying mechanisms could explain these observations. Simply put, we hypothesize that SSL may have no way to discriminate "useless" information (noise) from the one "useful" to the downstream task (signal). We propose a framework allowing future work to further investigate the question.

Contributions. To the best of our knowledge, we are the first to approach classification tasks on RDB using GNNs while leveraging unlabeled data using SSL pre-training. We empirically show that, while the use of SSL may confer some advantages for some datasets and labeled/unlabeled ratio, SSL can actually lead to severe performance decrease, *i.e.*, negative transfer. We introduce InfoNode, which helps to a limited extent. We propose an hypothesis as to what phenomenon may cause the observed negative transfer, and elaborate a plan for future characterization work.



(a) RDB. (b) Tabular data.

Figure 1: (a) A relational database (RDB) with three tables. (b) The same information presented as tabular data (single table). Not all RDBs may be so flattened without loss of information. In both cases, the task is to predict the value of the target column Y.

2. Background

Tabular data and relational databases. For our present purpose, we define a *Relational Database* (RDB) as a collection of tables whose columns may reference one-another (or even themselves), while we reserve the term *tabular data* to the case where there is a single table with no reference whatsoever. Figure 1 gives examples of both these formats.

Following Cvitkovic (2020), we focus on the broad class of problems where the goal is to predict values from a specified *target column* of a specified *target table* from a specified RDB, given all available relevant information in the RDB.

However, it is common in applications that some (or even most) of the target column's entries are unavailable at training time (*e.g.*, requiring human labelling and/or relating to future events), which precludes the use of the corresponding rows in a supervised learning paradigm. In this work, we extend the supervised learning problem statement (Cvitkovic, 2020), allowing algorithms to leverage such unlabeled data.

Handling RDBs in a Machine Learning Pipeline. RDBs (multiple tables) are commonly treated as tabular data (single table) in ML pipelines. Of course, we may seek to salvage some information from the other tables by adding columns to the target table through procedures such as join operations (*e.g.*, taking Fig. 1(a) to Fig. 1(b)). Methods like DFS (Kanter & Veeramachaneni, 2015) have been proposed to automate such feature engineering. Regardless of how the RDB was converted to a single table, it may be fed to any machine learning algorithm for tabular data, including tree-based (Ho, 1995; Chen & Guestrin, 2016) and deep learning (Arik & Pfister, 2021; Yoon et al., 2020; Huang et al., 2020; Somepalli et al., 2021; Bahri et al., 2022) ones.

Another approach represents a RDB as an *RDB graph*—a graph with labeled directed edges, node types and node features that has the same informational content as the RDB. Indeed, each row of a table can be represented as a node whose type identifies the corresponding table, the row's non-reference columns are stored as node features, and the row's reference columns specify the destination of a directed edge bearing a label identifying the corresponding column. The sampling strategy RDBToGraph (Cvitkovic, 2020) has been proposed as a pre-processing step to generate one subgraph of the RDB graph for each row of the target table.

Concretely, such an RDB subgraph can be represented as the model input $\boldsymbol{x} = (A, E)$. The node feature information A is such that A_i provides the values A_{ik} for the k-th non-reference column of the row associated with node i (excluding the target column for the *target node*, *i.e.*, the only node of this subgraph associated to the target table), to which we append the name of the associated table (*i.e.*, the node's type). The edge information E specifies the adjacency information: column r of the row associated with node i has a reference to the row associated with node j if and only if $(i, r, j) \in E$ (*i.e.*, there is a directed edge with label r from node i to node j). When it is known, we note y the entry in the target column of the row corresponding to the target node (*i.e.*, the value to be predicted).

Graph neural network on RDB graphs. GNNs are DNNs designed so that their computational graph aligns with the data's graph structure, and many variations have been proposed (Kipf & Welling, 2016; Gilmer et al., 2017; Xu et al., 2018; Corso et al., 2020). Given an input $\boldsymbol{x} = (A, E)$, the first step is to embed the node feature information A as an initial representation

$$\boldsymbol{h}_i^0 = \text{Embed}(A_i) \quad \forall i, \tag{1}$$

where the embedding function $\text{Embed}(\cdot)$ is typically adapted to each node's type. The structural information E is then leveraged to update this representation using an iterative scheme whose details depends on which kind of GNN is being used. In the case of message passing neural networks (MPNNs), message passing layers (Gilmer et al., 2017) provide the node representation $\boldsymbol{h}_i^t$ at layer t

$$\begin{aligned} \boldsymbol{m}_i^t &= \sum_{(i,r,j)\in E} \text{Message}_{t-1}\left(\boldsymbol{h}_i^{t-1}, r, \boldsymbol{h}_j^{t-1}\right), \\ \boldsymbol{h}_i^t &= \text{Update}_{t-1}\left(\boldsymbol{h}_i^{t-1}, \boldsymbol{m}_i^t\right), \end{aligned} \tag{2}$$

where Message_{t-1} and Update_{t-1} are the message passing and node update functions on layer $t-1$, and $\boldsymbol{m}_i^t$ is the message representation. Note that the message's dependency on r is not actually used in this work[1].

Repeating this for T message passing layers provides the node-level representation $\boldsymbol{h}_i^T$, which encodes information up to the T-hops neighborhood of node i. If a graph-level representation $\boldsymbol{h}_g$ is required, it can be obtained by by applying a readout function Readout to the aggregate $\boldsymbol{h}^T$ of all the individual $\boldsymbol{h}_i^T$

$$\boldsymbol{h}_g = \text{Readout}\left(\boldsymbol{h}^T\right). \tag{3}$$

[1]The datasets we consider are such that, given an edge $(i, r, j) \in E$, the type of node i and the type of node j uniquely specify the value of r, so having Message_{t-1} depend on r would be redundant information.

Table 1: Main results on three datasets. Dataset sizes are shown in parentheses. We use 80-10-10 for train-valid-test split. We run each algorithm with 3 different random seeds and we report the mean and standard deviation of the ROC-AUC. When present, SSL pre-training is always performed on the full (unlabeled) training set, while fine tuning is done on a (labeled) fraction S.

SSL pre-training	Model	Acquire (160k)		Home Credit (307k)		KDD Cup (619k)	
		S=10%	S=100%	S=10%	S=100%	S=10%	S=100%
—	RF	66.12 ± 0.03	68.38 ± 0.01	69.45 ± 0.23	71.68 ± 0.29	73.45 ± 0.34	76.98 ± 0.11
	XGB	66.68 ± 0.00	68.53 ± 0.00	73.10 ± 0.00	77.72 ± 0.00	75.01 ± 0.00	**79.86 ± 0.00**
Built-in	TabNet	67.45 ± 0.08	67.61 ± 0.09	73.47 ± 0.65	77.89 ± 0.09	74.87 ± 0.58	78.16 ± 1.12
	TabTrans	67.23 ± 0.13	68.45 ± 0.03	68.48 ± 0.42	70.29 ± 0.28	71.40 ± 0.64	76.83 ± 0.13
	VIME	64.46 ± 2.96	67.20 ± 0.28	62.74 ± 0.86	70.64 ± 0.44	66.41 ± 1.07	73.60 ± 1.13
—	GCN	67.93 ± 0.01	70.17 ± 0.08	76.08 ± 0.08	78.36 ± 0.08	75.09 ± 0.12	79.52 ± 0.21
	PNA	**69.31 ± 0.07**	**71.79 ± 0.05**	**76.38 ± 0.16**	**78.92 ± 0.06**	75.23 ± 0.29	79.63 ± 0.03
Generative	GCN	66.88 ± 0.08	69.75 ± 0.03	75.87 ± 0.07	78.16 ± 0.16	75.27 ± 0.06	79.75 ± 0.09
	PNA	68.50 ± 0.07	71.48 ± 0.06	75.52 ± 0.19	78.60 ± 0.13	75.37 ± 0.11	79.72 ± 0.05
Contrastive: InfoGraph	GCN	68.11 ± 0.16	70.06 ± 0.19	75.11 ± 0.08	77.96 ± 0.08	75.34 ± 0.06	79.43 ± 0.08
	PNA	68.75 ± 0.18	71.39 ± 0.14	75.39 ± 0.33	78.76 ± 0.10	61.10 ± 0.34	79.31 ± 0.04
Hybrid: InfoGraph	GCN	67.43 ± 0.02	69.95 ± 0.05	75.44 ± 0.01	78.04 ± 0.05	75.38 ± 0.04	79.73 ± 0.04
	PNA	68.32 ± 0.13	71.51 ± 0.13	75.74 ± 0.06	78.65 ± 0.16	74.88 ± 0.03	**79.87 ± 0.11**
Contrastive: InfoNode	GCN	66.23 ± 0.22	69.90 ± 0.06	75.02 ± 0.46	77.77 ± 0.17	74.06 ± 0.10	78.94 ± 0.05
	PNA	67.35 ± 0.17	71.50 ± 0.14	65.52 ± 0.26	78.73 ± 0.06	64.78 ± 0.39	78.36 ± 0.28
Hybrid: InfoNode	GCN	67.15 ± 0.05	69.78 ± 0.08	75.70 ± 0.04	77.90 ± 0.06	75.74 ± 0.05	79.80 ± 0.08
	PNA	68.64 ± 0.07	71.71 ± 0.21	75.23 ± 0.04	78.70 ± 0.07	**75.84 ± 0.05**	79.64 ± 0.11

Finally, the prediction $\hat{y}$ for the target column of the target table can be obtained by applying a multi-layer perceptron (MLP) to the graph-level representation $\boldsymbol{h}_g$, which is the method favoured in (Cvitkovic, 2020). Alternatively, because of the special role played by the target node in the RDB graph, we may directly apply the MLP to the target node's representation $\boldsymbol{h}_0^T$.

3. Self-Supervised Learning on RDB Graph

We introduce five SSL strategies—one generative, two contrastive and two hybrid—allowing us to leverage our large unlabeled dataset to pre-train the GNN.

Generative SSL. Denoising tasks are one of the most widely-used generative SSL method. Here we adopt a simple strategy: mask-out a small fraction of the node attributes by replacing them by random values. Concretely, for each node i, a binary mask vector β of the same length as A_i is generated, a node j of the same type as i is randomly selected from the current batch, and the masked attributes are $A'_i = \beta \cdot A_i + (1-\beta) \cdot A_j$. The objective is then to recover the original attributes A from the noisy representation $\boldsymbol{h}'^T = \text{MPNN}(A', E)$. The actual loss $\mathcal{L}_G$ is a sum of mean squared errors for the continuous attributes and of cross entropies for the categorical ones.

Contrastive SSL: InfoGraph. In general, contrastive SSL maximizes the mutual information between two views of the data. Taking the two views to be the node- and graph-level representations has been widely explored, including InfoGraph (Sun et al., 2020). Concretely, a pair of node- and graph-level representations is positive if the node comes from that graph and negative if it comes from another graph of the batch. Given a function $f(\cdot,\cdot)$ (here the dot product) and noting Pos (resp. Neg) the set of positive (resp. negative) pairs, the EBM-NCE objective (Liu et al., 2021) is:

$$\begin{aligned}\mathcal{L}_{\text{C-InfoGraph}} = \mathbb{E}_{(i,g)\in\text{Pos}}\big[\sigma(f(\boldsymbol{h}_i^T, \boldsymbol{h}_g)\big] \\ + \mathbb{E}_{(i',g')\in\text{Neg}}\big[1-\sigma(f(\boldsymbol{h}_{i'}^T, \boldsymbol{h}_{g'})\big].\end{aligned} \tag{4}$$

Contrastive SSL: InfoNode. Potential issues have been associated with GNN (Alon & Yahav, 2020). Of those issues, "over-smoothing" states that node-level representations may become indistinguishable and prediction performance may thus severely degrade as the number of layers increases. Conjecturing that it may be desirable for a node to "remember about itself", we introduce InfoNode: a variation of InfoGraph with extra contrastive terms where the two views are the node's initial ($\boldsymbol{h}_i^0$) and final ($\boldsymbol{h}_i^T$) representations:

$$\begin{aligned}\mathcal{L}_{\text{C-InfoNode}} = \mathbb{E}_{(i,g)\in\text{Pos}}\big[\sigma(f(\boldsymbol{h}_i^0, \boldsymbol{h}_i^T) + \sigma(f(\boldsymbol{h}_i^T, \boldsymbol{h}_g)\big] \\ + \mathbb{E}_{(i',j')\in\text{Neg}}\big[1-\sigma(f(\boldsymbol{h}_{i'}^0, \boldsymbol{h}_{j'}^T)\big] \\ + \mathbb{E}_{(i',g')\in\text{Neg}}\big[1-\sigma(f(\boldsymbol{h}_{i'}^T, \boldsymbol{h}_{g'})\big].\end{aligned} \tag{5}$$

Hybrid objective. We follow Somepalli et al. (2021); Liu et al. (2021), where combining contrastive and generative SSL can augment the pre-trained representation. Using $\mathcal{L}_{\text{C}}$ to denote either of $\mathcal{L}_{\text{C-InfoGraph}}$ or $\mathcal{L}_{\text{C-InfoNode}}$, and writing α_0 and α_1 the coefficients for the generative and contrastive objectives, the resulting objective function is:

$$\mathcal{L} = \alpha_0 \cdot \mathcal{L}_{\text{G}} + \alpha_1 \cdot \mathcal{L}_{\text{C}}. \tag{6}$$

4. Experiments and discussion

Datasets and evaluation metrics. We consider the same 3 RDB datasets as in Cvitkovic (2020), all pre-processed with RDBToGraph. For these datasets, the predicted labels are binary and imbalanced, motivating the use of ROC-AUC.

Baselines and backbone models. We consider 2 decision models: random forest (RF) and XGBoost (XGB) (Chen & Guestrin, 2016). We consider 3 recent deep tabular models supporting SSL pre-training: TabNet (Arik & Pfister, 2021), TabTrans (Huang et al., 2020), and VIME (Yoon et al., 2020). We consider 2 backbone models GNNs: GCN (Kipf & Welling, 2016) and PNA (Corso et al., 2020). The readout function is an attention module from Li et al. (2015).

Main results. Our main results are presented in Table 1. We first emphasize the strength of XGBoost (Chen & Guestrin, 2016) on RDBs that were flattened with DFS (Kanter & Veeramachaneni, 2015), and how straightforward it is to apply such methods compared to DNNs: these facts are often missing in the DNN literature narrative, and embody the "unconquered castle" mentioned in introduction. We also reproduce past results (Cvitkovic, 2019; 2020) on the advantage of leveraging more of the RDB's structure through GNN-based supervised learning.

However, it is difficult to reach a clear conclusion concerning our main hypothesis that leveraging unlabeled data through SSL pre-training should improve downstream task performances. Consider for example the case of the KDD Cup dataset with $S = 10\%$ of labeled samples: for PNA, both contrastive SSL strategies underperform by more than 10 points when compared to a purely supervised version of the same model, yet the generative–InfoNode hybrid brings the best results. For the two other datasets, we observe that SSL pre-training typically results in weaker downstream performances than the corresponding fully-supervised models. Appendix A reports linear probing results, with even clearer examples of both negative and positive transfer.

Updated hypothesis and future work. For illustration purpose, consider a dataset composed of single-node graphs, each such isolated nodes bearing exactly 3 properties whose probability distribution factorizes as $\mathrm{P}(A) = \mathrm{P}(A_{00})\,\mathrm{P}(A_{01}, A_{02})$. With such an unlabeled dataset, the "best" that any SSL strategy could do is to learn $\mathrm{P}(A_{00})$ and $\mathrm{P}(A_{01}, A_{02})$. Due to limited model capacity and dataset size, tradeoffs must be made according to the inductive biases of the model and SSL strategy. Now further suppose that $\mathrm{P}(y|A) = \mathrm{P}(y|A_{00})$: any model capacity dedicated on learning $\mathrm{P}(A_{01}, A_{02})$ during pre-training has, in retrospect, been wasted. From the perspective of predicting y, A_{01} and A_{02} are "noise", all the "signal" resides in A_{00}.

This general argument has long been known: there is no free lunch (Wolpert & Macready, 1997). The novel, nontrivial core component of our hypothesis is that this is not just a contrived edge cases: RDBs' data distributions contain *traps*—"interesting-looking noise"—that some SSL strategies may "fall for", thus limiting the capacity that remains for useful signal. Figure 2 expands on this train of thoughts.

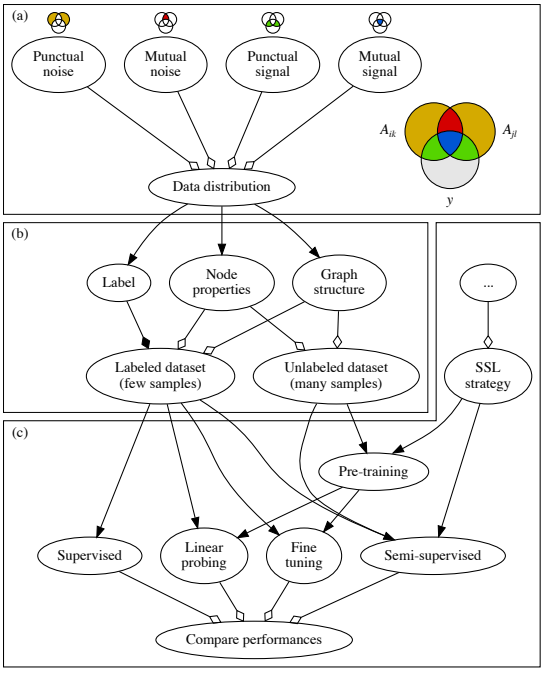


Figure 2: Plan for a future characterization work. Arrows indicate information flow while diamonds represent "and/or" aggregation (closed diamond means "always present"). **(a)** We distinguish four kinds of contributions to the data distribution using an information-theoretic perspective. We call *punctual noise* the information from an entry A_{ik} that is independent from both the label y and the other entries $A \setminus \{A_{ik}\}$, and *punctual signal* the information independent from other entries but shared with y. Similarly, we call *mutual noise* the information shared by two (or more) entries A_{ik} and A_{jl} but not with y, and *mutual signal* the one shared with y. **(b)** Conceptually, we can understand (labeled and unlabeled) datasets as being "sampled" from the data distribution. How much of the node properties and graph structure actually "makes it" to the model depends on the pre-processing and representation strategies: using the target table on its own gets less information than DFS (Kanter & Veeramachaneni, 2015), which itself gets less than RDBToGraph (Cvitkovic, 2020). But "more information" does not necessarily imply better downstream task performances: part of that information is "noise". **(c)** While supervised learning may directly learn to ignore "noise" contributions and focus on "signal" ones, only "punctual" and "mutual" information may be distinguished using unlabeled data alone. Our plan is to obtain analytical bounds and measure empirical performances for different training paradigms and/or SSL strategies. We will characterize real-world datasets and specify synthetic ones in terms of the four kinds of contributions listed above, as well as their relation to the graph structure. If our hypotheses are correct, semi-supervised approaches could dominate the other contenders by better discriminating "noise" from "signal" while leveraging more of the available data.

References

Alon, U. and Yahav, E. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*, 2020.

Arik, S. Ö. and Pfister, T. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 6679–6687, 2021.

Bahri, D., Jiang, H., Tay, Y., and Metzler, D. Scarf: Self-supervised contrastive learning using random feature corruption. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=CuV_qYkmKb3.

Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M., and Kasneci, G. Deep neural networks and tabular data: A survey. *arXiv preprint arXiv:2110.01889*, 2021.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020.

Chen, T. and Guestrin, C. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 785–794, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939785. URL http://doi.acm.org/10.1145/2939672.2939785.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on Machine Learning*, pp. 1597–1607, 2020.

Chen, X. and He, K. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15750–15758, 2021.

Corso, G., Cavalleri, L., Beaini, D., Liò, P., and Veličković, P. Principal neighbourhood aggregation for graph nets. *arXiv preprint arXiv:2004.05718*, 2020.

Cvitkovic, M. Supervised learning on relational databases with graph neural networks. In *Representation Learning on Graphs and Manifolds, ICLR 2019 Workshop, New Orleans, Louisiana, United States, May 6, 2019*, 2019. URL https://rlgm.github.io/papers/55.pdf.

Cvitkovic, M. Supervised learning on relational databases with graph neural networks. *arXiv preprint arXiv:2002.02046*, 2020.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

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. PMLR, 2017.

Ho, T. K. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pp. 278–282. IEEE, 1995.

Huang, X., Khetan, A., Cvitkovic, M., and Karnin, Z. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*, 2020.

Kadra, A., Lindauer, M., Hutter, F., and Grabocka, J. Well-tuned simple nets excel on tabular datasets. *Advances in Neural Information Processing Systems*, 34, 2021.

Kanter, J. M. and Veeramachaneni, K. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE international conference on data science and advanced analytics (DSAA)*, pp. 1–10. IEEE, 2015.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

Liu, S., Wang, H., Liu, W., Lasenby, J., Guo, H., and Tang, J. Pre-training molecular graph representation with 3d geometry. *arXiv preprint arXiv:2110.07728*, 2021.

Somepalli, G., Goldblum, M., Schwarzschild, A., Bruss, C. B., and Goldstein, T. Saint: Improved neural networks for tabular data via row attention and contrastive pre-training. *arXiv preprint arXiv:2106.01342*, 2021.

Sun, F.-Y., Hoffmann, J., Verma, V., and Tang, J. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *International Conference on Learning Representations, ICLR*, 2020.

Vasiloglou, N., Schleich, M., Makrynioti, N., Kordjamshidi, P., Pruhs, K., and Tavares, Z. Databases and AI (DBAI). [NeurIPS 2021 workshop call for papers] https://dbai-workshop.github.io/, 2021.

Wolpert, D. H. and Macready, W. G. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Yoon, J., Zhang, Y., Jordon, J., and van der Schaar, M. Vime: Extending the success of self-and semi-supervised learning to tabular domain. *Advances in Neural Information Processing Systems*, 33, 2020.

A. Linear Probing Results

Table 2 reports linear probing (LP) results as an indicator of the quality of the representations learned by different SSL strategies. Interestingly, contrastive methods taken on their own perform rather poorly from this linear probing perspective: the learned representations are at best comparable to random representations, and in many cases are much worse ("negative transfer"). While not being particularly impressive, generative and hybrid SSL results do not show this counter-intuitive behavior. Our understanding is that this generative/contrastive dichotomy is less visible in Table 1 because the models are given the opportunity to "unlearn" bad representations during fine-tuning. The mechanisms hypothesized in Figure 2 could *a priori* apply to both generative and contrastive SSL strategies, and investigating the mechanistic source of this dichotomy is part of our plan for a future characterization work.

Table 2: Continuation of Table 1 for linear probing results. In all cases, a linear classifier is trained on the representations of frozen models. For the first two rows, these frozen models have never been trained: they are still in their randomly-initialized state. In the remaining rows, the models were first pre-trained with the different SSL strategies before being frozen.

SSL pre-training	Model	Acquire (160k)		Home Credit (307k)		KDD Cup (619k)	
		S=10%	S=100%	S=10%	S=100%	S=10%	S=100%
Untrained (random init.)	GCN-LP	54.36 ± 0.12	54.16 ± 0.22	52.00 ± 0.08	56.37 ± 1.72	51.78 ± 1.03	58.95 ± 0.45
	PNA-LP	58.71 ± 0.73	61.68 ± 0.33	55.75 ± 1.96	62.76 ± 0.96	56.08 ± 2.43	62.05 ± 1.29
Generative	GCN-LP	56.78 ± 0.08	58.19 ± 0.11	55.85 ± 0.00	62.52 ± 0.05	59.38 ± 0.05	64.38 ± 0.01
	PNA-LP	64.85 ± 0.12	66.34 ± 0.06	61.53 ± 0.04	67.43 ± 0.04	65.65 ± 0.01	69.11 ± 0.06
Contrastive: InfoGraph	GCN-LP	53.97 ± 0.26	54.52 ± 0.53	52.75 ± 0.07	55.54 ± 0.06	51.59 ± 0.00	52.05 ± 0.04
	PNA-LP	50.83 ± 0.24	54.98 ± 0.22	52.68 ± 0.30	54.50 ± 0.69	51.20 ± 0.34	51.44 ± 0.10
Hybrid: InfoGraph	GCN-LP	57.47 ± 0.13	58.43 ± 0.09	54.20 ± 0.01	60.54 ± 0.04	59.00 ± 0.01	62.86 ± 0.03
	PNA-LP	63.78 ± 0.08	66.48 ± 0.02	59.57 ± 0.15	67.13 ± 0.14	55.39 ± 0.05	66.18 ± 0.02
Contrastive: InfoNode	GCN-LP	53.76 ± 0.06	54.12 ± 0.09	54.78 ± 0.00	56.00 ± 0.04	52.69 ± 0.02	53.98 ± 0.08
	PNA-LP	51.51 ± 0.33	51.64 ± 0.22	55.09 ± 0.49	55.80 ± 0.40	51.81 ± 0.42	53.12 ± 0.12
Hybrid: InfoNode	GCN-LP	56.02 ± 0.15	58.19 ± 0.11	55.69 ± 0.02	59.79 ± 0.09	57.33 ± 0.01	60.21 ± 0.02
	PNA-LP	65.42 ± 0.00	66.60 ± 0.03	59.35 ± 0.07	66.46 ± 0.15	64.52 ± 0.07	70.24 ± 0.02