Graph Neural Network Experiments and Findings

Liu Xinran

**Introduction**

A few experiments were carried out to explore different methods of graph neural network. Below is a summary of explorations and findings using pytorch geometric to predict the chemical property of molecules. For more details of literature review, see this [document](https://docs.google.com/presentation/d/1Q4T2UH2JUMhlMQMS-7D2GwWutLiBJCc1z-oQc9eRS8Q/edit?usp=sharing).

**Dataset**

The QM9 dataset that was used in this experiment has been termed as the golden standard for Machine Learning (ML) predictions of various chemical properties. (Glavatskikh et al., 2019) It is a comprehensive dataset that provides geometric, energetic, electronic and thermodynamic properties for a subset of GDB-17 database, comprising 134 thousand stable organic molecules with up to nine heavy atoms. In the study carried out by Gilmer et al. (2017), QM9 was selected as the training and testing dataset and 13 regression targets were selected one by one to train and test their graph neural network models. In QM9, there are 19 regression targets in total and the dipole moment was chosen as our prediction target, the rest were ignored.

The node and edge information are summarized in the table 1.

|  |
| --- |
| Nodes information:  The first 5 node attributes in QM9 are five different atoms N,C,O,N,F (one hot)  Average size of a molecule graph: 18.03 nodes  Minimum number of nodes in one graph: 3 nodes  Maximum number of nodes in one graph: 29 nodes |
| Edge information:  The 4 edge attributes in QM9 are four different bonds (one hot), and the distance was computed and added as the fifth attribute.  Average edge Euclidean distance: 1.27 Angstrom  Average number of edges: 37.33 |

Table 1

**Models**

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Model name | Test loss before training | Test loss after training |
| Self-defined Message Passing | b128\_e100\_r001\_adam\_mse\_msgAdd\_No\_edgeAttr\_2fc | 13.1298 | 1.1826 |
| b128\_e100\_r001\_adam\_mse\_msgMean\_ poolMaxAggr\_2fc\_wandb | 12.2103 | 0.9688 |
| b128\_e135\_r001\_adam\_mse\_msgMean\_poolMean\_2fc | 11.4386 | 0.9127 |
| b128\_e135\_r001\_adam\_mse\_msgAdd\_set2set\_2fc | 10.2645 | 0.8732 |
| NNConv | b20\_e50\_r001\_adam\_mse\_nnconvAdd\_poolMeanAggr\_3fc | 10.3316 | **0.6963** |
| b20\_e100\_r001\_adam\_mse\_nnconvAdd\_poolMeanAggr\_2fc | 9.6425 | 0.7146 |
| b20\_e100\_r001\_adam\_mse\_nnconvMax\_poolMeanAggr\_2fc | 11.2977 | 0.7605 |
| b20\_e100\_r001\_adam\_mse\_nnconvMean\_poolMaxAggr\_2fc | 8.4049 | **0.6803** |
| b20\_e100\_r001\_sgd\_mse\_nnconvMean\_poolMeanAggr\_2fc | 8.4486 | 0.8811 |
| b20\_e100\_r005\_adam\_mse\_nnconvMean\_poolMeanAggr\_2fc | 9.3637 | 0.7975 |
| b50\_e100\_r001\_adam\_mse\_nnconvMean\_poolMaxAggr\_2fc\_wandb | 9.4620 | 0.7092 |
| GCNConv | adam\_mse\_gcnconv\_poolMaxAggr\_2fc\_wandb | 8.8354 | 0.7523 |
| GATConv | b128\_e150\_r001\_adam\_mse\_gatconv\_poolMaxAggr\_2fc\_wandb | 10.9331 | 0.8452 |
| b128\_e300\_r001\_adam\_mse\_gatconv\_poolMaxAggr\_2fc\_wandb | 10.1213 | **0.6733** |

Table 2

Note: for the naming of models, ‘b’ indicates batch size, ‘e’ indicates number of epochs, ‘r’ indicates learning rate (e.g. r001 means learning rate = 0.001), ‘poolXXAggr’ means what kind of aggregator was used for the pooling layer, ‘wandb’ suffix means it was watched and recorded by wandb, ‘2fc’ suffix means there were 2 fully connected layers in the structure. Three lowest test loss were marked in bolded form. They are b128\_e300\_adam\_mse\_gatconv\_poolMaxAggr\_2fc

**Results**

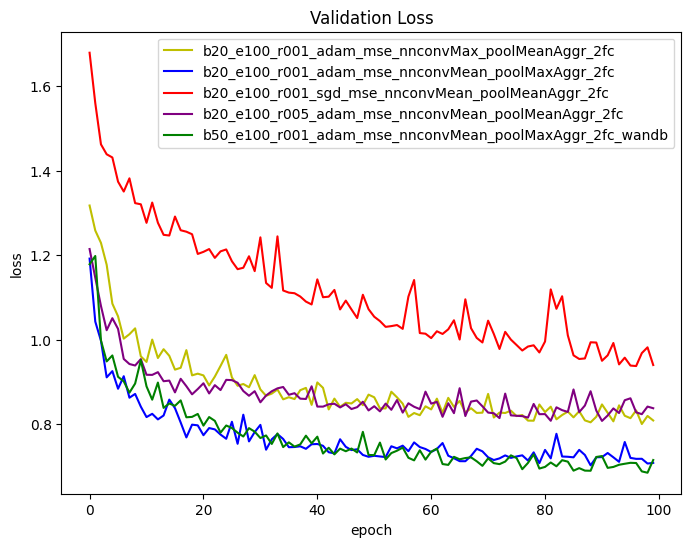
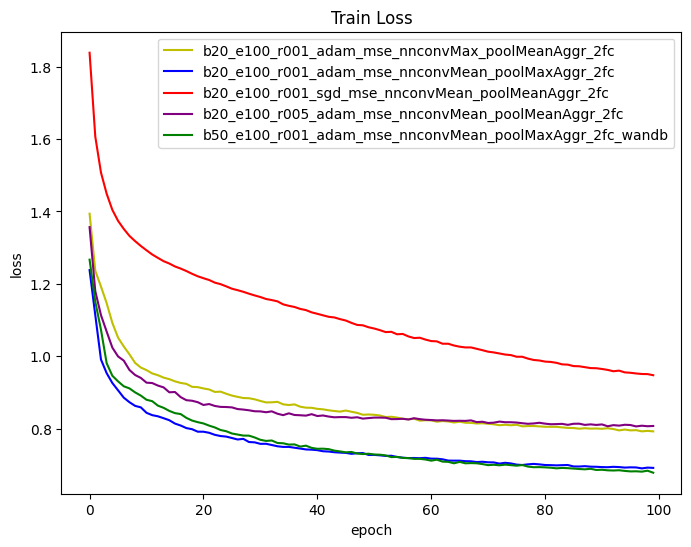


Figure 1 Figure 2

The training loss values and validation loss values of five NNConv models are plotted in the figure 1 and 2.

The following table explains the different GNN layers used by the models and these layers are all supported by Pytorch Geometric. (Fey et al., 2019)

|  |  |
| --- | --- |
| Self-defined Message Passing | Used the base class of message passing, with a chosen aggregation methods between ‘add’, ‘mean’ and ‘max’ and self-defined message function and update function.  It is noticed that the model with edge attributes and node attributes as input tend to have better results (lower loss) compared to the ones with only node attributes. |
| [NNConv](https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.nn.conv.NNConv.html) | Experimented with three aggregation methods ‘add’, ‘mean’ and ‘max’ in NNConv layer.  The results showed that the use Max Aggregator in the model helps to boost the model performance in general with the lowest loss values in both training loss and validation loss. |
| [GCNConv](https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.nn.conv.GCNConv.html#torch_geometric.nn.conv.GCNConv) | Used Weights & Biases Sweeps to automate hyperparameter search with random search method and other pre-defined hyperparameter values. The more detailed report including a parallel coordinates plot, a parameter importance plot, and a scatter plot, supported by Weights & Biases for model ‘adam\_mse\_gcnconv\_poolMaxAggr\_2fc’ can be found [here](https://api.wandb.ai/links/ntuwb/fg06ekm9). |
| [GATConv](https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.nn.conv.GATConv.html#torch_geometric.nn.conv.GATConv) | Experimented with larger size of epochs but failed to achieve the ‘U’ shape of validation loss. See figure 3 below. Even when the epoch size is 300, the validation loss is still dropping. Therefore, more improvement can be done by increasing the epoch size further. |

Table 3

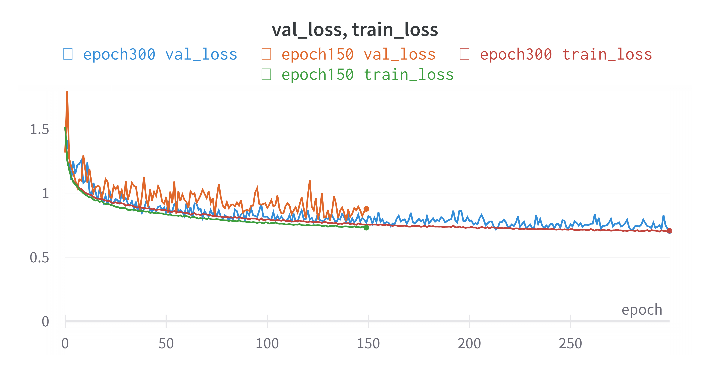


Figure 3

Adam was chosen as the default optimizer and SGD was tested in one of the models. The red lines in figure 1, 2 represent the model using SGD optimizer, and they show a higher loss value in both training and validation loss values compared to the other models using Adam. One of the differences between Adam and SGD is that Adam tends to converge faster, while SGD often converges to more optimal solutions. Here when the training ends, SGD is still converging, not reaching the optimal solution. Therefore, the number of training epochs should be further increased to observe the convergence of the results that uses SGD optimiser.

**Conclusion and Reflection**

Several experiments have been carried out to discover the different effects of number of epochs, batch size, learning rate, optimiser, message passing layer, pooling layer, the number and dimension of fully connected layers. The average loss value per epoch from both training and validation are recorded and selectively plotted. Weights & Biases was used as a documentation and visualisation tool from the second half of the project, and several reports are generated automatically including interactive plots and dashboards.

There are few improvements to be made: first, the number of epochs should be further increased to find the optimal solution, because the validation loss is still dropping when the training ended, meaning the model is not well-trained. Second, the documentation of the whole project lacks consistency because python logger and Weights & Biases were introduced at the second half of the project, such that some models have certain results, but some don’t, some are stored in local while some are stored in Weights & Biases workplace. Third, the time of training was not recorded for the first half of the project but recorded by the second half, so a comparison cannot be made across different models.

**References**

Fey, M., & Lenssen, J. E. (2019). Fast graph representation learning with PyTorch Geometric. arXiv preprint arXiv:1903.02428.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017, July). Neural message passing for quantum chemistry. In International conference on machine learning (pp. 1263-1272). PMLR.

Glavatskikh, M., Leguy, J., Hunault, G. et al. (2019). Dataset’s chemical diversity limits the generalizability of machine learning predictions. J Cheminform 11, 69. <https://doi.org/10.1186/s13321-019-0391-2>