CTU student No.: 539347

I Wayan Iswara Jay Junior
CTU email: iwayajay@cvut.cz
TUM email: iswarajay.junior@tum.de

Final Project: GNN for Multiclass Enzyme Classification

1 Introduction

Computational rational enzyme design may offer an alternative option for limiting experimental trials needed for the often-laborious enzyme engineering cycle. Various task can be solved with machine learning (ML) models adapted to the protein design, ranging from structure prediction, enzyme discovery and classification, and enzyme properties prediction (Tripp, 2024). One recent ML approach for these tasks is Graph neural networks (GNNs) for its suitability for biological structures like protein that can be represented as 2D, or 3D graph. Commission (EC) number system, which assigns enzyme functions using four hierarchical digits is important for the annotation and classification of enzymes in the preamble part of enzyme engineering. In this project, we want to develop a simple GNN model to classify the top-level EC class (EC 1.X.X.X – EC 6.X.X.X). ENZYMES dataset from TUDataset was used and our success criteria were (1) test accuracy ≥0.75, (2) class-wise F1-score >0.7, and (3) training memory ≤100 MB.

2 Methods

2.1 Dataset and Experimental Setup

ENZYMES public dataset was used for this project (Morris, 2020). The dataset consists of 600 graphs representing the tertiary structure of enzymes. Each graph is a static graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and nodes represent secondary structure with 21 total attributes. Node attributes consist of 2 groups, 3 attributes represent one-hot encoding of secondary structure type (helix, sheet, or turn) and 18 attributes represent its physico-chemical properties. Edges were made between two nodes if they are neighbours along the amino acid sequence or one of three nearest neighbours in 3D space.

Universal experimental protocol was used during each step of model development. Dataset was split into 8:1:1 train:validation:test set with 32 samples batch size. For all training, AdamW optimizer with learning rate of 0.001 was used. Each training run was varied using different random seeds and done in 100 epochs, except for the training for the final (best) model which uses 200 epochs. Comparison between variations was done using paired t-test with p = 0.05.

2.2 Model Development

Model development starts with selecting the GNN convolution algorithm for a base model. The base model is described in **Figure 1a**, and 4 GNN convolution algorithm was tested (GCN, GraphSAGE, GIN, and GATv2). The result of test set accuracy is shown in **Figure 2a** and GraphSAGE algorithm was picked for hyperparameter tuning.

I Wayan Iswara Jay Junior
CTU email: iwayajay@cvut.cz
TUM email: iswarajay.junior@tum.de

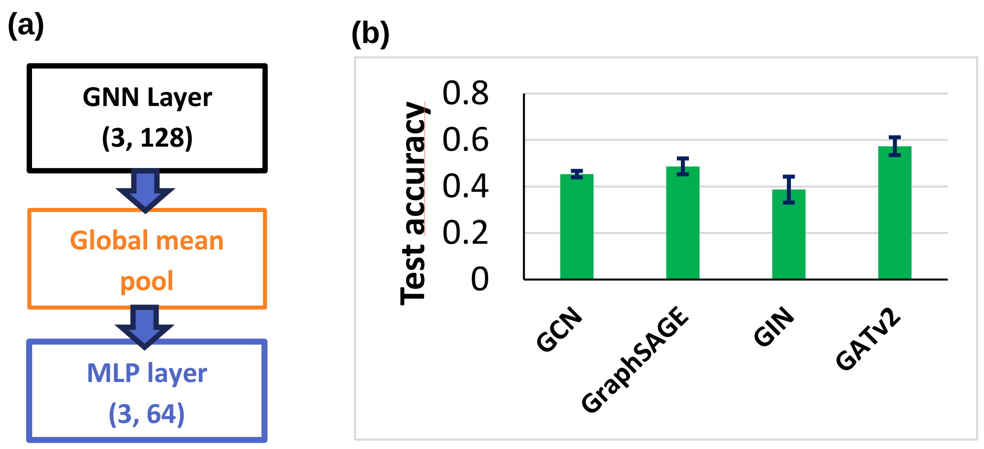

Figure 1: a. Base model architecture for GNN algorithms selection, for each neural network blocks, (number of layers, hidden channels) is notated; b. Test set accuracy for tested GNN algorithms (5 runs), error bars represent SD.

Hyper-parameter tuning was done by optimizing model's GNN design space. We used sequential method with 5 runs using the following order: 1. Dropout after each GNN layer in {0.0, 0.2, 0.3, 0.5}; 2. Normalization algorithm after each GNN layer in {None, Batch, Layer, Graph}; 3. Jumping knowledge algorithm for GNN block in {None, concatenate, maximum}. Additionally, the effect of model complexity was evaluated by training model with number of GNN layers in {3, 4, 5, 6} and number of GNN hidden channels in {64, 128, 256}.

3 Result

During base model screening, GATv2 yields the best test set accuracy (0.573) and significantly higher than GraphSAGE as the second best (0.487). However, it also came with a cost of ~144.6 MB maximum training memory compared to GraphSAGE's ~38.5 MB. Therefore, we used GraphSAGE as a winning option for hyper-parameter tuning for its balance between performance and efficiency.

Figure 2 shows the result of design space optimization of the GraphSAGE model. We found that adding modest dropout and normalization significantly improved accuracy. For example, using 0.2 dropout probability and BatchNorm resulted in 9.1% and 11.0% relative increase in test set accuracy compared to without it. Meanwhile, incorporating jumping knowledge yielded smaller boost, where 'maximum' aggregation method gives 3.1% increase in test set accuracy compared to model without it. In summary, the optimal GNN convolutional layer design space was set to dropout, normalization, and jumping knowledge parameters in {0.2, Batch, maximum}.

CTU student No.: 539347

I Wayan Iswara Jay Junior
CTU email: iwayajay@cvut.cz
TUM email: iswarajay.junior@tum.de

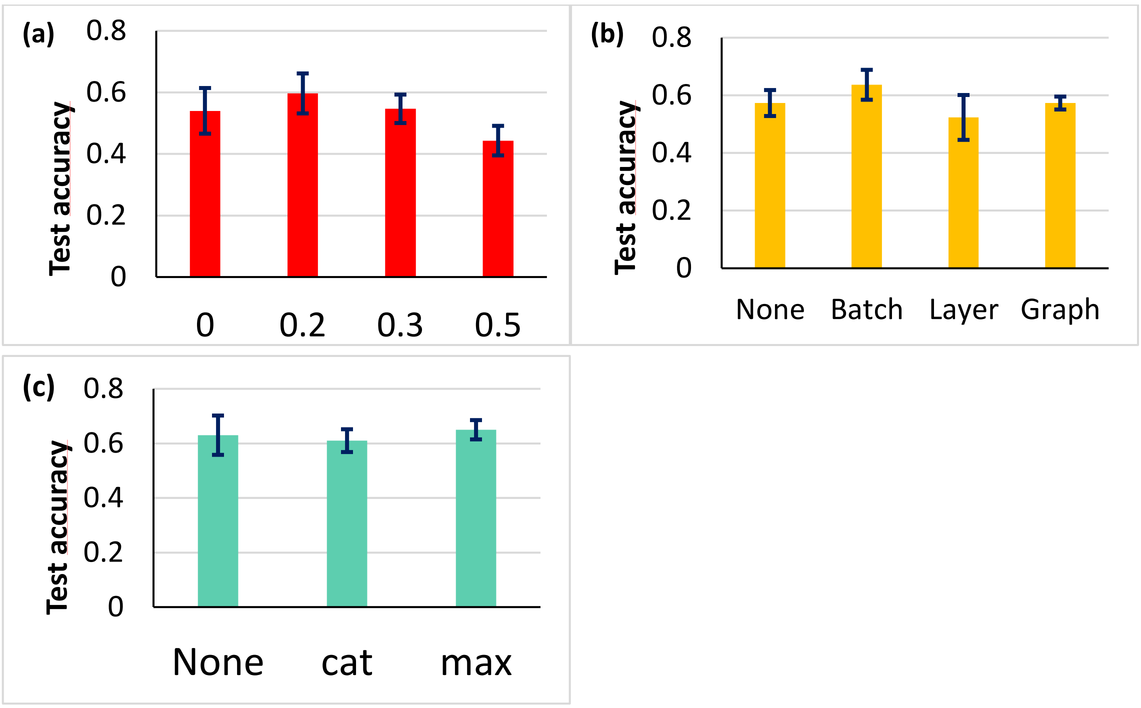

Figure 2: GraphSAGE model sequential hyper-parameter tuning result based on dropout probability (a), normalization type (b), and jumping knowledge (c) (5 runs). Error bar represents SD.

On this setting, the effect of model complexity on its performance was investigated (**Figure 3**). We found that increasing model complexity by adding layers or hidden dimensions tends to produce negative results on test set accuracy. Interestingly, simpler model (64 hidden channels, 3 layers) produces the best test set accuracy at 0.728. This may mean that in the case of training data used, smaller model is better at generalizing small dataset since it is more driven to learn meaningful feature due to implicit regularization. Therefore, we picked the (64, 3) configuration for our best model and train it to get the final model performance . The final GraphSAGE model performance is described in **Table 1** with final set accuracy of 0.783. Most classes exceeded the 0.70 f1-score goal, except EC 4.X.X.X (0.667). Overall, the model met objective (1) and (3) (accuracy and memory) and nearly met (2).

Table 1: Best model performance on test set.

Class	n sample	True Positive	f1-score
EC 1.X.X.X	10	6	0.706
EC 2.X.X.X	10	7	0.823
EC 3.X.X.X	10	10	0.800
EC 4.X.X.X	**10**	5	**0.667**
EC 5.X.X.X	10	9	0.818
EC 6.X.X.X	10	10	0.833
Accuracy			**0.783**

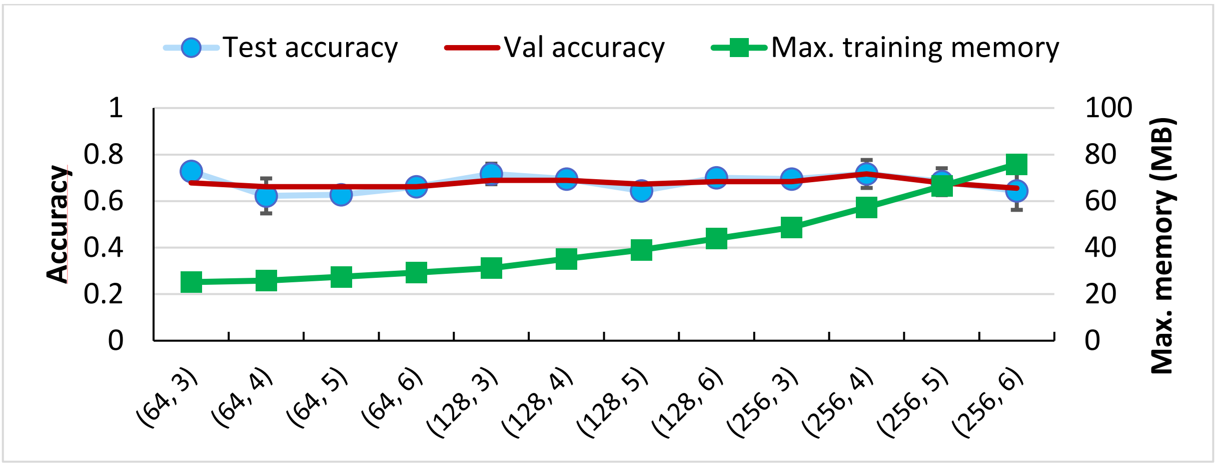

Figure 3: GraphSAGE model sequential hyper-parameter tuning result based on dropout probability (a), normalization type (b), and jumping knowledge (c) (5 runs). Error bar represents SD.

Visualization of selected good and bad model predictions are shown in **Figure 4**. Visually, no consensus pattern can be seen based on graph 2D representation or node secondary structure type. Therefore, exploring model explainability using GNN explainer algorithms may provide a better understanding of the underlying patterns.

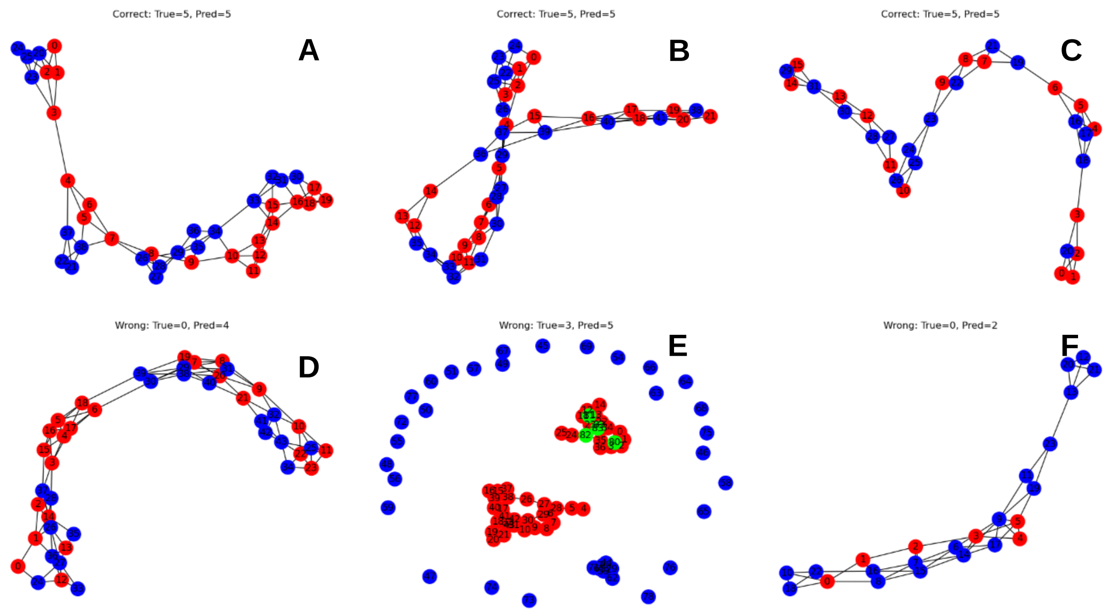

Figure 4: Example of good (A-C) and bad (D-F) predictions of the final GNN model. Red, blue, and green nodes notate helix, sheet, and turn secondary structures of the protein, respectively.

I Wayan Iswara Jay Junior
CTU email: iwayajay@cvut.cz
TUM email: iswarajay.junior@tum.de

4 Conclusion

We developed a simple GNN framework for multiclass enzyme classification on the ENZYMES dataset. Model screening showed that GraphSAGE algorithm achieved the best performance based on trade-off between accuracy and training memory. By tuning dropout, normalization, jumping knowledge, and model size, our final GraphSAGE model attained 0.783 test accuracy and good class-wise F1-scores (>0.7 on 5 of 6 classes) while keeping training memory <30 MB. These results closely fulfil the success criteria and suggest that the model is a good start for this task. Future improvements may include dataset augmentation, enrichment, or using more feature engineering. Using GNN explainer to evaluate bad predictions can become useful for future improvements and benchmarking more algorithms and pretrained model are also solid options for developing deeper classification solutions.

Reference

Tripp, A., Braun, M., Wieser, F., Oberdorfer, G. & Lechner, H. Click, Compute, Create: A Review of Web-based Tools for Enzyme Engineering. ChemBioChem e202400092 (2024) doi:10.1002/cbic.202400092.

Morris, C. *et al.* TUDataset: A collection of benchmark datasets for learning with graphs. Preprint at https://doi.org/10.48550/arXiv.2007.08663 (2020).