Test-time Adaptation for Graph-based Molecular Solubility Prediction

Philipp Sepin
165.164 Selected Topics in Theoretical Chemistry

June 1, 2025

Abstract

Molecular solubility prediction is a critical task in drug discovery, but models often struggle with out-of-distribution data. This project addresses this challenge by implementing test-time adaptation techniques for graph neural networks and applying them to molecular solubility prediction.

This project was carried out as part of the seminar **165.164 Selected Topics in Theoretical Chemistry** at TU Wien, under the supervision of Prof. Esther Heid.

1 Introduction

Molecular solubility prediction is a critical task in drug discovery, directly impacting a compound's bioavailability and therapeutic potential. Experimental solubility measurement requires substantial time and resources, making computational prediction essential for screening large molecular libraries in early-stage drug development [3].

Recent advances in solubility prediction have been driven by deep learning architectures and molecular embedding approaches. Feature-based neural networks, graph-based neural networks (GNNs), and structural attention methods have emerged as powerful predictive models. These approaches leverage molecular structure representations to capture complex structure-property relationships [3].

However, when there is a certain distribution shift between the training and test data, as with *OChemUnseen* [3] and *AqSolDB* [4], these models often struggle to generalize. This project aims to solve this by utilizing test-time adaptation (TTA) for graph neural networks (GNNs) to shift the test set distribution towards the training set distribution, thereby improving generalization.

TTA describes the process of training a model on a source domain, and then adapting it to a target domain at test time. It has been applied in various domains, such as semantic segmentation, object detection, medical image processing, video depth prediction, question answering, sentiment analysis, entity recognition, speech processing, social network analysis, as well as in protein and enzyme classification [2, 1].

2 Methods

2.1 Dataset

For this project, the *OChemUnseen* dataset [3] was used for training. It contains about 8000 molecules as SMILES strings, along with their solubility values. For testing, the *AqSolDB* dataset [4] was used, which contains about 2000 molecules as SMILES strings, along with their solubility values. This dataset is fully orthogonal to the training dataset.

The SMILES strings were converted to molecular graphs using the RDKit library [**?**], and one-hot encoded node and edge features were added. The node features included element type,

number of bonds, electric charge, aromaticity atomic mass, and orbital hybridization, while the edge features included bond order, aromaticity, conjugation, and whether the bond is in a ring. The graphs were then converted to PyTorch Geometric data objects.

2.2 Model

The model used for this project is a Y-shaped architecture, which consists of a shared encoder, which branches into a decoder and a prediction head.

The encoder is a convolutional bidirectional message passing neural network (MPNN), which is a GNN that applies convolutional operations to aggregate information from neighboring nodes in both directions through iterative message passing. It consists of three graph convolutional layers, followed by a global pooling layer that aggregates the node features into a single embedding vector for each graph. This embedding vector is an information-dense representation of the molecular graph. A 2D projection of the embedding space with corresponding solubility values is shown in Figure 1.

The decoder consists of two fully connected layers that reconstruct node and edge features from the embedding vectors. The prediction head also employs two fully connected layers that map the same embedding vectors to the predicted solubility value, creating a multi-task learning architecture.

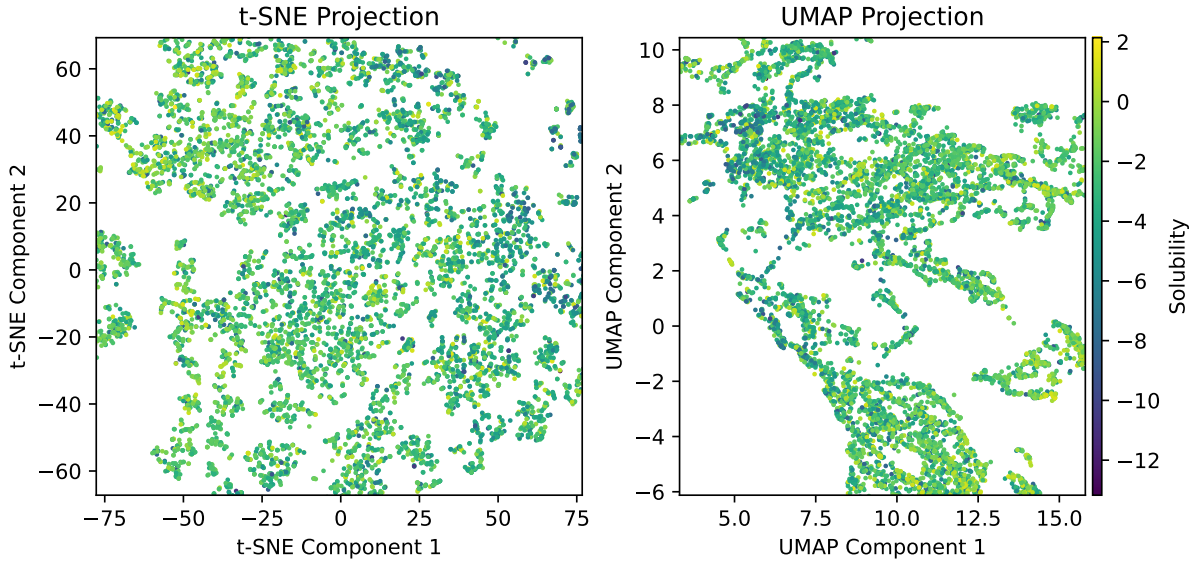

Figure 1: 2D projection of the embedding vectors with solubility.

2.3 Training

Following our architecture design, the model can be trained on two tasks. The first one is a self-supervised task, were node and edge features are denoised and reconstructed. For this, the features were pertrubed by randomly flipping a percentage of the one-hot encoded node and edge features. The encoder then learns to create an information-dense representation in form of the embedding vector, from which the decoder learns to reconstruct the denoised node and edge features. The second task is the supervised task, where the model learns to predict the solubility value from the embedding vector.

As in literature, both tasks are trained simultaneously by combining the losses [1, 5]. We followed this approach by using a weighted sum of the two losses, where the weight was optimized to $\alpha = 0.3$ for the self-supervised task and $1 - \alpha = 0.7$ for the supervised task. The model was

trained for 20 epochs with a batch size of 512, using the Adam optimizer with a learning rate of 0.0026, weight decay of $5 \cot 10^{-6}$, and dropout of 0.3.

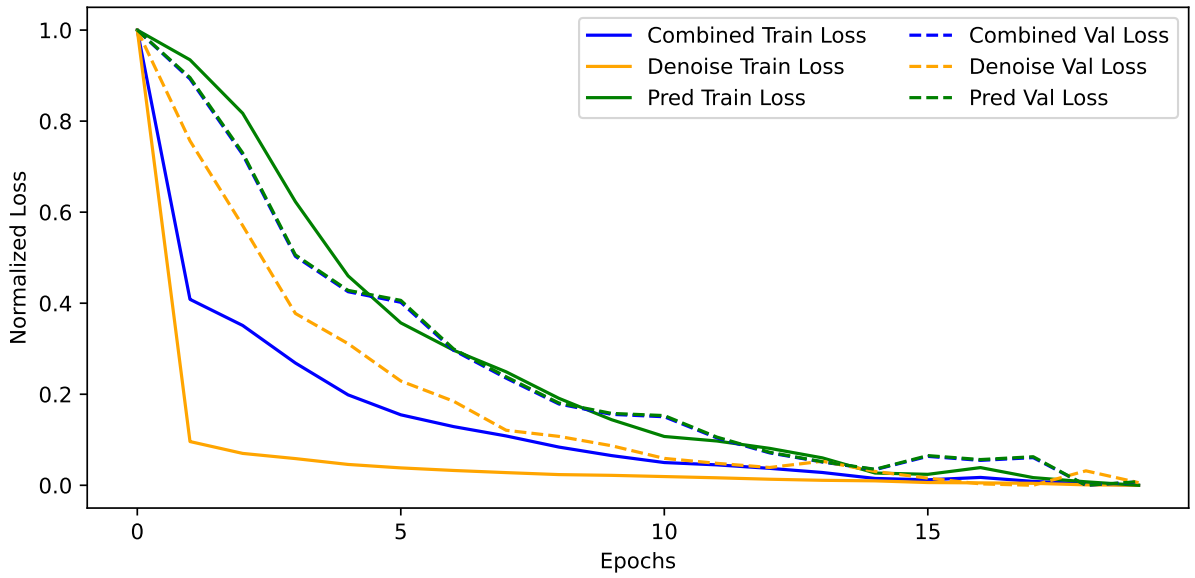

Figure 2: Training and validation losses.

2.4 Test-time Adaptation and Prediction

For TTA, each test sample is processed individually, with the encoder being adapted to the specific molecular structure through a single gradient descent step on the self-supervised loss. The adapted model is then utilized to predict the solubility using the standard prediction head. Following this, the model is set back to its original state, ready for the next test sample. The step size for the gradient descent step was optimized to $2.1 \cdot 10^{-4}$. Doing this, the embedding vector of the test samples should be shifted towards the training set distribution, allowing the model to generalize better. This can be seen in the 2D projections of the embedding vectors in Figure 3 and 4.

3 Results

Model	Validation RMSE	Test RMSE
Model without TTA	1.05	1.41
Model with TTA	1.06	1.41
Reference Model	2.09	2.26

Table 1: Performance comparison of different model configurations.

As shown in Table 1, TTA did not improve the performance of the model on the test set, which is suprising, given that the test set centroid clearly shifts towards the training set centroid after TTA, as shown in the 2D projections of the embedding space in Figure 4. Maybe this would work better for other datasets with a larger distribution shift.

For comparison, a reference model trained solely on the prediction task without any self-supervised learning (SSL) was also included. This showed that SSL clearly improves the prediction performance.

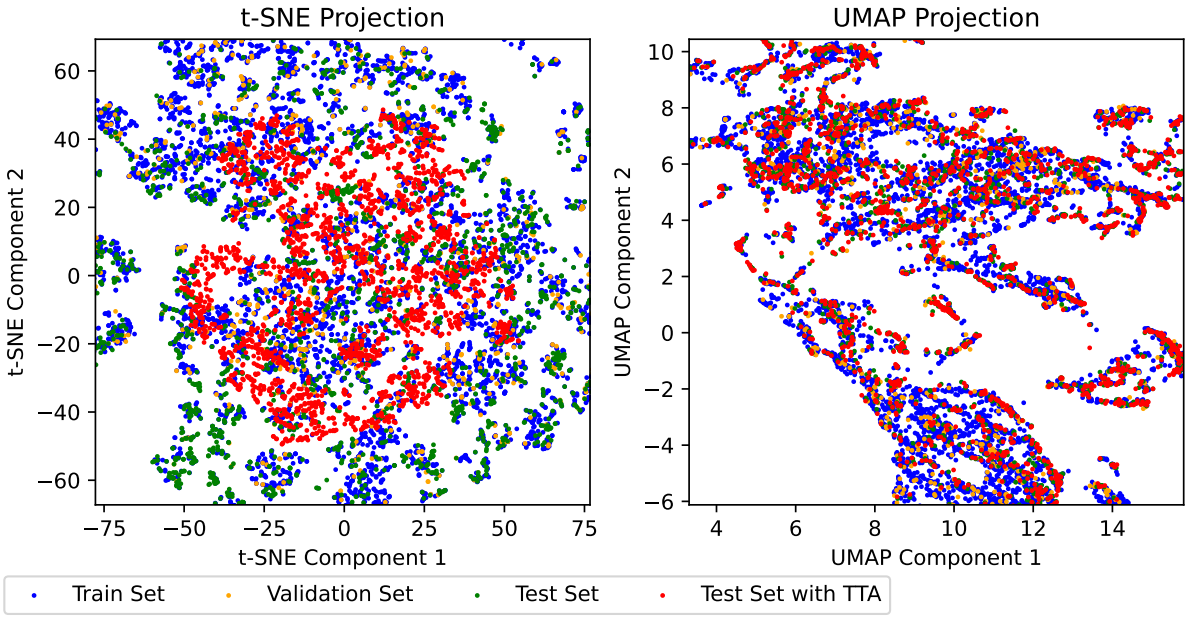

Figure 3: 2D projection of the embedding vectors with sets.

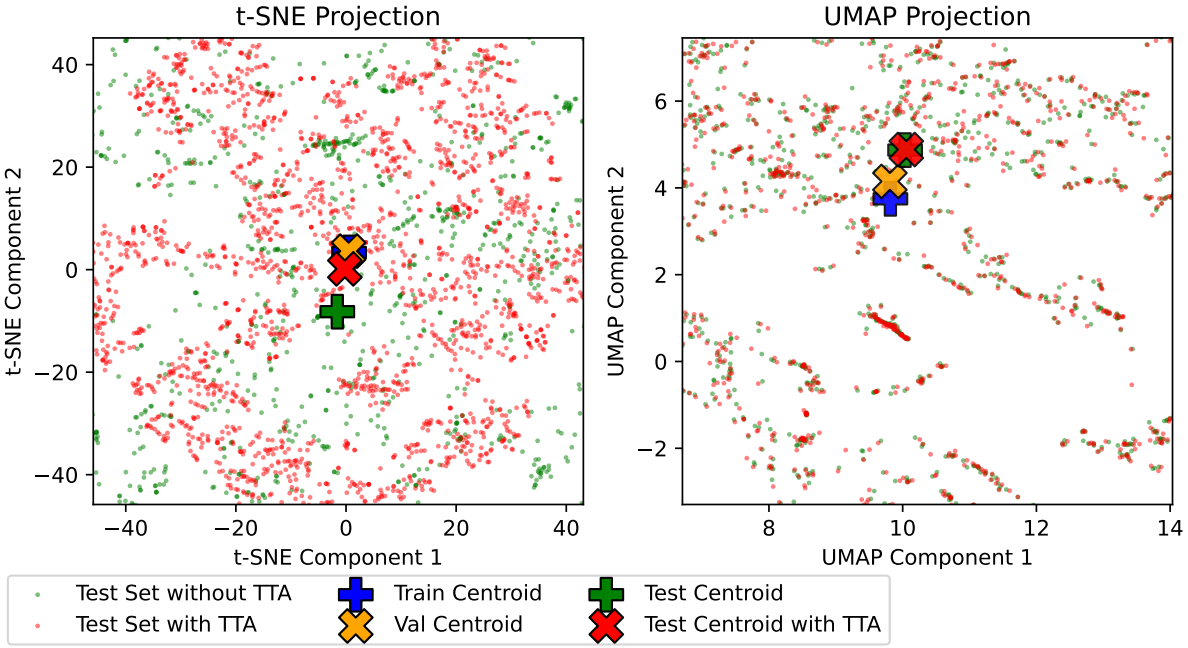

Figure 4: 2D projection of the embedding vectors with set centroids.

4 Conclusion

5 Conclusion

This project investigated the application of TTA to GNNs for molecular solubility prediction across different datasets. While TTA successfully shifted the test set embeddings towards the training distribution, as demonstrated through dimensionality reduction visualizations, this did not translate into improved predictive performance on the test set used here.

References

[1] Taoyong Cui, Chenyu Tang, Dongzhan Zhou, Yuqiang Li, Xingao Gong, Wanli Ouyang, Mao Su, and Shufei Zhang. Online test-time adaptation for better generalization of interatomic potentials to out-of-distribution data. *Nature Communications*, 16(1):1891, 2025.

[2] Jian Liang, Ran He, and Tieniu Tan. A comprehensive survey on test-time adaptation under distribution shifts. *International Journal of Computer Vision*, 133(1):31–64, 2025.

[3] P Llompart, C Minoletti, S Baybekov, Dragos Horvath, G Marcou, and A Varnek. Will we ever be able to accurately predict solubility? *Scientific Data*, 11(1):303, 2024.

[4] Murat Cihan Sorkun, Abhishek Khetan, and Süleyman Er. Aqsoldb, a curated reference set of aqueous solubility and 2d descriptors for a diverse set of compounds. *Scientific data*, 6(1):143, 2019.

[5] Yiqi Wang, Chaozhuo Li, Wei Jin, Rui Li, Jianan Zhao, Jiliang Tang, and Xing Xie. Test-time training for graph neural networks. *arXiv preprint arXiv:2210.08813*, 2022.