Deep Synergy Prediction

SujayKumar Reddy M

AstraZeneca R&D IT Intern

May 2024

Abstract

This document presents an overview of the Deep Synergy Prediction tasks for the Drug-Drug Interactions (DDI) problem. We develop a new model based on Graph Neural Networks (GNNs) to predict drug-drug synergy by using molecular structure data to build graph representations of drugs and applying GNNs to learn interaction patterns. Additionally, we incorporate the ULTRA model to leverage advanced learning-to-rank techniques, enhancing the performance and interpretability of the DDI prediction.

The model is evaluated against previous models in ChemicalX. We utilize existing functions and features from the `chemicalx` package for the construction and evaluation of the model. A Stratified K-Fold Cross Validation DataLoader is also implemented on top of `chemicalx`. This work is compared and analyzed across various datasets. By using molecules as graph data structures, we perform the inductive task of graph classification.

1 Datasets

The datasets used for the task of DSP (Deep Synergy Prediction) are DrugCombDB, DrugComb, and OncologyPharma from **?**. We utilized DrugCombDB and DrugComb for the evaluations, as the OncoPolyPharmacology dataset encountered loading issues with the `chemicalx` package. We applied the Multi-Head GAT-GIN Network model and the Ultra Neural Network model to these two datasets using different metrics. Specifically, the DrugComb dataset comprises 527,466 rows and 800 columns, while the DrugBankDDI dataset consists of 306,892 rows and 598 columns. More dataset statistics are available in Table 1 below.

2 Multi Deep DDS Model

The multi-Deep DDS model consists of a multi-head architecture, which fuses GIN (Graph Isomorphic Network) **?** and GAT (Graph Attention Network) **?**. GIN generalizes the WL test and achieves the maximum discriminative power among GNNs, while GAT operates on graph-structured data using self-attention layers to enhance model performance and improve the quality of graph embeddings. These embeddings are then forwarded into a feed-forward neural network.

The left side of the code refactoring below presents the original package code, which includes a dropout layer, two hidden layers, one input layer, and one output layer. It is important to note that the Vanilla Deep DDS model on the left uses 32 units for all hidden layers. Conversely, the right side of the refactored code gradually decreases the number of hidden units to transform the sparse data matrix into relevant features for task classification. Figure 1 below depicts the architecture used in the multi-Deep DDS model.

Dataset name	**Total features**	**samples**
DrugBankDDI	598	306,892
DrugComb	800	527,466
DrugCombDB	624	153112
TwoSides	522	399665

Table 1: Data Statistics for Deep Synergy model

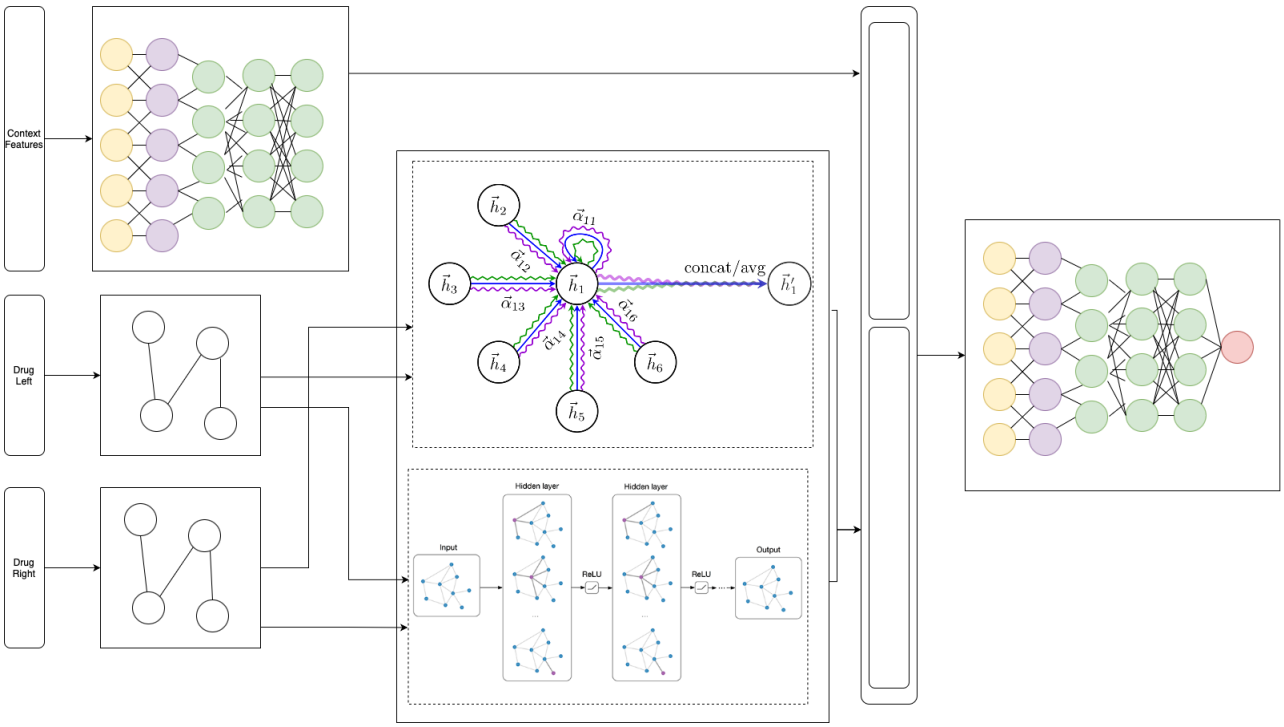


Figure 1: GAT-GIN Multi Head Architecture (Multi-Deep DDS model)

Original Code (chemicalx ‘deep_dds.py‘)

```
def forward(
        self, context_features: torch.FloatTensor, molecules_left: PackedGraph, molecules_right: PackedGraph
    ) -> torch.FloatTensor:
        """Run a forward pass of the DeeDDS model.

        :param context_features: A matrix of cell line features
        :param molecules_left: A matrix of left drug features
        :param molecules_right: A matrix of right drug features
        :returns: A vector of predicted synergy scores
        """


        # Run the MLP forward for the cell line features
        mlp_out = self.cell_mlp(normalize(context_features, p=2, dim=1))

        # Run the GCN forward for the drugs: GCN -> Global Max Pool -> MLP
        features_left = self._forward_molecules(molecules_left)
        features_right = self._forward_molecules(molecules_right)

        # Concatenate the output of the MLP and the GNN
        concat_in = torch.cat([mlp_out, features_left, features_right], dim=1)

        return self.final(concat_in)
```

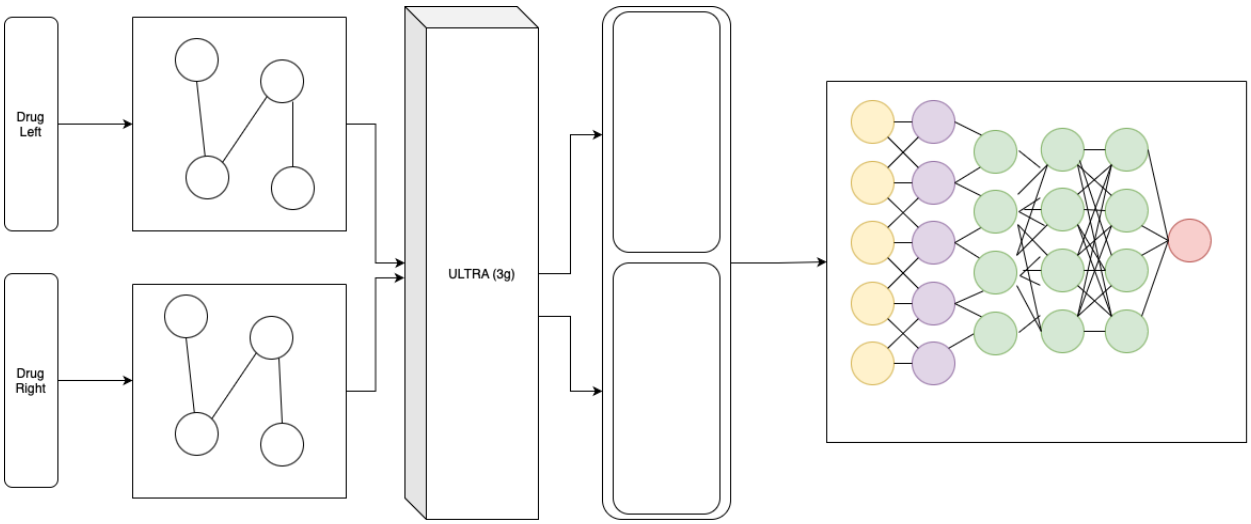


Figure 2: Ultra Neural Network

After Code Refractoring

```
def forward(
        self, context_features: torch.
    FloatTensor, molecules_left:
    PackedGraph, molecules_right:
    PackedGraph, plot=False,
    ) -> torch.FloatTensor:


        mlp_out = self.cell_mlp(normalize(
    context_features, p=2, dim=1))

        features_gat_left = self.
    _forward_molecules_gat(molecules_left)
        features_gat_right = self.
    _forward_molecules_gat(molecules_right)

        features_gin_left = self.
    _forward_molecules_gin(molecules_left)
        features_gin_right = self.
    _forward_molecules_gin(molecules_right)


        if plot is True:
            plot_embeddings(
features_gat_left,molecules_left,"GAT
LEFT DRUG")
            plot_embeddings(
features_gat_right,molecules_right,"GAT
 RIGHT DRUG")

            plot_embeddings(
features_gin_left,molecules_left,"GIN
LEFT DRUG")
            plot_embeddings(
features_gin_right,molecules_right,"GIN
 RIGHT DRUG")

        concat_in = torch.cat([mlp_out,
features_gcn_left, features_gcn_right,
features_gin_left,features_gin_right],
dim=1)

        return self.final(concat_in)
```

3 Ultra NN model

ULTRA (Unified, Learnable, TRAnsferable representations for any KG) is a foundation model for knowledge graph (KG) reasoning. A single pre-trained ULTRA model performs link prediction tasks on any multi-relational graph with any entity/relation vocabulary **?**. We utilize the ultra3g model from the Huggingface hub, testing each molecule of the Drugset from the `chemicalx` package.

We first use `chemicalx` to load the dataset. After loading the dataset, we use the `torchdrug` package to extract the edge lists from the Graph/PackedGraph data structures. These edge lists are then passed into the test function from Huggingface ultra3g to obtain the Hits@10 and MRR metrics from the ULTRA model. Given that this is an inductive task, we perform this process on every drug molecule (graph).

For predicting drug synergy, we use a Feed Forward model similar to the DeepSynergy model. We extract the graph embedding from h_{pred} of the model. The context features in this model are passed through but are mainly used for optimizing the training process in terms of speed. Figure 2 depicts the model architecture used for ULTRA NN.

```
    def forward(self,  context_features: torch.FloatTensor, molecules_left: PackedGraph,
    molecules_right: PackedGraph, plot=False,
    ) -> torch.FloatTensor:

        drug_embedding_left = self.get_embeddings(molecules_left)
```

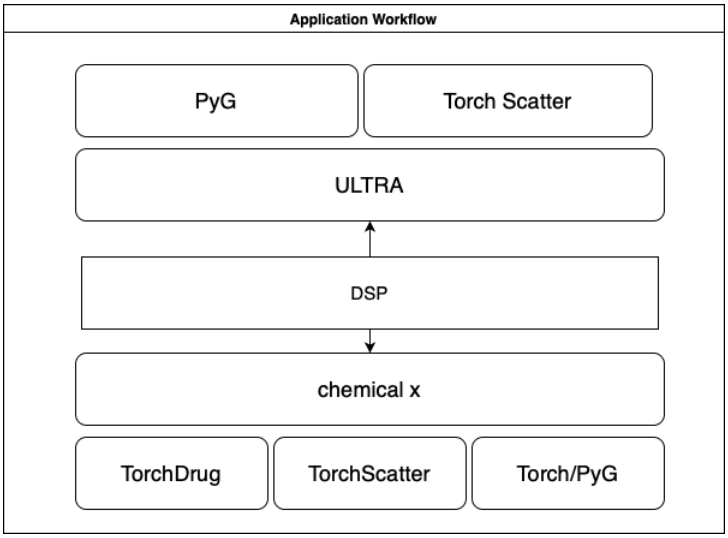


Figure 3: Workflow Architecture of the Code Construction

```
        drug_embedding_right = self.get_embeddings(molecules_right)

        out = self.lin1(torch.cat([drug_embedding_left,drug_embedding_right],dim=1))
        out = self.relu(out)

        out = self.lin2(out)
        out = self.relu(out)

        out = self.lin3(out)
        out = self.relu(out)

        out = self.lin4(out)
        out = self.relu(out)

        out = self.sigmoid(out)

        return out
```

4 Code Construction and Workflow

The code integrates multiple features using different dataloaders built on top of the `chemicalx.data` package `BatchGenerator` class, as illustrated in Figure 3. Configuration is handled via command line arguments specified by the user. We have implemented two DataLoader classes:

- **Vanilla DataLoader**: Used for both the Ultra and Multi-Deep-DDS models.
- **StratifyDataLoader**: The original dataset is split into 90% training and 10% validation. The 90% training data undergoes Stratified KFold operation, being trained and evaluated on each fold. The 10% validation data is assessed after the K-Fold training process.

4.1 Training Pipeline

As depicted in Figure 4, the training pipeline comprises two main components: the Normal Training Pipeline and the Stratified Training Pipeline. Both pipelines are interconnected and share many similarities. The Vanilla Training Pipeline is inherited within the `StratifyTrain`. Currently, we include two models, with the potential to add more, such as DeepDDS, to enhance the results.

4.2 Output and Results

The final results are saved in the `runs/` directory, capturing the user arguments, model weights, confusion matrix, embeddings of a single example, final metrics, Precision-Recall Curve, ROC Curve, and training metrics for each epoch or fold.

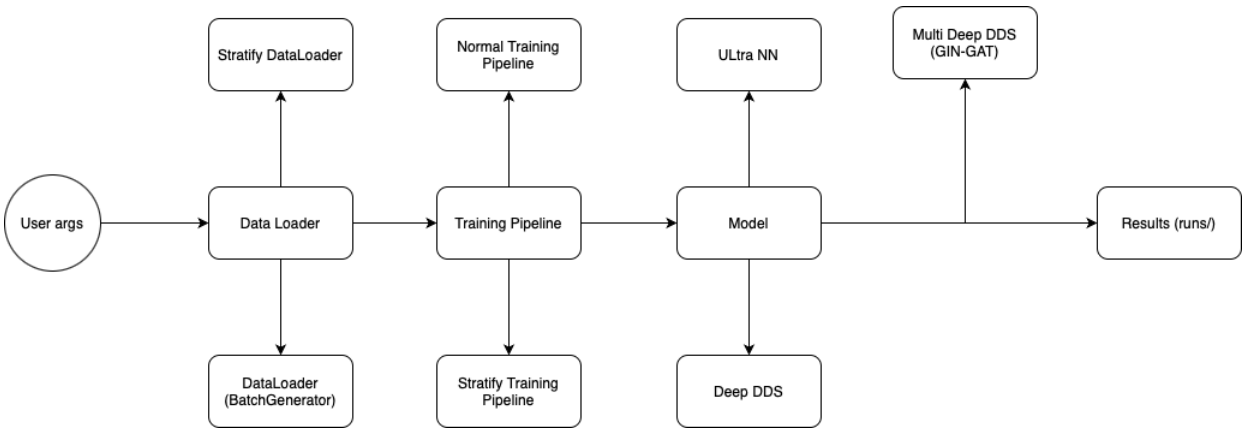
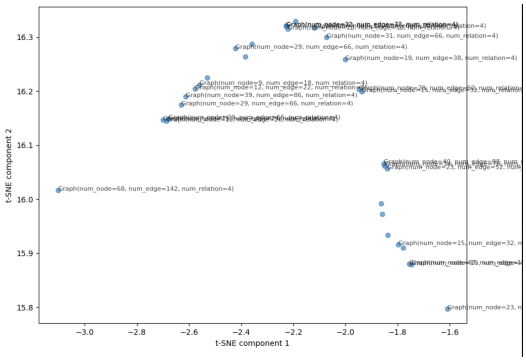
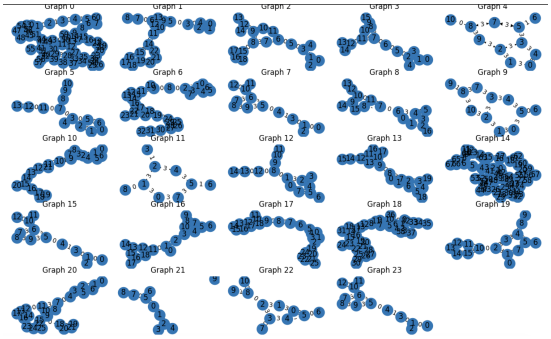


Figure 4: Process Workflow of our DSP application

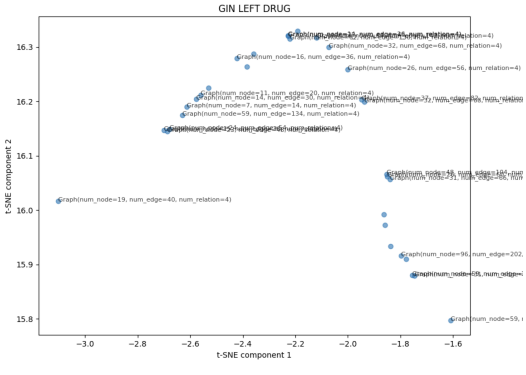
5 Results and Discussion



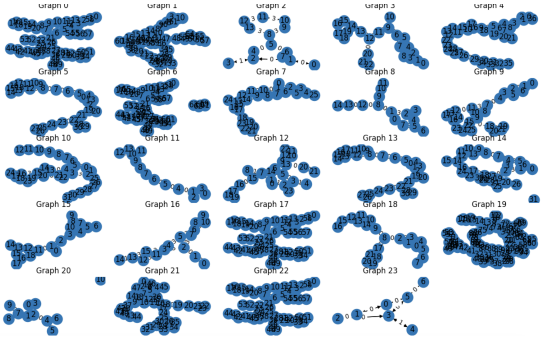
(a) GAT LEFT Drug Embeddings



(b) GAT LEFT Drugs



(c) GIN LEFT Embeddings



(d) GIN LEFT Drugs

Figure 5: GAT and GIN Head Network Embeddings and their corresponding Graph Left Drug Structures

Table 2 presents the Mean Reciprocal Ranking (MRR) and Hits@10 metrics for evaluating the prediction of missing links in the Knowledge Graph. These metrics measure the percentage of positive triples ranked within the top k positions. The DrugComb dataset achieves the highest MRR and Hits@10 scores. The high Hits@10 is notable given the inductive task performed on a relatively small graph rather than a typical knowledge graph. This outcome aligns with expectations based on the ULTRA paper **?** for the pre-trained ULTRA3G model.

Figure 5 illustrates the drug embeddings for the PackedGraph Left batch, while Figure 6 shows the embeddings for the PackedGraph Right batch. Figure 8 compares our model against other variants, including GAT-GCN, GCN-DeepDDS, GIN-GCN, and GAT-GIN. Among these, the GAT-GIN model variant achieves the highest accuracy across all evaluated models.

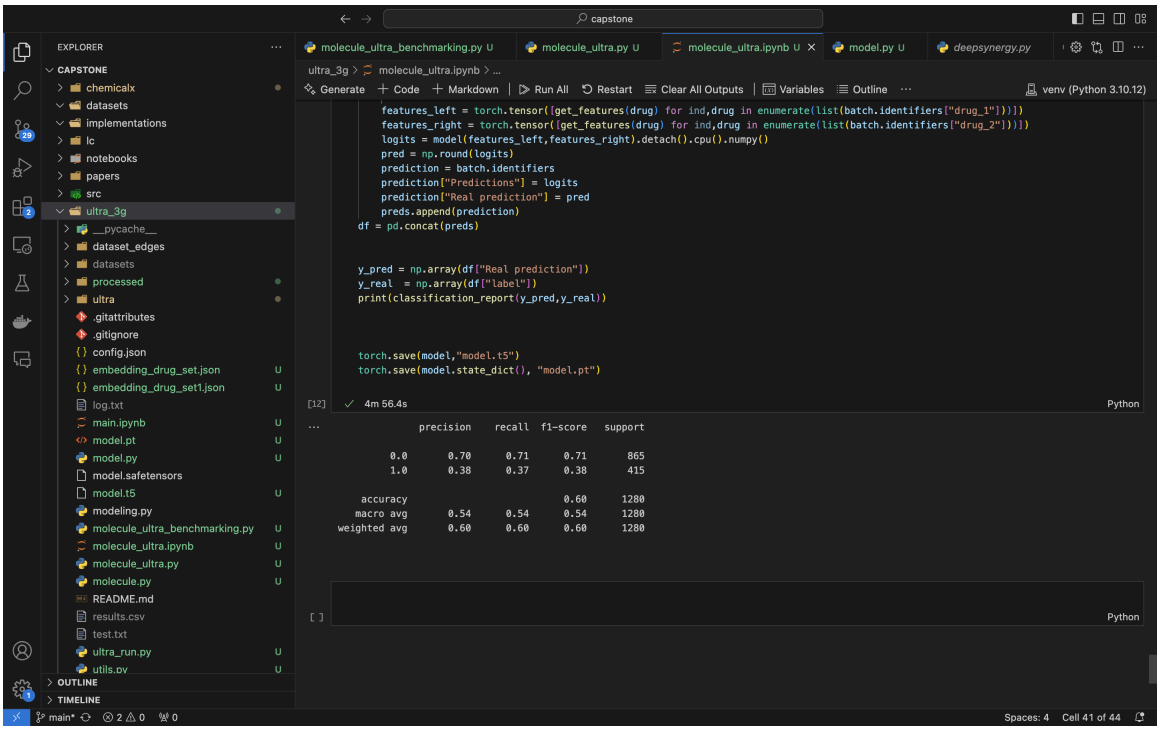
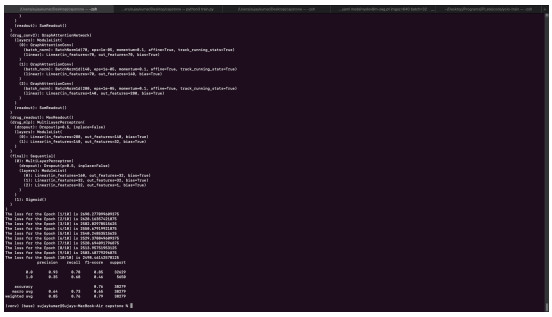
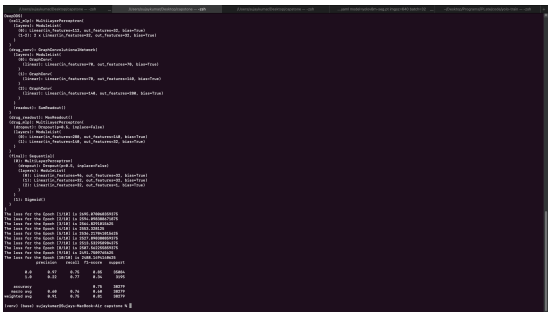


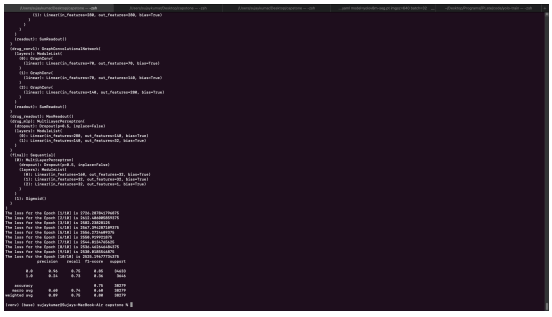
Figure 7: UltraNN model partial results



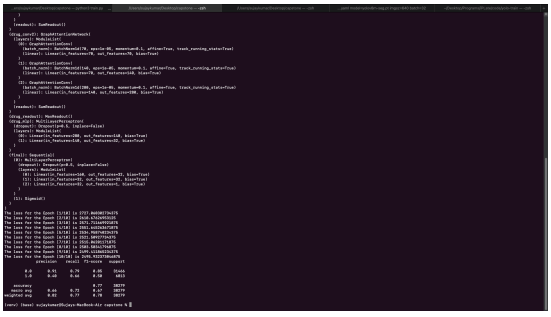
(a) GAT-GCN model results



(b) GCN-DeepDDS model results

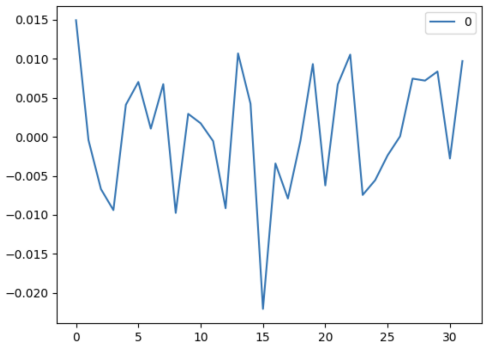


(c) GIN-GCN model results

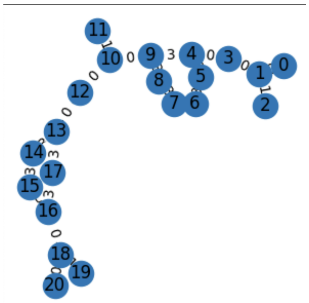


(d) GAT-GIN model results

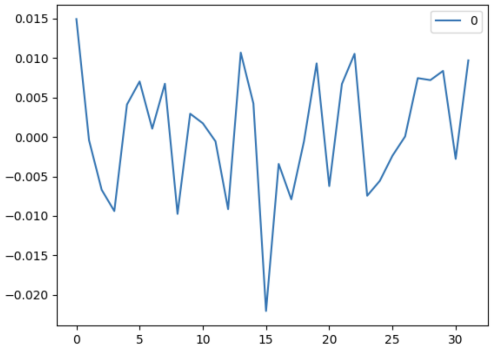
Figure 8: Model Comparisons with GAT-GIN model



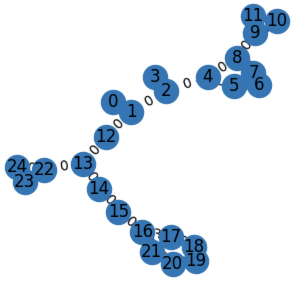
(a) GAT Drug Embeddings



(b) GAT Drug Graph

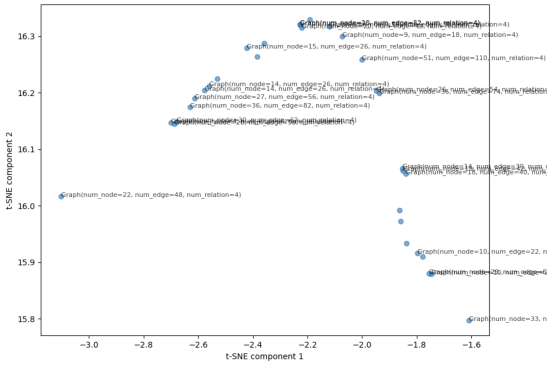


(c) GIN Drug Embeddings

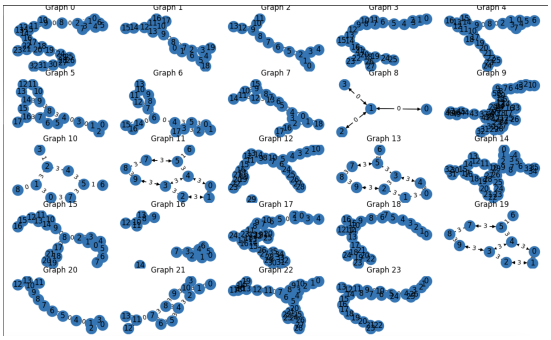


(d) GIN Drug Graph

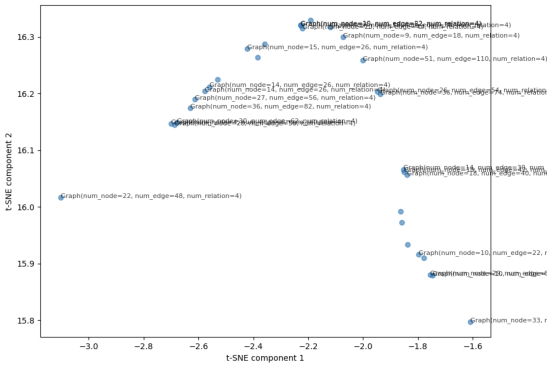
Figure 9: GAT and GIN Head Network Embeddings and their corresponding Graph Drug Structures for a single Drug



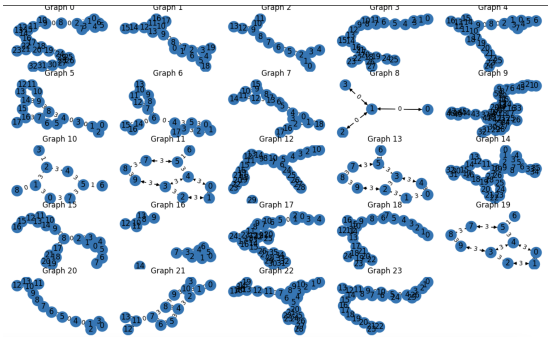
(a) GAT RIGHT Drug Embeddings



(b) GAT RIGHT Drugs



(c) GIN RIGHT Embeddings



(d) GIN RIGHT Drugs

Figure 6: GAT and GIN Head Network Embeddings and their corresponding Graph Right Drug Structures

Dataset	MRR	Hits@10
DrugComb	0.32455	0.924233
DrugCombDB	0.32176	0.921922

Table 2: ULTRA Performance of models on different datasets (averaged)

5.1 Improvements

We have successfully reduced overfitting in our model, similar to the DeepSynergy approach. Given that we are working with molecular drugs, our model offers a degree of intuitive explainability regarding its predictions.

5.2 Trade-offs

The computational complexity of UltraNN is significantly higher compared to GAT-GIN networks. Each epoch or model pass takes approximately 30-40 minutes. This increased complexity can be evaluated using GFLOPs for comparison.

6 References

1. Rozemberczki, Benedek, et al. "Chemicalx: A deep learning library for drug pair scoring." Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 2022.

2. Xu, Keyulu, et al. "How powerful are graph neural networks?." arXiv preprint arXiv:1810.00826 (2018).

3. Veličković, Petar, et al. "Graph attention networks." arXiv preprint arXiv:1710.10903 (2017).

4. Galkin, Mikhail, et al. "Towards foundation models for knowledge graph reasoning." arXiv preprint arXiv:2310.04562 (2023).