Master MVA

Advanced Learning for Text and Graph Data

Course taught by Michalis Vazirgiannis

Molecule Retrieval with Natural Language Queries

Authors:

Benjamin Lapostolle* - `benjamin.lapostolle@polytechnique.edu`
Jean-Baptiste Himbert* - `jean-baptiste.himbert@polytechnique.edu`

*These authors contributed equally to this work

February 4, 2024

MATHÉMATIQUES
VISION
APPRENTISSAGE

Abstract

In this technical report, we present our solution to the Kaggle competition "Molecule Retrieval with Natural Language Queries", a challenge based on the Chebi-20 dataset whose aim is to pair chemical compounds to a textual description. We start by presenting the challenge, and we then develop our incremental approach to finally discuss in detail our contrastive learning framework and the main parameters of our experiments. We achieved a 94.3% MRR score in the public test set.

1 Presentation of the Challenge

Dataset: The dataset contains 33,010 pairs of molecules and their descriptions, divided into three sets: 80% for training, 10% for validation, and 10% for testing. The objective of this task is to identify the appropriate molecule based on a given natural language description. This task is referred to as "Text2Mol".

In the Text2Mol task, a text query is provided alongside a list of molecules, which lack any accompanying textual information and are typically represented as SMILES strings, graphs, or similar formats. The goal is to retrieve the molecule that corresponds to the provided query. The model is required to transform the information in the text description of the molecule into a semantic representation that can be directly used to identify the correct molecule. This task involves integrating two distinct types of information: the structured knowledge contained in the text and the chemical characteristics found in molecular graphs.

Evaluate Metric It is assumed that each description has only one correct and relevant molecule associated with it. The measure used to test the quality of the prediction is the mean reciprocal rank (MRR).

Previous works To address the Text2Mol task, a potential strategy involves utilizing multimodal foundation models. This approach involves leveraging two encoders and optimizing them using contrastive learning to create a coherent semantic space that aligns both types of data. Recently, several multi-modal foundation models have been developed to connect images and texts[2, 7]. In the original CLIP paper [8], they utilized a text embedding architecture and an image embedding architecture and trained their model by computing a tempered cosine similarity between the embeddings.

In the study titled *A Molecular Multimodal Foundation Model Associating Molecule Graphs with Natural Language*, the authors suggested using the Graph Isomorphism Network (GIN) [12] as the graph encoder, alongside a variation of the BERT model, Sci-BERT [1], which is more adept at processing text related to molecular properties. Subsequent research has identified ways to enhance the performance of these architectures further. Specifically, the paper *Graph Contrastive Learning with Augmentations* [13] explores various data augmentation techniques applicable to graph data and examines their impact on model performance. The authors highlight the challenge in determining the most suitable data augmentation technique for a specific dataset. A crucial aspect of this multi-modal foundation model is the formulation of the loss function, which influences how the encoders process data. References [5] and [9] delve into methods for computing the loss and strategies for implementing a loss function within the graph embedding space. This aims to ensure that similar graphs are positioned closely, rather than uniformly distributed in the latent space.

2 Main approach

In this section, we review the main components of our solution. We start by presenting the main improvements we added to the baseline and we then present the final solution.

2.1 Data augmentation

In graph-structured data, the principal augmentation methods include node dropping, edge perturbation, attribute masking, and subgraph sampling via random walks. Predicting which augmentation technique will yield optimal performance for a specific dataset is challenging (referenced in [13]). For example, node removal strategies, particularly based on node degrees, can significantly alter a molecule's chemical properties. Eliminating a hydrogen atom might minimally impact the graph's informational content, but removing an oxygen or nitrogen atom could fundamentally transform the data. In our research, we opt for a uniform node removal approach across the graph. The quantity of nodes to be removed is determined by sampling from a binomial distribution $B(n,p)$, where n is the total number of nodes in the graph. This approach allows for proportionally more nodes to be removed in larger graphs, maintaining a balance in the augmentation process.

2.2 Constrastive Loss Function

During training, we process a mini-batch comprising N molecular graphs $\{G_1,\dots,G_N\}$ with two distinct augmentations, resulting in $2N$ augmented graphs. These graphs are then input into a graph encoder, obtaining representation vectors $\{z_1^G, z_1^{\widetilde{G}}, \dots, z_N^G, z_N^{\widetilde{G}\}}$, where z_i^G and $z_i^{\widetilde{G}}$ represent the two augmented variants of the i-th graph G_i. Concurrently, the embeddings of the associated text are obtained, yielding representations z_i^T. The contrastive loss for $(z_i^G, z_i^{\widetilde{G}}, z_i^T)$ is $L = L_G + L_T$, with L_G and L_T expressed as (with the notation $z_{N+k}^G = z_k^{\widetilde{G}}$):

$$L_G(z_i^G, z_i^{\widetilde{G}}, z_i^T) = -\log\left(\frac{\exp(\mathrm{sim}(z_i^G, z_i^T)/\tau)}{\sum_{j=1}^{N}\exp(\mathrm{sim}(z_i^G, z_j^T)/\tau)}\right) - \log\left(\frac{\exp(\mathrm{sim}(z_i^{\widetilde{G}}, z_i^T)/\tau)}{\sum_{j=1}^{N}\exp(\mathrm{sim}(z_i^{\widetilde{G}}, z_j^T)/\tau)}\right), \quad (1)$$

$$L_T(z_i^G, z_i^{\widetilde{G}}, z_i^T) = -\log\left(\frac{\exp(\mathrm{sim}(z_i^G, z_i^T)/\tau)}{\sum_{j=1}^{2N}\exp(\mathrm{sim}(z_j^G, z_i^T)/\tau)}\right) - \log\left(\frac{\exp(\mathrm{sim}(z_i^{\widetilde{G}}, z_i^T)/\tau)}{\sum_{j=1}^{2N}\exp(\mathrm{sim}(z_j^G, z_i^T)/\tau)}\right), \quad (2)$$

where τ is the temperature parameter. The temperature was a parameter learned during the optimization instead of a fixed value, which significantly improved the performance. The similarity function, sim, calculates the cosine similarity between the vectors. The contrastive loss on the text 2 was chosen because it yields better results during the optimization [5]. We describe the graph contrastive loss in annexe.A.

2.3 Text Encoder

Text processing was a key part of the challenge: we had to ensure that our text encoder could produce robust yet accurate text features. As our text was scientific, we decided to leverage pre-trained models specific to scientific data. Our choice settled on "SciBERT" [1], a 110 million parameter model based on the BERT architecture and trained on scientific publications. Compared to the classic BERT, SciBERT features a special tokenizer enabling it to process scientific text more accurately. Scibert led to the biggest improvement compared to the baseline. In the final model, we added a fully connected layer at the end of the text encoder to have more flexibility on the embedding space. This eased training and allowed us to test for different text-graph joint-space dimensions.

2.4 Graph Encoder

To process the molecules - which are stored as graphs - we used a simple graph encoder using a series of convolutional layers followed by a pooling operator and a feed-forward network. The graph convolutions are designed to produce embeddings for the molecule nodes, and the pooling allows for creating a unique feature vector for the graph.

Convolution Operator: There exists a huge variety of convolution operators in graph neural networks, the most popular being GraphConv [6], GraphSAGE [3], and GraphAttention [10]. Motivated by our challenge involving not a node-level but a graph-level task, we decided to opt for the architecture with theoretically the most expressivity power: the graph isomorphism operator (also known as GIN) [12].

The GINConv operator was specially designed to solve one of the issues involving the common graph convolution operator: the expressivity test for graph isomorphism. The issue arises from the fact that operators like GConv and GraphSAGE fail to distinguish graphs that are not isomorphic, leading them to produce the same embedding. The GINConv operator, on the other hand, has been proven to be at least as powerful as the Weisfeiler-Lehman test for isomorphism.

More concretely, the GINConv operator aggregates node information according to equation 3. It uses a parameter ϵ which denotes the importance of the target node compared to its neighbors and a MLP acting on the aggregated features. An interesting aspect of the operator is that both the ϵ and the MLP parameters can be learned, making for a more universal convolution operator. In our case, the MLP was composed of two fully connected layers separated by a batch norm layer, as seen in Figure 2 in Appendix B.

$$h_i = MLP((1+\epsilon)x_i + \sum_{j\in\mathcal{N}_i} x_j) \tag{3}$$

Pooling Operator: However, convolution operators output node features: to produce graph embeddings, it is thus necessary to aggregate the information to produce a unique feature vector. To do so, we leverage pooling operators. Three types of pooling operators are commonly found in the graph literature, among which are the $\max$, the sum, and the $mean$ pooling. Still preoccupied with the expressivity power of our network, we decided to use the sum operator, which has the biggest discriminative power of all three.

Final encoder: The final architecture we used can be found in Figure 3. It features 4 GINConv layers, followed by a sum pooling and two fully connected layers. We experimented with many hyperparameters but finally settled on hidden dimensions of 512 for all the linear layers. Increasing the complexity of the MLP or increasing the number of GINConv layers didn't seem to increase performance.

2.5 Training Procedure

To train our models, we used the same train/val/test split as in the baseline and computed the evaluation metric on the valid set at each epoch. Illustrative training graphs with losses and validation scores can be found in Appendix C. We leveraged the models presented in sections 2.3 and 2.4 and used the contrastive loss from section 2.2. More details follow hereafter:

Learning rate: We tried to divert from the original learning rate of 2.10^{-5} with larger and smaller values, but our findings were not conclusive. We finally settled on two schemes: a cosine annealing scheduler starting at 2.10^{-5} and decreasing to 10^{-6} in 60 epochs and a "reduce on plateau" scheduler conditioned on the validation score, starting at 2.10^{-5} and decreasing to 2.10^{-7}. The "reduce on plateau" seemed a little better and allowed to train on more epochs.

Embedding space dimension: We experimented with different dimensions for the joint-feature space (512, 768, 1024) and found that the original 768 was the better choice. However, we chose to keep the final fully connected layer at the end of the text encoder as an additional projection.

Optimizing the graph encoder separately: As in Generative adversarial network (GAN), we propose to only train the graph encoder during the first epoch, as the text encoder was pretrained, and we use the graph contrastive loss only during that epoch. We found that keeping this loss hinders in the long run the performance.

Batch Size: The original CLIP paper [8] suggested that contrastive training should use the largest possible batch size. For our GPU capacity, this corresponded to a batch size of 64.

Number of Epochs: Our different models peaked at different epoch values, however we found that the best performance was reached around epoch 80. A good "reduce on plateau" scheduler allowed us to train up to 200 epochs and still gain performance.

Regularization: To improve robustness and prevent overfitting, we used a dropout layer before the final layer of our graph encoder. We found that a value of $p = 0.2$ was the best trade-off between robustness and training speed.

2.6 Ensemble strategy

We finally experimented with basic ensembles of our models. This consisted of cleverly combining the submission files, as we would usually do with logit models. We experimented with different ensemble strategies in the validation sets and we settled on two ensembling methods: a weighted sum aggregation based on each method score with and without a min-max scaling of the cosine similarities. While this strategy seems very simple and naive, it enabled us to gain $1 - 2\%$ in the final accuracy. Although more advanced ensembling methods could have been implemented, we didn't have the computing power to do so. A key part of the ensembling came with choosing the models to include. Our last ensembles gathered the prediction of 9 different models, whose diversity was essential. Amongst them, there were two GCNs, three small GINs, two big GINs, one GIN trained with data augmentation, and one GIN trained with Galactica. Their scores on the validation set were between 83% and 92%, and the ensemble was able to have a score of 94.3% on the public test score.

3 Other non-conclusive variations

3.1 Text Encoder

When it comes to the text encoder, we tested two other models alongside DistilBERT (used in the baseline) and SciBERT. We first tried to leverage the e5-mistral-7b-instruct model, a variant of the Mistral 7b model fine-tuned for embedding text [4, 11]. Being at the top of the MTEB benchmark and having been trained for academic publication clustering, we thought it might be of use for the challenge. Although we weren't able to fine-tune it for our specific task, an appealing aspect was the possibility of conditioning the model through instruction: by giving it appropriate information on the challenge and the task at hand, we believed it could perform as well as smaller more specific models such as SciBERT. However, the lack of an appropriate scientific tokenizer coupled with the impossibility of fine-tuning it made our experiments with it non-conclusive.

We secondly leveraged Galactica $125m$, the smallest model from the Galactica LLM family developed by Facebook trained on a mix of scientific corpus. Its small size enabled us to fine-tune completely, and we found that its performance was similar to SciBERT. However, its very different structure allowed for a better ensembling result at the end.

3.2 Graph Encoder

When it comes to graph encoders, we mainly tried two types of models: Graph Isomorphism Networks (GIN) and Graph Convolution Networks (GCN).

While GCNs (and even the same as in the baseline) performed well in the first stage of the challenge, we found that we were limited by the graph encoder when reaching above 80% of MRR. Stacking more Convolution layers proved to be not very efficient, as we suffered from the vanishing gradient problem. Implementing skip-steps helped a bit, and allowed us to reach a MRR of 88%. However, we found that the GIN model presented in Section 2 scaled better, and had a more stable training.

However, GCNs proved to be very useful in the ensemble phase, as they allowed for better diversity amongst learners. We even found out that poorly performing GCNs (with an MRR of 80%) were particularly useful in the final ensemble.

4 Main Results

In this section, we present our main results, and we keep track of our improvement timeline. Due to the limited time of the challenge, we couldn't perform an exhaustive grid search. We thus present the results by pair, following successive improvements.

We can see from Table 1 that only a few improvements - mainly the loss and the use of SciBERT - lead to a very high increase in performance, while all the other improvements were only marginal compared to it, and enabled us to only gain a few percents. One thing to note between the performances of the GCN and the GIN networks is that, while they performed relatively similarly, the gap was due to the increased stability of the GIN in the learning phase. They thus constituted a better base for further technical improvements. We finally see that the use of learning rate schedulers helped training a little bit, while data augmentation did not increase scores as much as we thought it would.

We summarize Table 1 in Figure 1 using an improvement timeline tracking our best solution. We can see that the use of SciBERT, followed by the improved loss, leads to the best performance improvement. This is coherent with our intuition that the text description required models and tokenizers tailored for scientific data. The fact that different scientific text encoders (SciBERT and Galactica-125m) lead to similar results further suggests it.

Conclusion

We thus presented our incremental approach and main results for the Kaggle competition entitled "Molecule Retrieval with Natural Language Queries", under the team name *Tempest*. Although we get very good results on the cross-retrieval problem, the next step would be to be able to cross-generate compounds and textual descriptions given the other. To do so, we could leverage our co-encoding scheme. This would be very beneficial in chemistry and biology to design custom drugs or precursors for specific illnesses or synthesis.

Table 1: Main results of our experiments

ExpName	TextModel	GraphModel	Loss	Scheduler	Augm	MRR
Baseline	DistBERT	GCN	Normal	No	No	34.8
SciBERTBaseline	SciBERT	GCN	Normal	No	No	47.4
NewBase	SciBERT	GCN	Improv[1]	No	No	82.4
NewBaseCosine	SciBERT	GCN	Improv[1]	Cosine	No	83.6
NewBaseReduce	SciBERT	GCN	Improv[1]	Reduce	No	88.6
NewBaseBig	SciBERT	BigGCN[2]	Improv[1]	Reduce	No	82.3
NewBaseBig+	SciBERT	BigGCN_{+}[2]	Improv[1]	Reduce	No	87.3
BaseGIN	SciBERT	GIN[3]	Improv[1]	Reduce	No	86.1
BaseGIN+	SciBERT	GIN_{+}[3]	Improv[1]	Reduce	No	91.9
BaseGIN+/Val[5]	SciBERT	GIN_{+}[3]	Improv[1]	Reduce	No	92.7
BaseGIN++	SciBERT	GIN_{++}[3]	Improv[1]	Reduce	No	90.4
BaseGINaug	SciBERT	GIN_{+}[3]	Improv[1]	Reduce	Yes	91.8
GINgalactica	Galactica	GIN_{+}[3]	Improv[1]	Reduce	No	89.6
GINgalactica+	Galactica	GIN_{++}[3]	Improv[1]	Reduce	No	90.4
Ensemble[4]	-	-	Improv[1]	-	-	**94.3**

[1] This refers to the loss improvement of Section 2.2.
[2] BigGCN and BigGCN+ refer respectively to GCN of increased size, with and without skip-steps to improve gradient flow.
[3] GIN_{+} and GIN_{++} refer to bigger version of our base GIN network. They both use a hidden dimension of 768 and GIN_{++} features an additional GINConv layer.
[4] The Ensemble experiment uses a variety of models with different hyperparameters. Refer to Section 2.6 to have a better description of the experiment.
[5] This experiment was trained on the train and validation sets, hence its higher score on the test dataset.

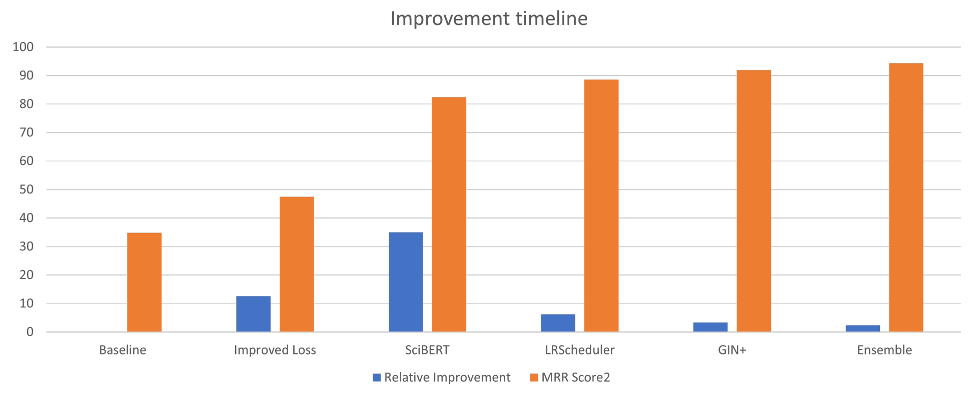

Figure 1: Sequential improvement for the different contributions. The plot tracks MRR scores and relative improvement.

References

[1] Iz Beltagy, Arman Cohan, and Kyle Lo. Scibert: Pretrained contextualized embeddings for scientific text. *CoRR*, abs/1903.10676, 2019.

[2] Nanyi Fei, Zhiwu Lu, Yizhao Gao, Guoxing Yang, Yuqi Huo, Jingyuan Wen, Haoyu Lu, Ruihua Song, Xin Gao, and et al. Xiang, Tao. Towards artificial general intelligence via a multimodal foundation model. *Nature Communications*, 13(1):1–13, 2022.

[3] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *CoRR*, abs/1706.02216, 2017.

[4] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

[5] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*, 2020.

[6] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

[7] Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, and et al. Wei, Furu. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *European Conference on Computer Vision*, pages 121–137. Springer, 2020.

[8] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020, 2021.

[9] Bing Su, Dazhao Du, Zhao Yang, Youjie Zhou, Jiangmeng Li, Anyi Rao, Hao Sun, Zhiwu Lu, and Wen Ji-Rong. A molecular multimodal foundation model associating molecule graphs with natural language. 2022.

[10] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

[11] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving text embeddings with large language models, 2024.

[12] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *CoRR*, abs/1810.00826, 2018.

[13] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. 2022.

A Graph Contrastive Loss

During mini-batch processing, we identified pairs of closely related graphs. To capitalize on this information, we employ a graph contrastive loss within the graph encoder. This loss encourages the encoder to organize the latent space in a way that forms clusters for similar graphs. The optimization objective involves minimizing the normalized temperature-scaled cross-entropy loss. Specifically, for the i-th graph, the graph-modal contrastive loss is defined as follows:

$$L(z_i^G, \tilde{z}_i^G) = -\log\left(\frac{\exp(\mathrm{sim}(z_i^G, \tilde{z}_i^G)/\tau)}{\sum_{j=1}^{N}\exp(\mathrm{sim}(z_i^G, \tilde{z}_j^G)/\tau)}\right), \tag{4}$$

where τ represents the temperature parameter. It is important to note that this final loss is computed across all samples within the mini-batch.

B GIN model

Here are two figures illustrating the final GIN$_+$ network used for the experiments.

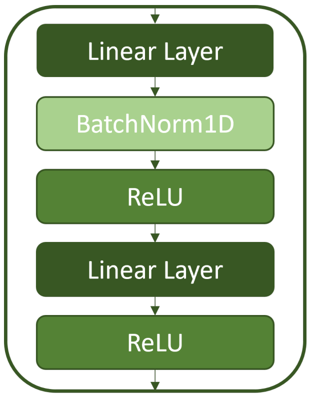

Figure 2: MLP of our GINConv operator

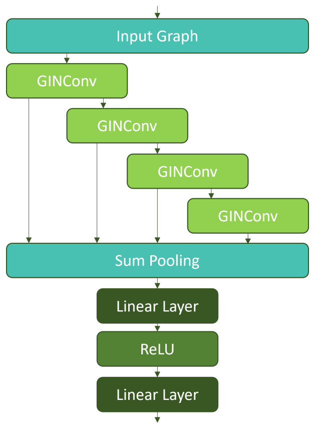

Figure 3: Architecture of our Graph Encoder

C Training plots

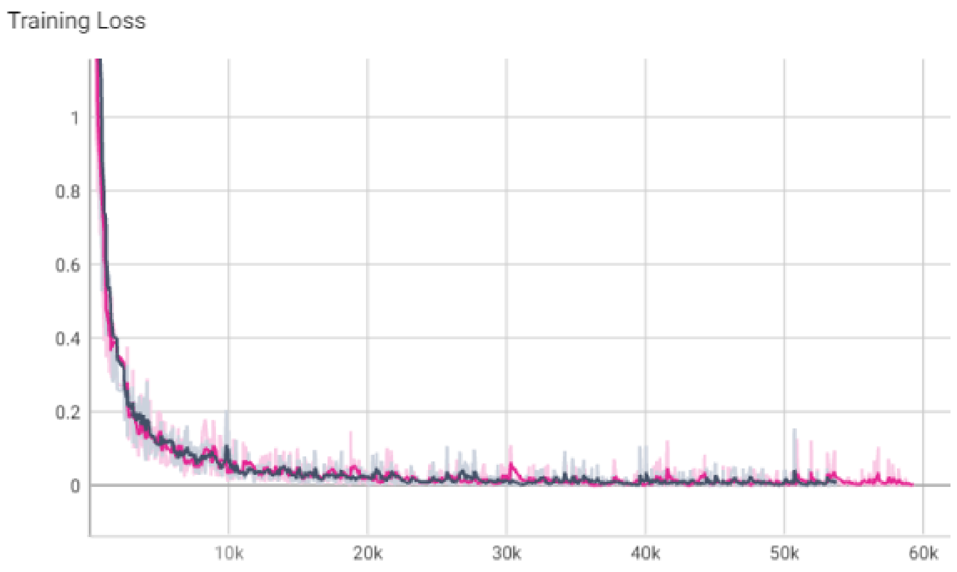

Figure 4: Training loss for two of our experiments: two GINs trained with SciBERT (grey) and Galactica (pink). X-axis is the number of optimization steps.

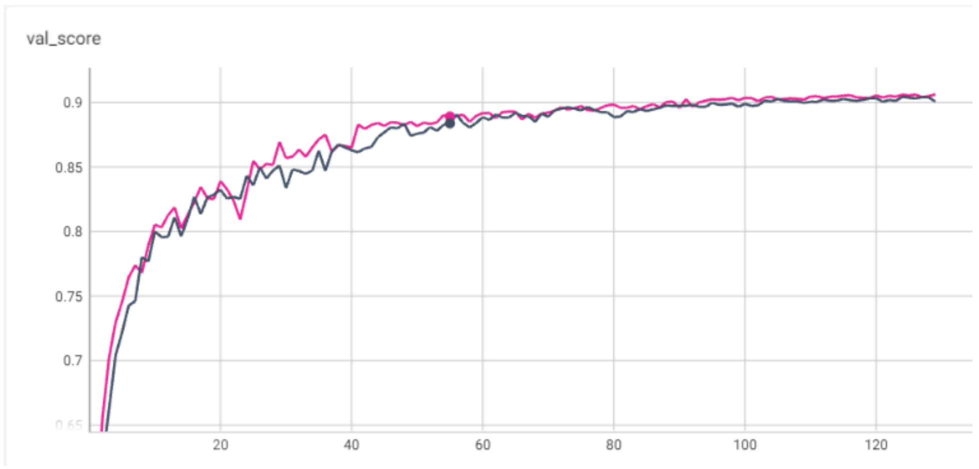

Figure 5: Training loss for two of our experiments: two GINs trained with SciBERT (grey) and Galactica (pink). X-axis is the number of epochs.