Graph Neural Network to predict the physical chemical property from structural information in QM9 dataset

Jihun Jeung

jihun@gm.gist.ac.kr

Life Data Mining Lab, School of Life Science, GIST

2021.12.19

1. Project statement

Graph Neural Network (GNN) is a neural model that leverages the structure and properties of graphs. Compared to convolution network (CNN) which requires the rigid grid, GNN can be applied to flexible graph structure. A typical example of GNN application is molecular structure. A molecule can be projected as a graph, whose nodes are their atom and edges are their bonds. AlphaFold2 could increase its performance by adopting GNN in *Evoformer* block.[1] Another promising result is antibiotic discovery in 2020. A directed message passing neural network that is trained with molecular properties from publicly opened database predicts Halicin as an effective antibiotic.[2] GNN is a promising model in cheminformatics and bioinformatics.

Molecular properties such as toxicity in biological systems are mainly determined by 3-dimensional conformation. Free energy is the key quantity to describe molecular dynamics and conformation. Although calculating free energy is always a crucial issue in chemistry and biology, molecular dynamics (MD) simulation, which calculates the lowest free energy conformation and predicts its 3-dimensional structure, requires heavy computational cost. Can GNN become a breakthrough for structural chemistry and biology?

The QM9 dataset from the "MoleculeNet: A Benchmark for Molecular Machine Learning" paper contains about 130,000 molecules with 19 targets, such as dipole moment, HOMO, LUMO, internal energy, free energy, and so on. Each molecule includes complete 3-dimensional spatial information for the single low energy conformation. [3]

2. Methods

2-1. Dataset

I utilize QM9 dataset as mentioned earlier. Each graph represents a molecule of interests. Nodes correspond to atoms in a molecule and edges to (undirected) bonds. Each atom was embedded into

---

[1] Jumper, J., Evans, R., Pritzel, A. *et al.* Highly accurate protein structure prediction with AlphaFold. *Nature* **596,** 583–589 (2021). https://doi.org/10.1038/s41586-021-03819-2

[2] https://www.cell.com/cell/pdf/S0092-8674(20)30102-1.pdf

[3] https://pubs.rsc.org/en/content/articlelanding/2018/sc/c7sc02664a

node features of 11-dimensional vector: 5-dimensional one hot encoding (hydrogen, carbon, nitrogen, oxygen, fluorine) and 5-dimensional chemical properties (atomic number, aromatic, sp, sp2, sp3, number of hydrogen atoms). Each bond was embedded into edge features of 4-dimensional vector: 4-dimensional chemical bond type (single bond, double bond, triple bond, aromatic bond)

2-2. Methods

PyTorch Geometry (PyG) [4] is a library built upon PyTorch to train GNN for a wide range of applications. The dataset would be split into training set (80%), validation set (10%), test set (10%). Adam or SGD will be used as an optimizer (the optimizer with best performance was selected). Mean squared error (MSELOSS) was used as loss function. The model will make graph-level prediction using all the node embedding in the graph.

Dataset --> node embedding --> [GNN layer] X N --> global pooling --> graph-level prediction

The type of GNN layer and the number of GNN layers were selected among GCNConv, SAGEConv, and so on in terms of performance. To reduce overfitting, dropout and skip connection was considered. For graph-level prediction, global sum pooling that has the highest representation compared to global mean pooling and global max pooling, was used.

3. Experimental results

To find the model architecture that reflects my problem, I tried to train two models. One (referred as GCN_NET) is a model with GCNConv[5] and the other (referred as GAT_Net) is GATConv[6]. Note that GNN with more than 7 graph convolution layers were not properly trained[7]. Considering this fact, I designed the model with 5 graph convolution layers in order to capture the complex molecular structure as much as possible. GCNConv has 80 hidden channels. GCN_NET is composed like following:

Graph embedding – GCNconv – GCNconv – GCNconv – GCNconv – GCNconv – global add pooling - fully connected layer (0.1 drop-out)– fully connected layer

GATConv contains the multi-head attention mechanism and it was expected to reflect the molecular local environment. In my implementation, GATConv has 150 hidden channels and 15 multi-head attention with drop-out of the ratio 0.2. GAT_NET is built like following:

Graph embedding – fully connected layer - GATConv layer - GATConv layer - GATConv layer - global add pooling – Fully connected layer (0.2 drop-out) – fully connected layer

Out of two models, the model with GCN_NET showed the greater performance in terms of validation mean square error (MSE) and test MSE (Table1, Figure 1). During the training of GAT, loss value was

[4] https://github.com/pyg-team/pytorch_geometric
[5] https://arxiv.org/abs/1609.02907
[6] https://arxiv.org/abs/1710.10903
[7] https://arxiv.org/abs/1609.02907

unstable and jump up frequently (Figure 2). To further investigate the advanced setting for GAT_NET, skip mechanism and ensemble were added and compared.

Method	Epoch	Loss	Val MAE	Test MAE
GCN	300	171919.1016684	708.8797153	446.8069962
GCN	400	172970.8735012	650.3755138	421.9882445
GCN_skip	300	172503.9681173	219.6515542	167.3754175
GCN_skip_ensemble	189 (stopped)	59296.6363744	355.8193538	323.9060308
GAT	3278	211696.6485454	3092.9200011	2017.4051895

Table 1. The comparison between different GNN models for QM9 dataset.

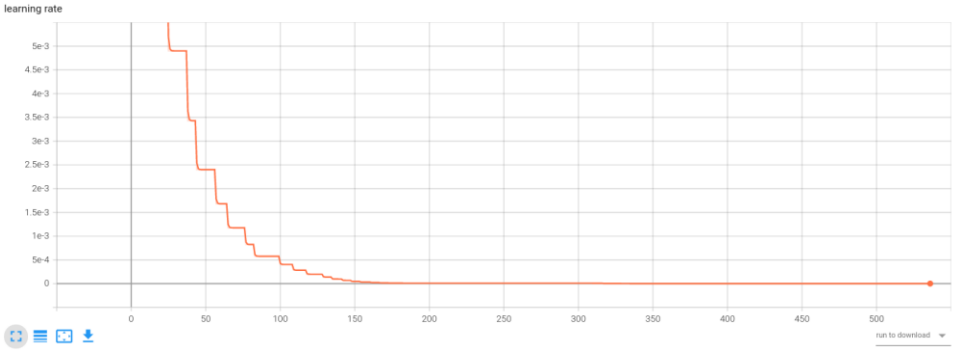

Figure 1(a). The learning rate during training GCN_Net

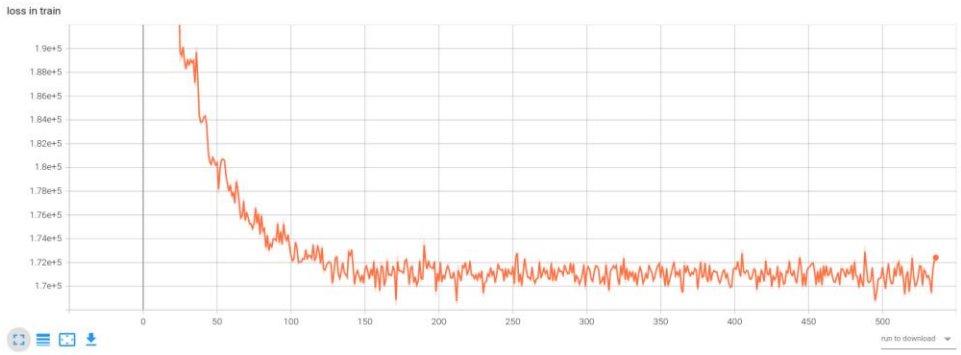

Figure 1(b). The training loss during training GCN_Net

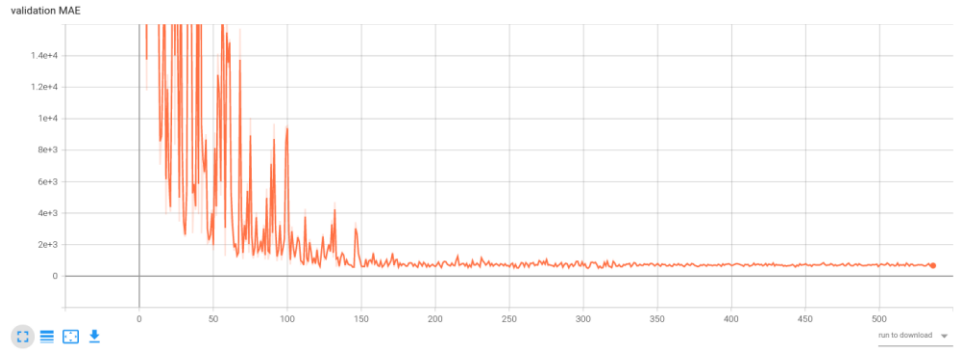

Figure 1(c). The validation mean square error (MSE) during training GCN_Net

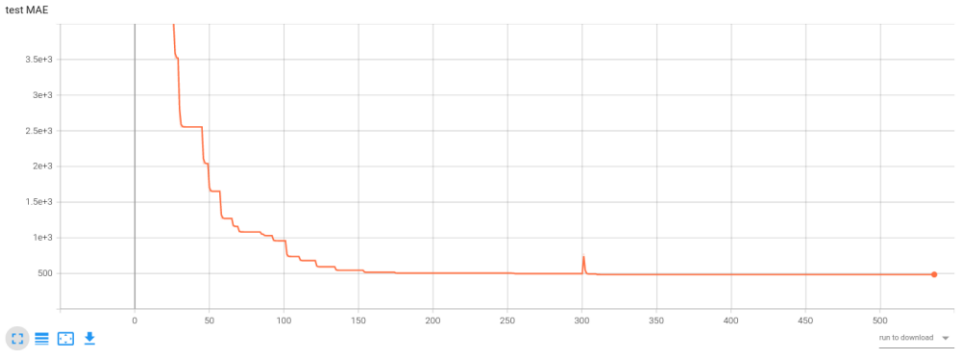

Figure 1(d). The test mean square error (MSE) during training GCN_Net

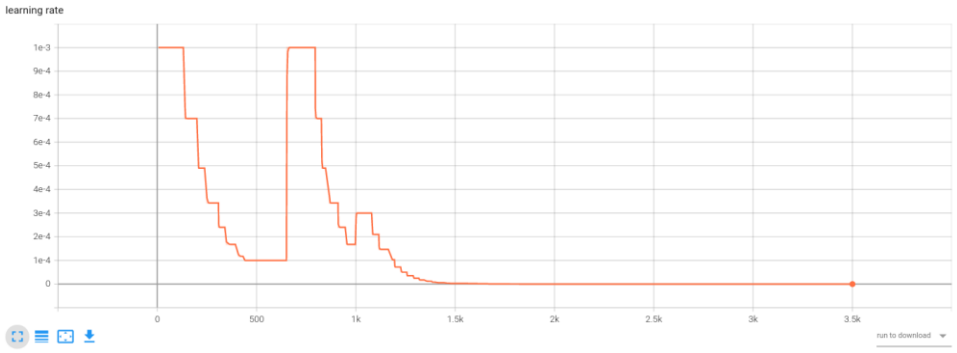

Figure 2(a). The learning rate during training GAT_Net

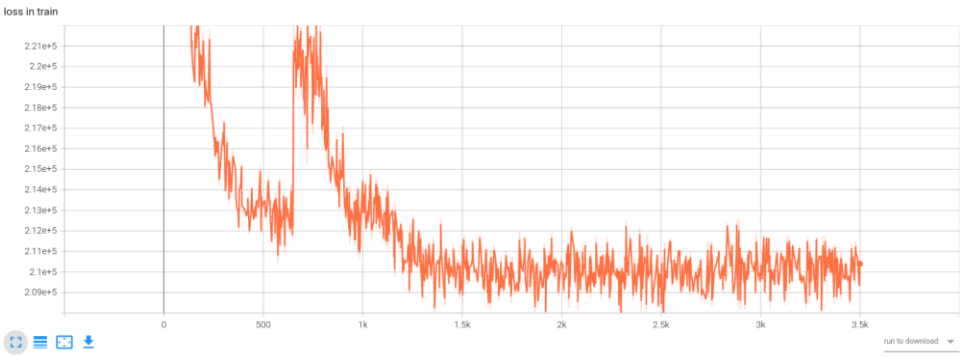

Figure 2(b). The training loss during training GAT_Net

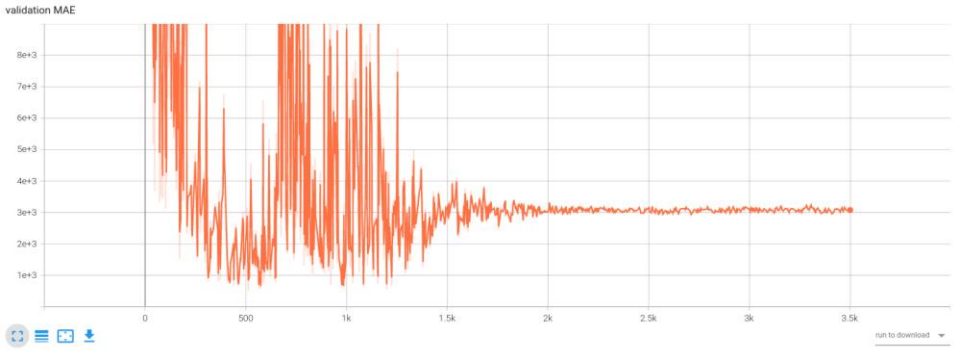

Figure 2(c). The validation mean square error (MSE) during training GAT_Net

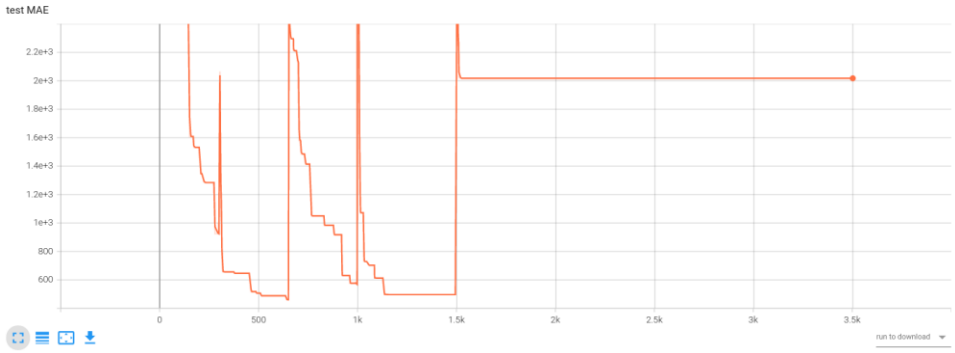

Figure 2(d). The test mean square error (MSE) during training GAT_Net

To investigate which learning schedule is effective to train graph neural network, I compared the two types of learning schedule on same GNN model. To control the training environment, I made all training settings equal except the type of learning schedule. I used the model with the following architecture (check *GAT_overfitting_Net* on the submitted ipynb note) :

Graph embedding - Fully connected layer – GAT – GAT – fully connected layer

Both were trained with the optimizer Adam (weight decay = 5e-4). The only difference was the type of scheduler. One (**type 1, Figure 3**) is constant over time (Class torch.optim.lr_scheduler.ReduceLROnPlateau, mode='min', factor=0.7, patience=10, min_lr=0.0025) and the other (**type 2, Figure 4**) oscillates over time (Class torch.optim.lr_scheduler.CyclicLR, base_lr=1e-5, max_lr=1e-3, cycle_momentum=False, step_size_up=2000).

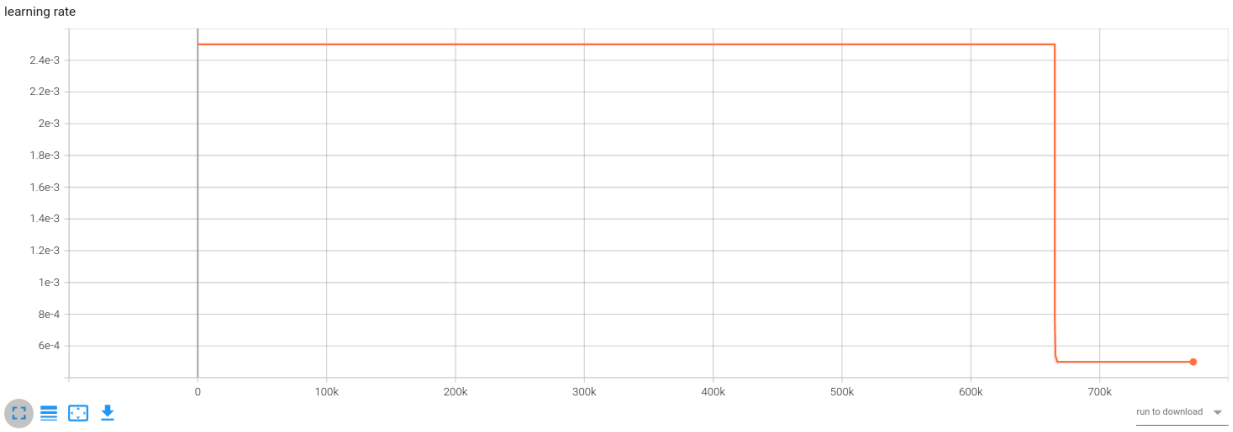

Figure 3(a). Learning rate of type 1 scheduler. Note that learning rate was reset on around epoch 650k.

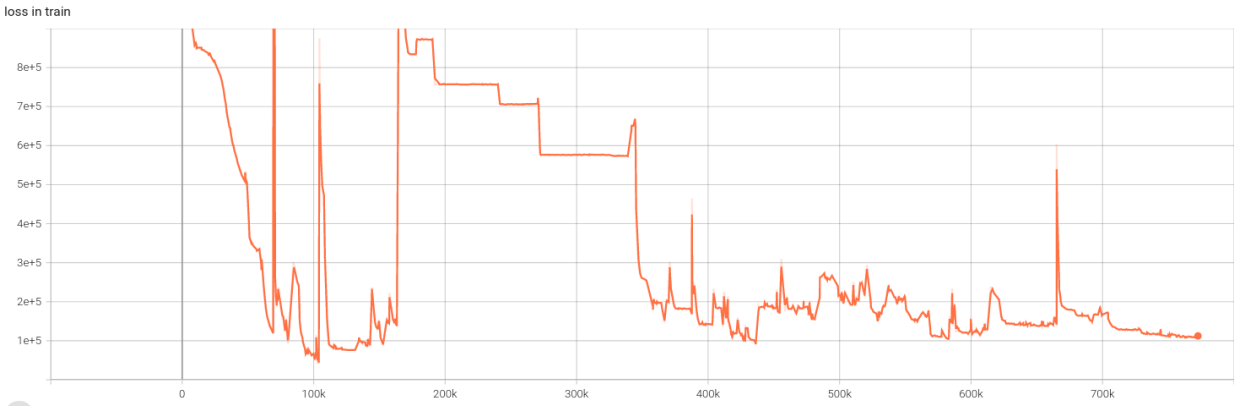

Figure 3(b). Loss of type 1 scheduler

Type 1 learning scheduler has a constant learning rate over time. The loss reached the local minimum around epoch 100k. After 150k epoch, the loss increases to search another local minimum and loss reached another local minimum around 400k epoch. Even though the loss has some fluctuation owing to another local minima, my result showed that the model was trained enough around 100k epoch with type 1 schedule. Note that the learning rate was reset to decrease from around 650k epoch in order to check whether loss can reduce more (min_lr=0.0005). The overall loss value did not decrease significantly.

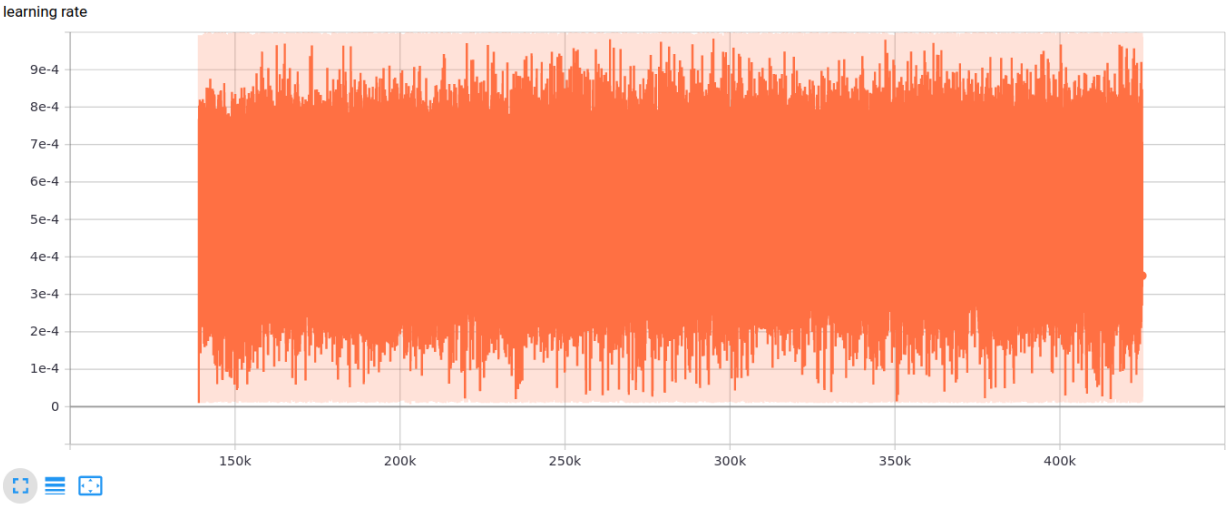

Figure 4(a). Learning rate of type 2 scheduler.

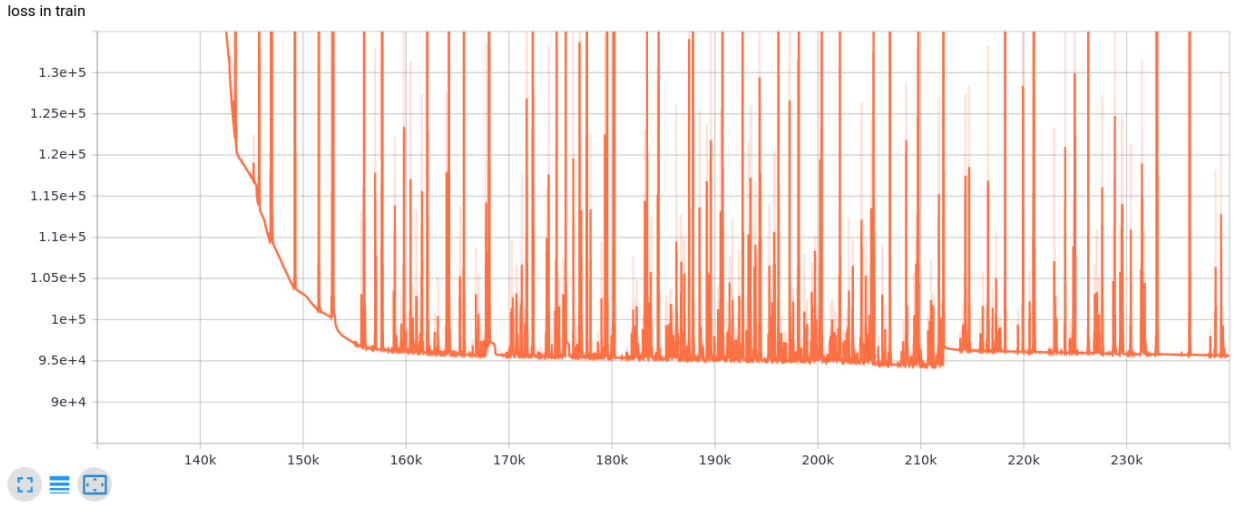

Figure 4(b). Loss of type 2 scheduler.

Type 2 learning scheduler oscillates learning rate over time. In my experiment, the maximum learning rate is 0.001 and the minimum learning rate is 0.00001. Type 2 learning scheduler has effective characteristics of global search on optimization. When a learning rate reaches maximum, an optimization algorithm conducts the global search (exploration stage). In contrast, when a learning rate reaches minimum, an optimization algorithm conducts the local search (refinement stage)[8]. Because of its oscillation in learning rate, loss also oscillates. The overall loss reaches a plateau on 160k epoch, and its overall loss becomes constant around 95000.

[8] I borrowed the term 'exploration (global) stage' and 'refinement (local) stage' from numerical optimization terminology to depict the properties of type 2 learning scheduler. These two terminology areused to explain the method of stochastic global optimization such as genetic algorithm (GA) and simulated annealing (SA)

There is one perfect learning scheduler for all models. To select a proper scheduler for a model, we need to consider what a problem is and what resources are available. My result shows that if a resource is limited, then training GNN with a constant learning scheduler (type 1) is good enough, but it does not guarantee the convergence to local minimum. A model has the probability that the loss is out of local minimum (check the plateau interval between 200k epoch and 350k epoch on loss of type 1 scheduler). If available computing resources are enough, then training with an oscillating learning scheduler (type 2) is a good choice, because it guarantees stable convergence to the local minimum.

4. Conclusion

Deep learning is a promising framework that makes a task that requires too expensive cost to conduct without deep learning possible. Graph neural network (GNN) is one of the promising mainstreams of deep learning. The strength of GNN is that GNN can train with arbitrary size and complex topological structure (i.e. no spatial locality). The key to applying GNN in a specific domain including cheminformatics and bioinformatics is to understand the architecture of graph convolution layers and design the GNN model architecture in proper way. However, selecting a best model architecture is not easy and tedious. My implementation shows that the model with the five simplest graph convolution layer GCN is enough to train the structural information in QM9 dataset. Also, I suggest a way to select a proper learning scheduler for GNN. My application of GNN on QM9 dataset may give some insight to select the model architecture in cheminformatic domain.