Practice 06

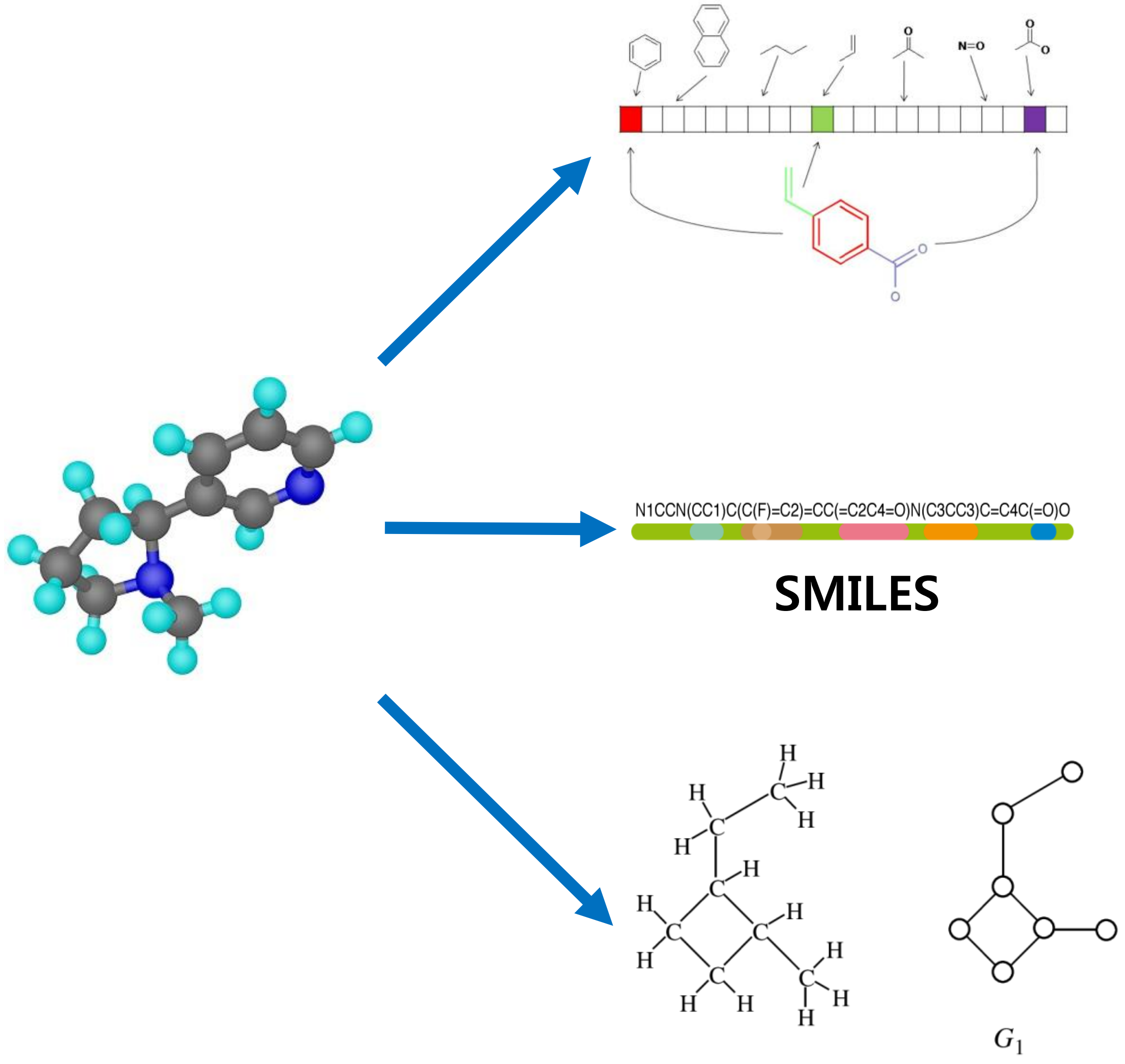
Molecular graph and GNN

Seongok Ryu

Department of Chemistry, KAIST

Contents

Molecular Graph



Practice 03, 04

SMILES

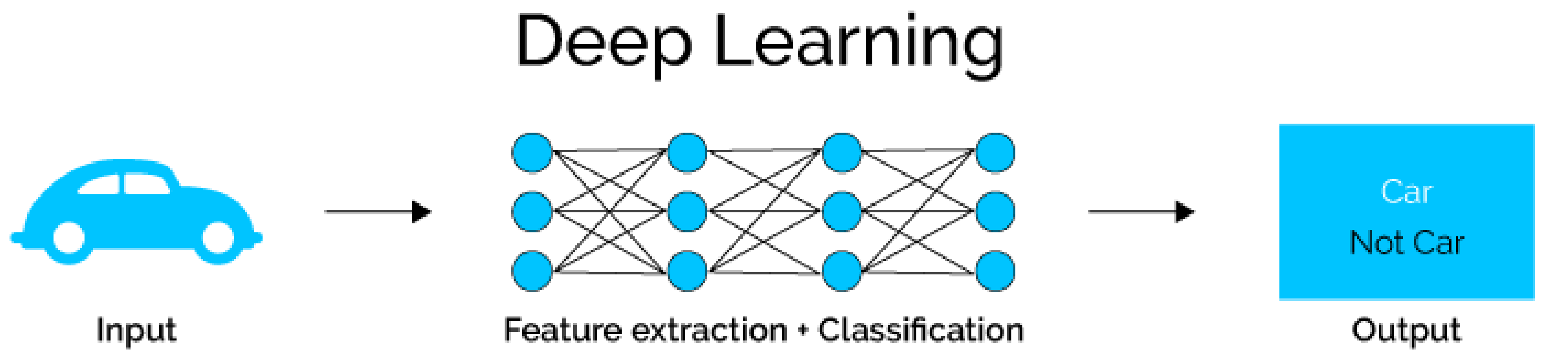
Practice 05, 07

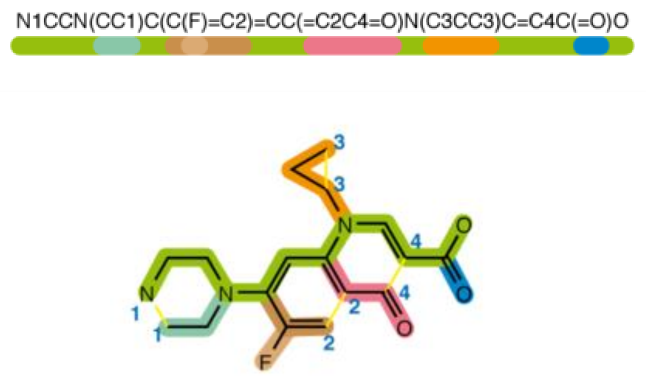
Molecular graph

Practice 06, 08

Molecular Graph

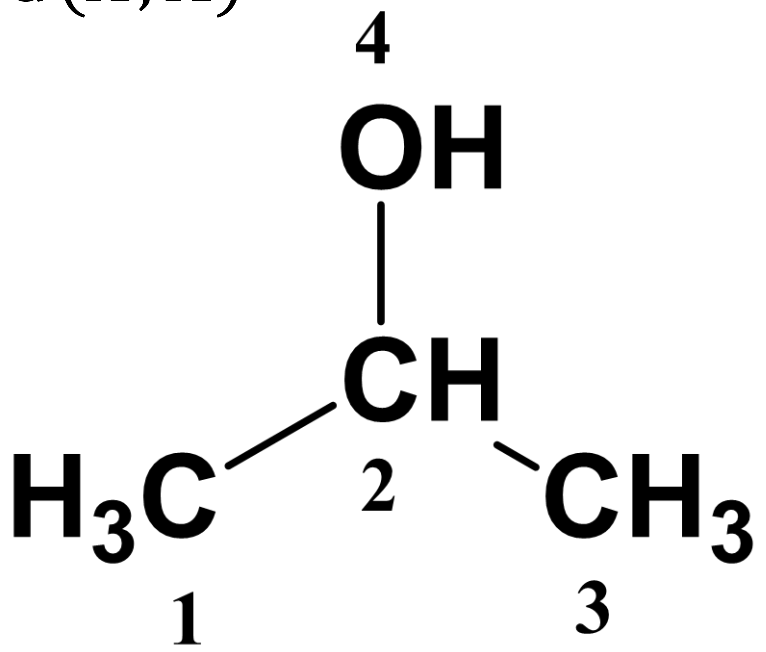
- How about use raw inputs rather than featurized inputs?
- SMILES and molecular graph can describe the molecular structure.
- Let machines to learn both featurization and prediction by itself. – It is the heart of deep learning!

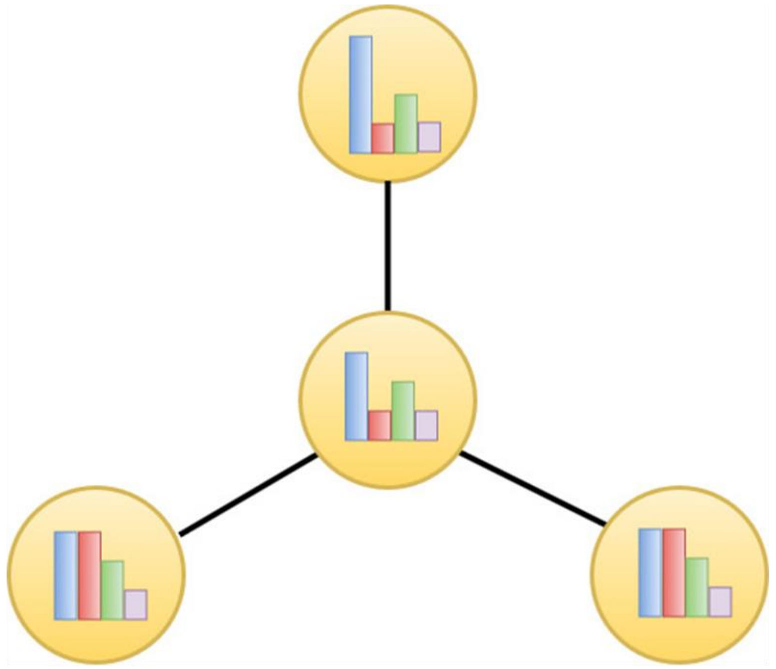




Molecular Graph

$Graph = G(X, A)$

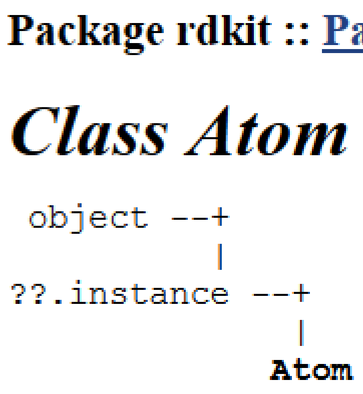




$$X_1 = \begin{bmatrix} 6 \\ 3 \\ 4 \\ 0 \end{bmatrix} \quad \cdots \quad X_4 = \begin{array}{cl} \begin{bmatrix} 8 \\ 1 \\ 4 \\ 0 \end{bmatrix} & \begin{array}{l} \textbf{Atom type} \\ \textbf{\# of Hs.} \\ \textbf{\# Valence} \\ \textbf{Aromaticity} \end{array} \end{array} \quad A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

Molecular Graph

You can easily obtain the molecular graph using RDKit

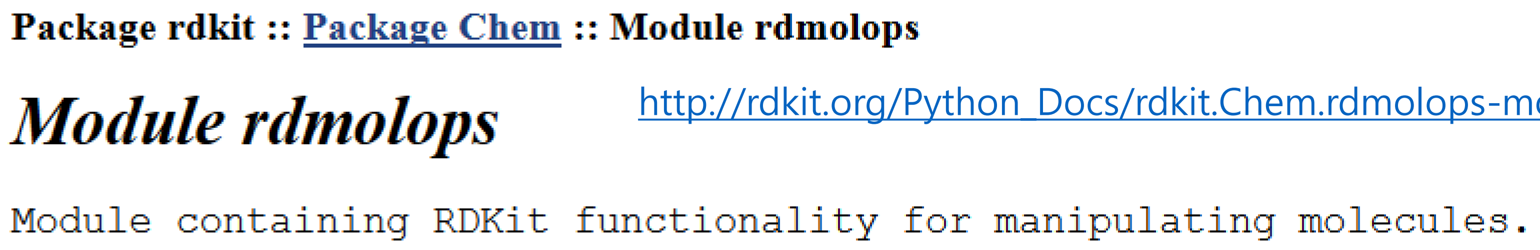


Package rdkit :: Package Chem :: Module rdchem :: Class Atom

Class Atom

```
 object --+
          |
??.instance --+
              |
             Atom
```

http://www.rdkit.org/Python_Docs/rdkit.Chem.rdchem.Atom-class.html



Package rdkit :: Package Chem :: Module rdmolops

Module rdmolops

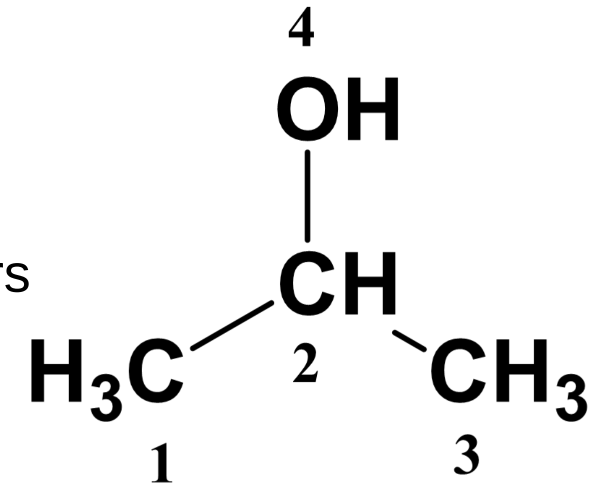
```
Module containing RDKit functionality for manipulating molecules.
```

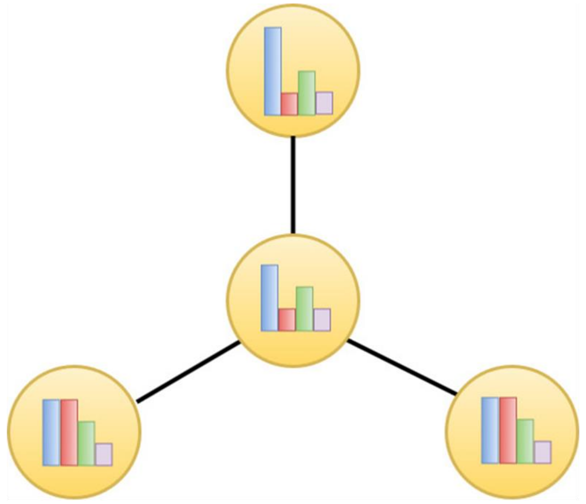
http://rdkit.org/Python_Docs/rdkit.Chem.rdmolops-module.html

Molecular Graph

We need following atom descriptors.

- Atom type
  atom.GetSymbol()
- The number of directly-bonded neighbors
  atom.GetDegree()
- Number of hydrogens
  atom.GetTotalNumHs()
- Number of valence
  atom.GetImplicitValence()
- Aromaticity indicator
  atom.GetIsAromatic()





$$X_1 = \begin{bmatrix} 6 \\ 3 \\ 4 \\ 0 \end{bmatrix} \quad \cdots \quad X_4 = \begin{bmatrix} 8 \\ 1 \\ 4 \\ 0 \end{bmatrix} \begin{array}{l} \textbf{Atom type} \\ \textbf{\# of Hs.} \\ \textbf{\# Valence} \\ \textbf{Aromaticity} \end{array}$$

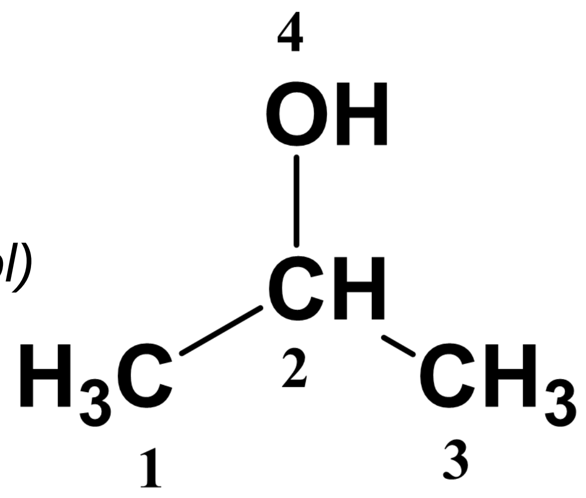
$$A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

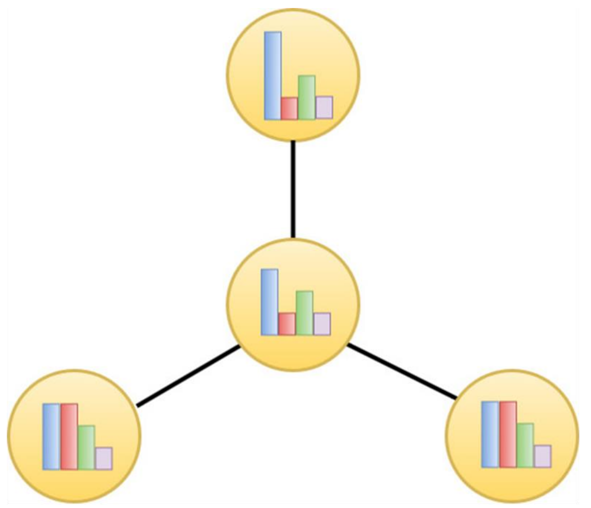
Search documents of each descriptor and understand how the functions work.

Molecular Graph

We need the adjacency matrix of a molecule.

- The adjacency matrix represents the connectivity between the atom pairs in a molecule. *Chem.rdmolops.GetAdjacencyMatrix(mol)*





$$X_1 = \begin{bmatrix} 6 \\ 3 \\ 4 \\ 0 \end{bmatrix} \quad \cdots \quad X_4 = \begin{bmatrix} 8 \\ 1 \\ 4 \\ 0 \end{bmatrix} \begin{array}{l} \textbf{Atom type} \\ \textbf{\# of Hs.} \\ \textbf{\# Valence} \\ \textbf{Aromaticity} \end{array}$$

$$\boxed{A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}}$$

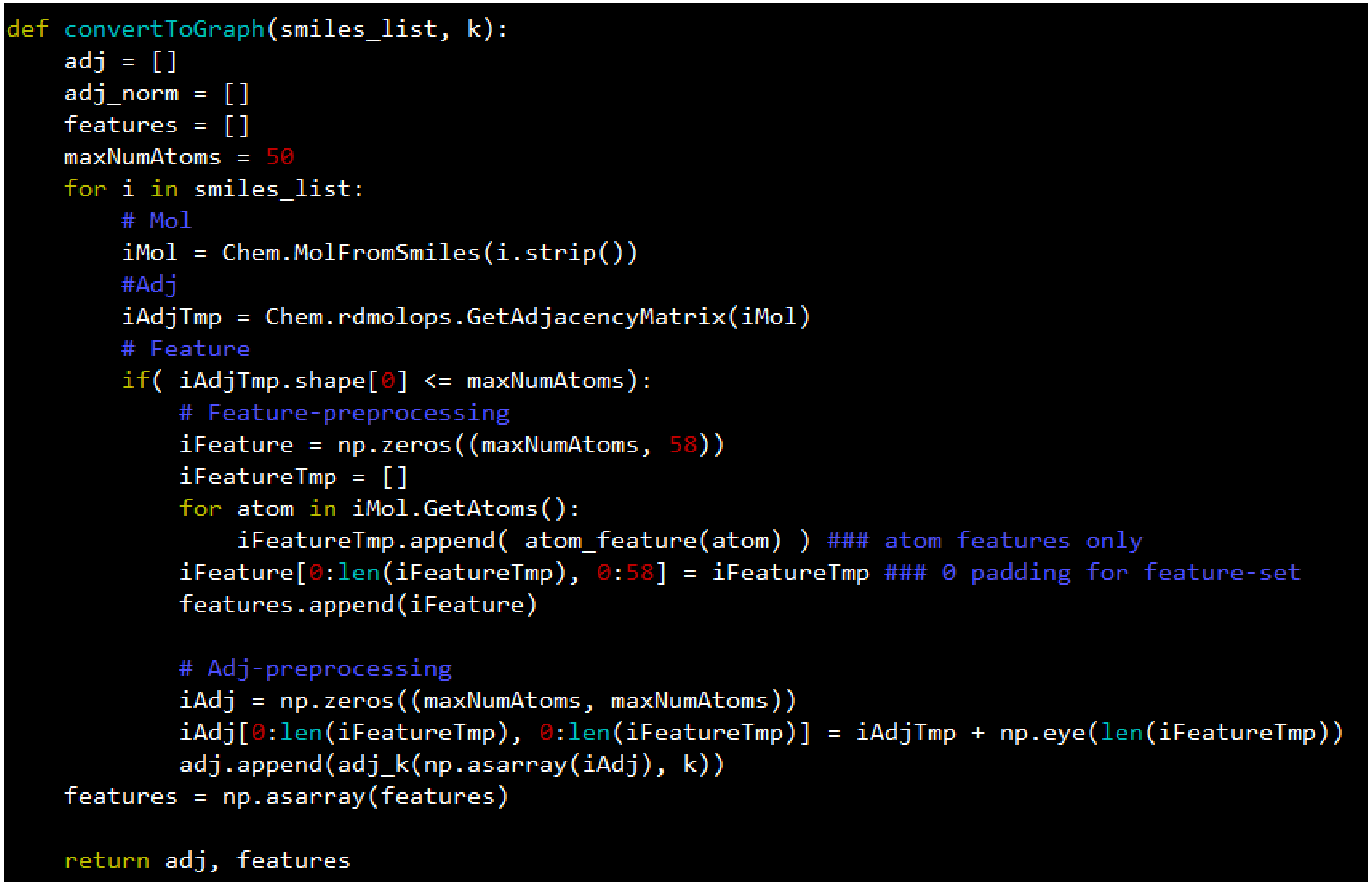
Question)

1. Do we have to differentiate bond types? e.g. single/double/aromatic/…

2. Do we need a distance matrix instead of the adjacency matrix?

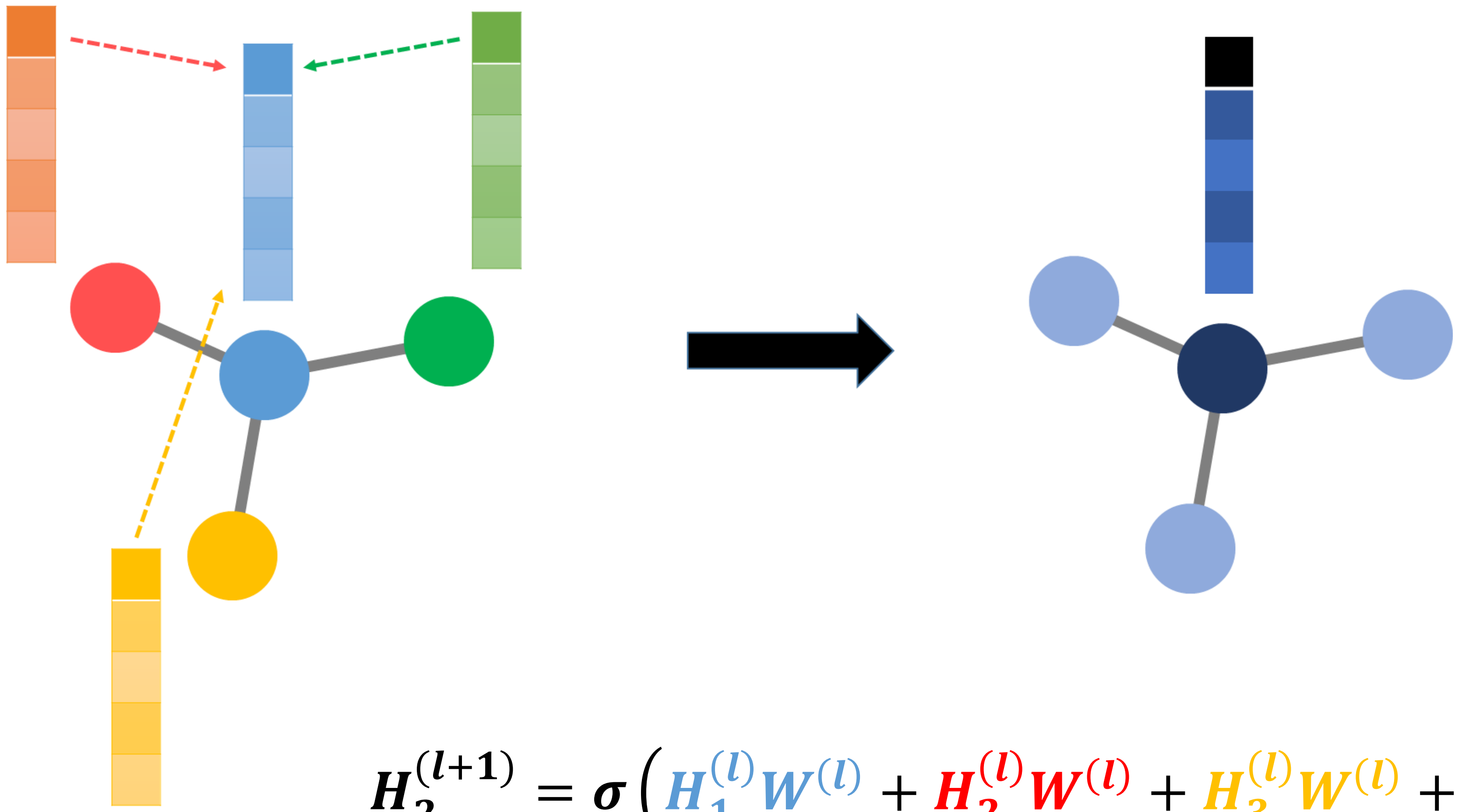
Molecular Graph

The script is implemented in utils.py, but you must know the details.



```python
def convertToGraph(smiles_list, k):
    adj = []
    adj_norm = []
    features = []
    maxNumAtoms = 50
    for i in smiles_list:
        # Mol
        iMol = Chem.MolFromSmiles(i.strip())
        #Adj
        iAdjTmp = Chem.rdmolops.GetAdjacencyMatrix(iMol)
        # Feature
        if( iAdjTmp.shape[0] <= maxNumAtoms):
            # Feature-preprocessing
            iFeature = np.zeros((maxNumAtoms, 58))
            iFeatureTmp = []
            for atom in iMol.GetAtoms():
                iFeatureTmp.append( atom_feature(atom) ) ### atom features only
            iFeature[0:len(iFeatureTmp), 0:58] = iFeatureTmp ### 0 padding for feature-set
            features.append(iFeature)

            # Adj-preprocessing
            iAdj = np.zeros((maxNumAtoms, maxNumAtoms))
            iAdj[0:len(iFeatureTmp), 0:len(iFeatureTmp)] = iAdjTmp + np.eye(len(iFeatureTmp))
            adj.append(adj_k(np.asarray(iAdj), k))
    features = np.asarray(features)

    return adj, features
```

Graph Convolutional Network (GCN)

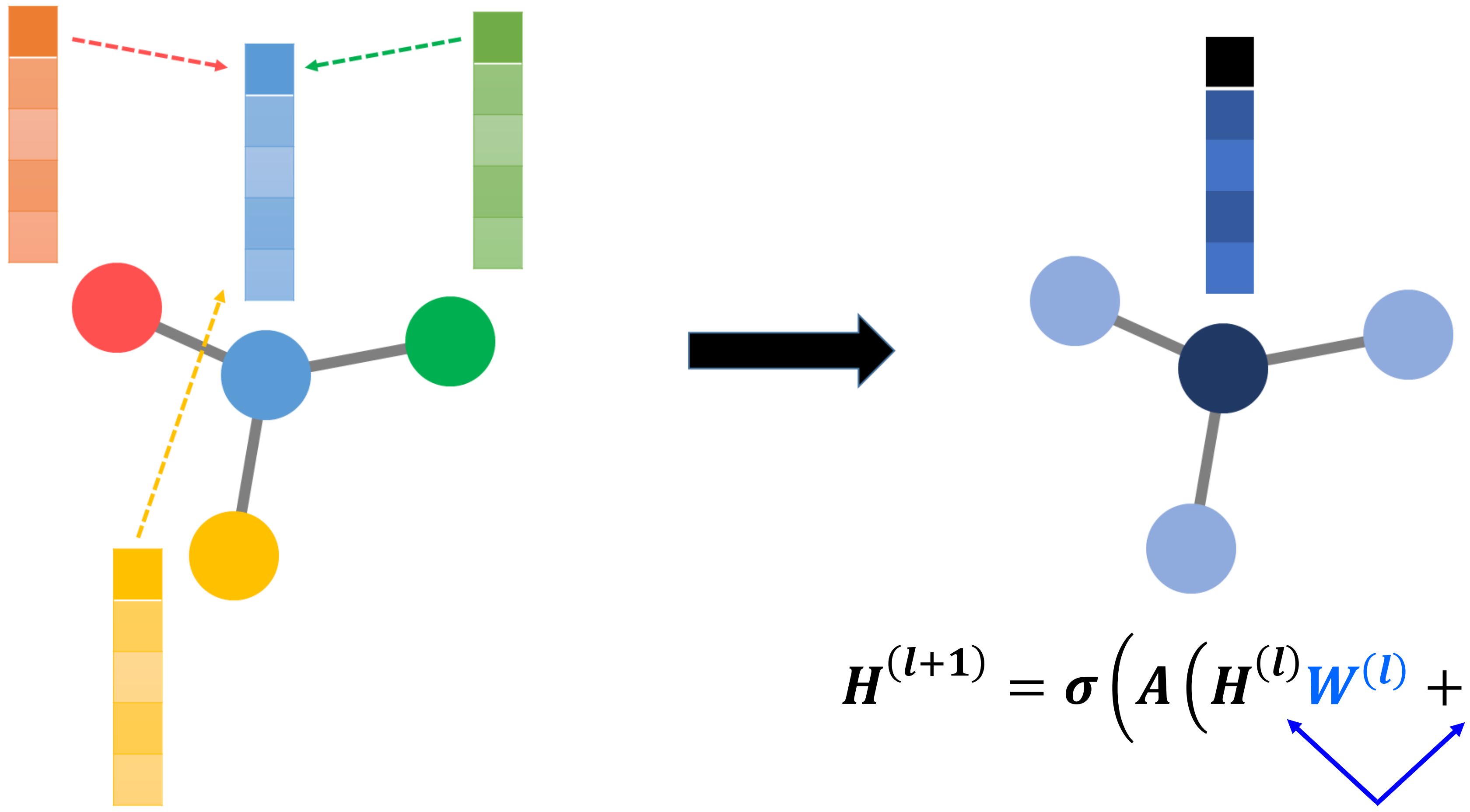


$$H_2^{(l+1)} = \sigma\left(H_1^{(l)}W^{(l)} + H_2^{(l)}W^{(l)} + H_3^{(l)}W^{(l)} + H_4^{(l)}W^{(l)} + b^{(l)}\right)$$

$$\Rightarrow \quad H_i^{(l+1)} = \sigma\left(\sum_{j \in N(i)} H_j^{(l)}W^{(l)} + b^{(l)}\right)$$

KAIST

Graph Convolutional Network (GCN)



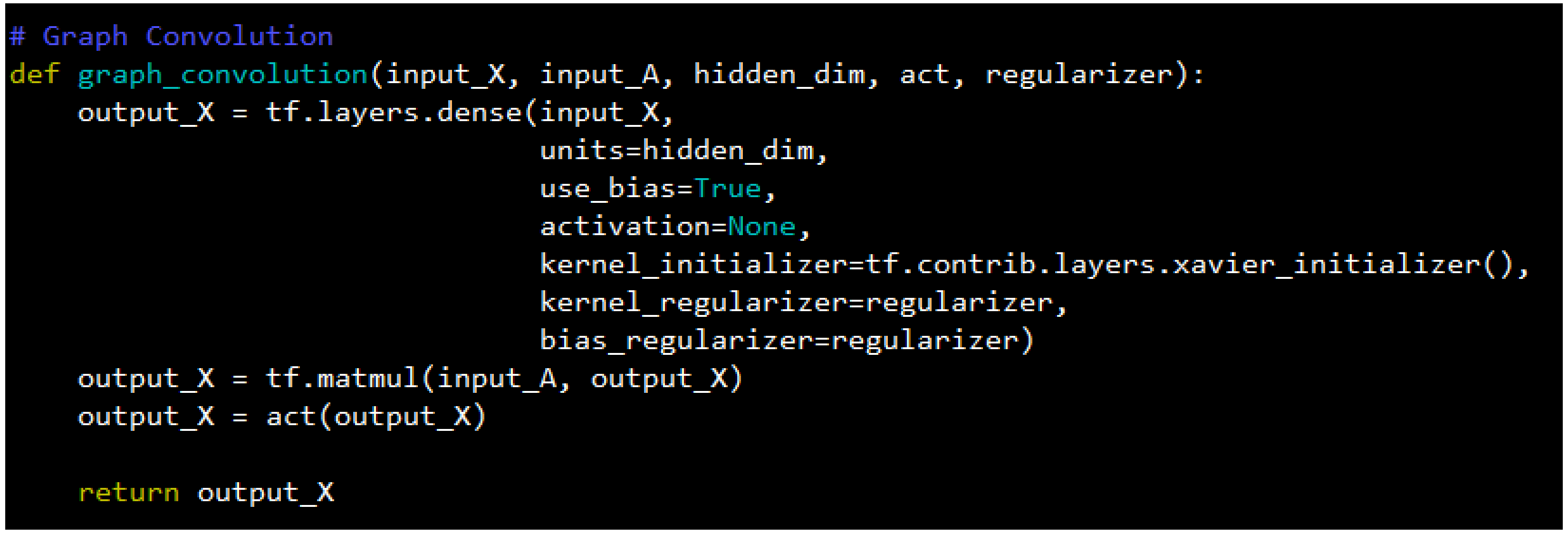
$$H^{(l+1)} = \sigma\left(A\left(H^{(l)}W^{(l)} + b^{(l)}\right)\right)$$

learnable parameters are shared

Sharing weights for all nodes in graph,
but nodes are differently updated by reflecting individual node features, $H_j^{(l)}$

KAIST

Graph Convolutional Network (GCN)



```
# Graph Convolution
def graph_convolution(input_X, input_A, hidden_dim, act, regularizer):
    output_X = tf.layers.dense(input_X,
                               units=hidden_dim,
                               use_bias=True,
                               activation=None,
                               kernel_initializer=tf.contrib.layers.xavier_initializer(),
                               kernel_regularizer=regularizer,
                               bias_regularizer=regularizer)
    output_X = tf.matmul(input_A, output_X)
    output_X = act(output_X)

    return output_X
```

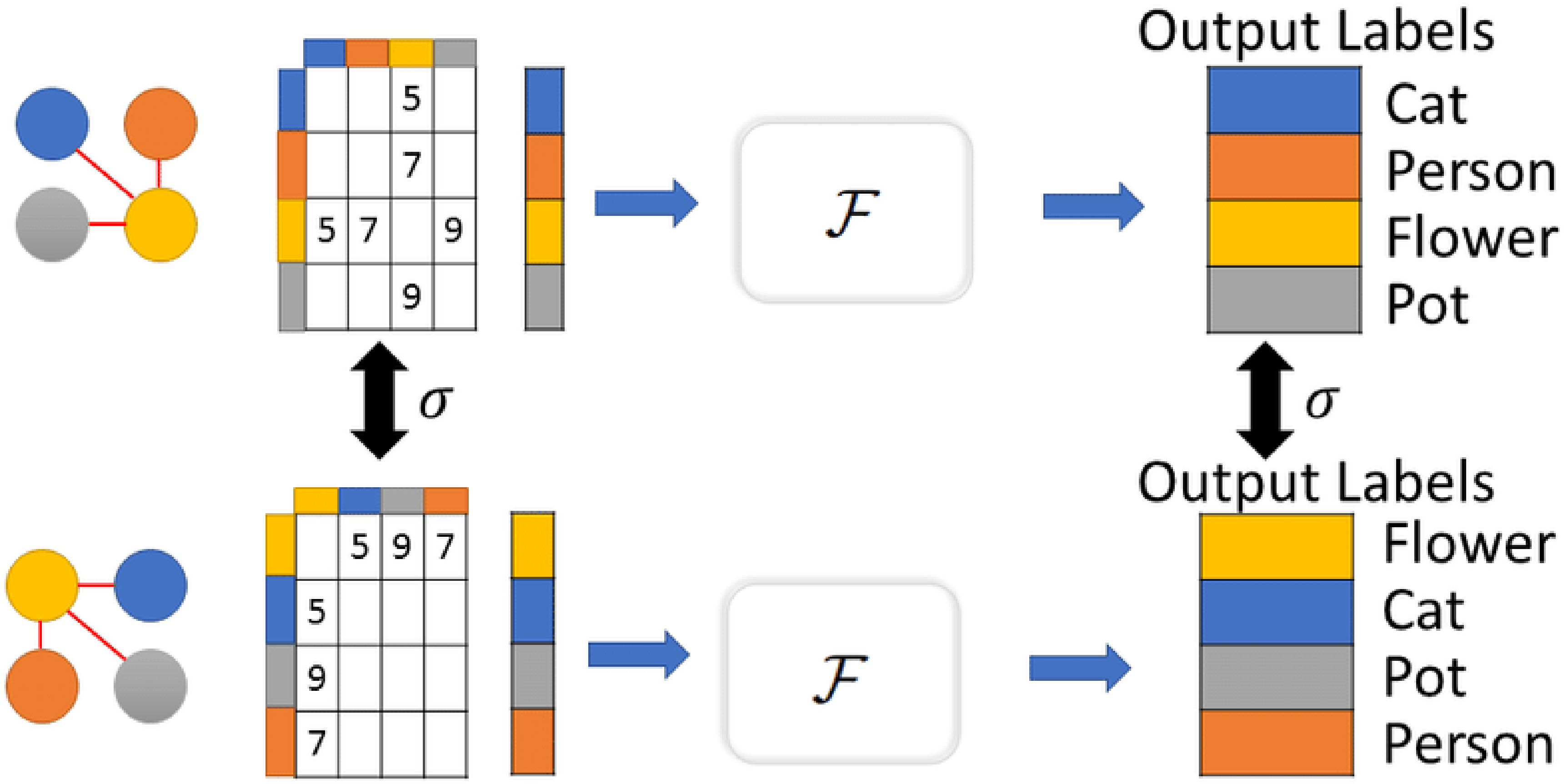
$$H^{(l+1)} = \sigma\left(A\left(H^{(l)}W^{(l)} + b^{(l)}\right)\right)$$

learnable parameters are shared

Sharing weights for all nodes in graph,
but nodes are differently updated by reflecting individual node features, $H_j^{(l)}$

Graph Convolutional Network (GCN)

Readout makes graph features a permutation invariance



Mapping Images to Scene Graphs with Permutation-Invariant Structured Prediction - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/Graph-permutation-invariance-and-structured-prediction-A-graph-labeling-function-F-is_fig1_323217335 [accessed 8 Sep, 2018]

Graph Convolutional Network (GCN)

Readout makes graph features a permutation invariance

- **Graph feature**

$$z_G = f\left(\left\{H_i^{(L)}\right\}\right)$$

1) Node-wise summation

$$z_G = \tau\left(\sum_{i \in G} MLP\left(H_i^{(L)}\right)\right)$$

Summation over all existing nodes in the graph satisfies the permutation invariance

2) Graph gathering

$$z_G = \tau\left(\sum_{i \in G} \sigma\left(MLP_1\left(H_i^{(L)}, H_i^{(0)}\right)\right) \odot MLP_2\left(H_i^{(L)}\right)\right)$$

- τ : ReLU activation
- σ : sigmoid activation

Gilmer, Justin, et al.
"Neural message passing for quantum chemistry." *arXiv preprint arXiv:1704.01212* (2017).

Graph Convolutional Network (GCN)

Readout makes graph features a permutation invariance

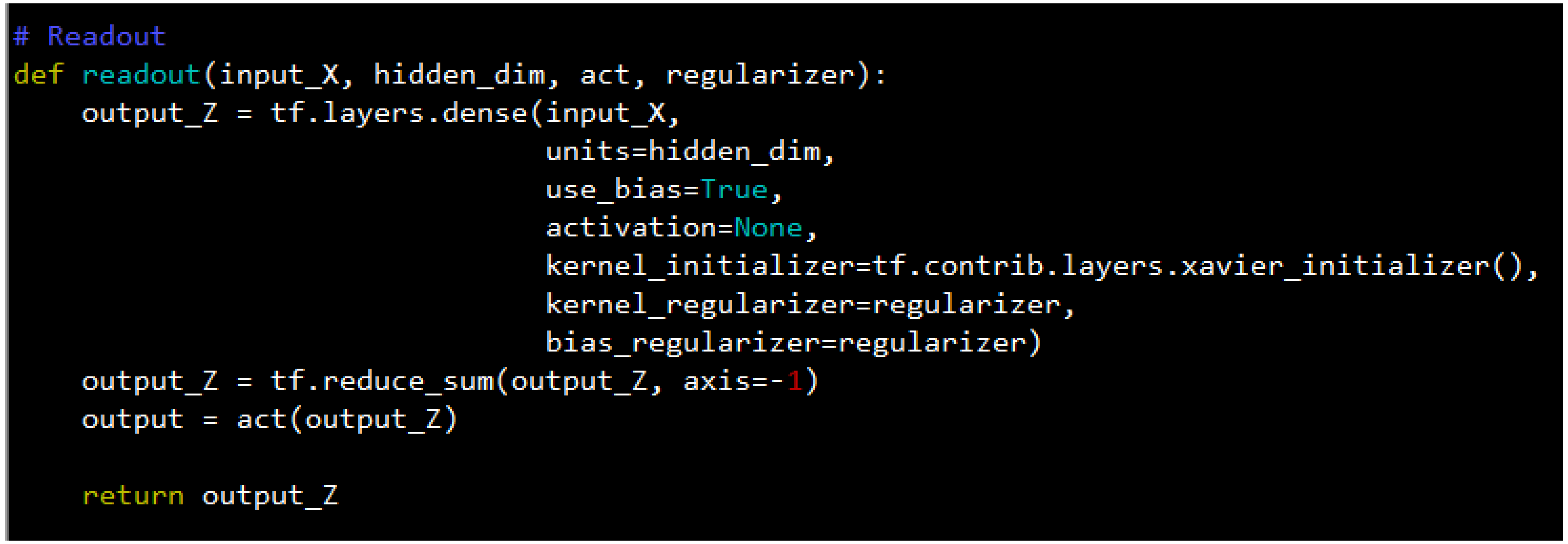
- **Graph feature**

$$z_G = f\left(\left\{H_i^{(L)}\right\}\right)$$

1) Node-wise summation

$$z_G = \tau\left(\sum_{i\in G} MLP\left(H_i^{(L)}\right)\right)$$

Summation over all existing nodes in the graph satisfies the permutation invariance

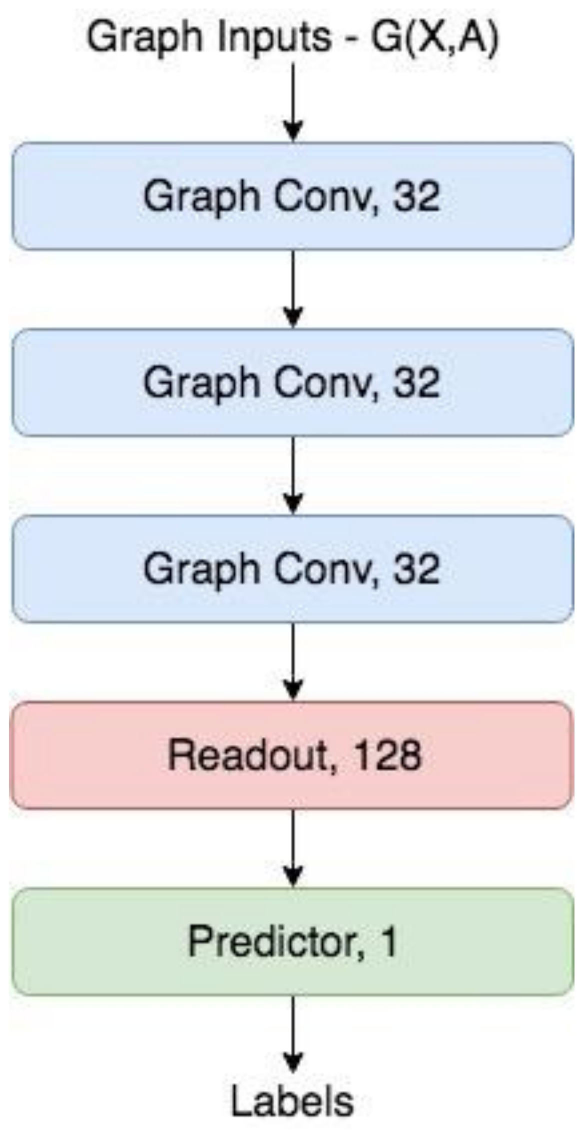


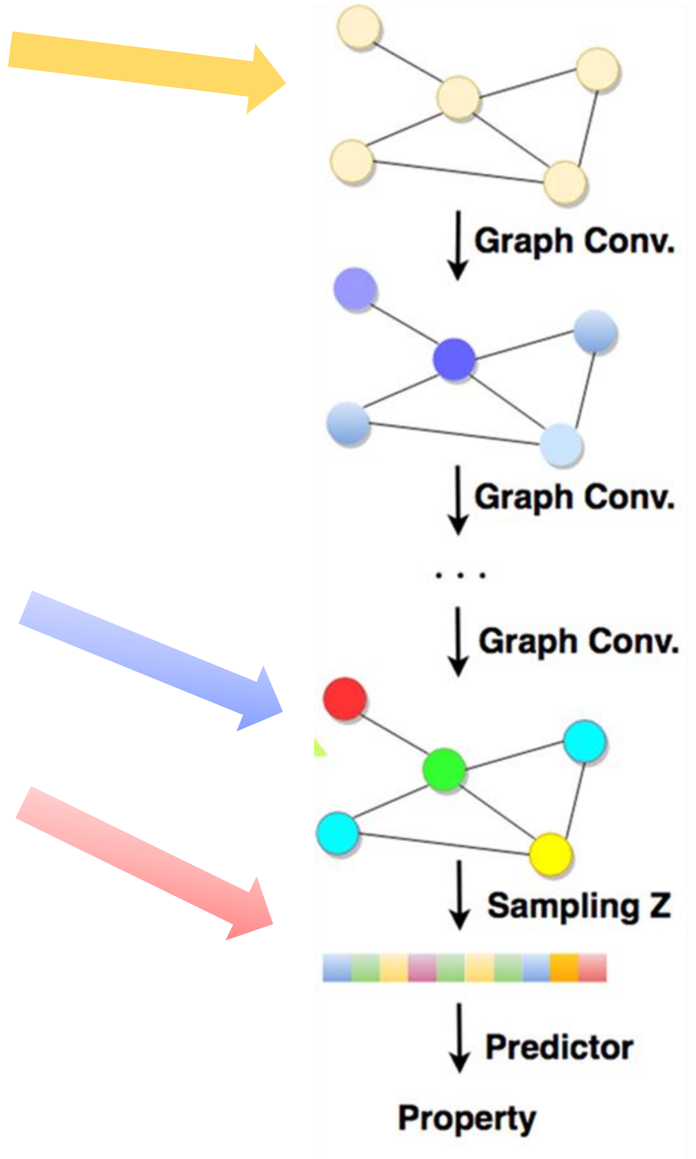
```
# Readout
def readout(input_X, hidden_dim, act, regularizer):
    output_Z = tf.layers.dense(input_X,
                               units=hidden_dim,
                               use_bias=True,
                               activation=None,
                               kernel_initializer=tf.contrib.layers.xavier_initializer(),
                               kernel_regularizer=regularizer,
                               bias_regularizer=regularizer)
    output_Z = tf.reduce_sum(output_Z, axis=-1)
    output = act(output_Z)

    return output_Z
```

- τ : ReLU activation
- σ : sigmoid activation

Gilmer, Justin, et al.
"Neural message passing for quantum chemistry." *arXiv preprint arXiv:1704.01212* (2017).

Graph Convolutional Network (GCN)





Input node features, $\left\{H_i^{(0)}\right\}$

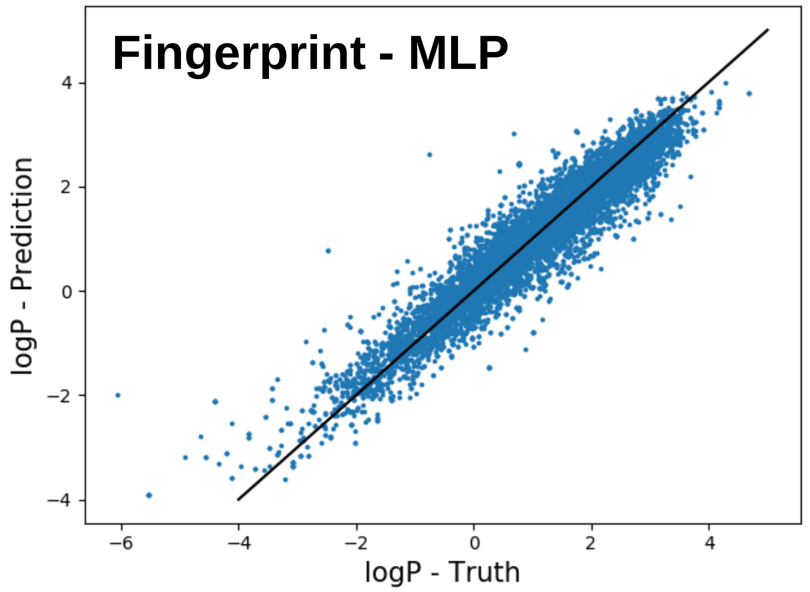
Raw node information

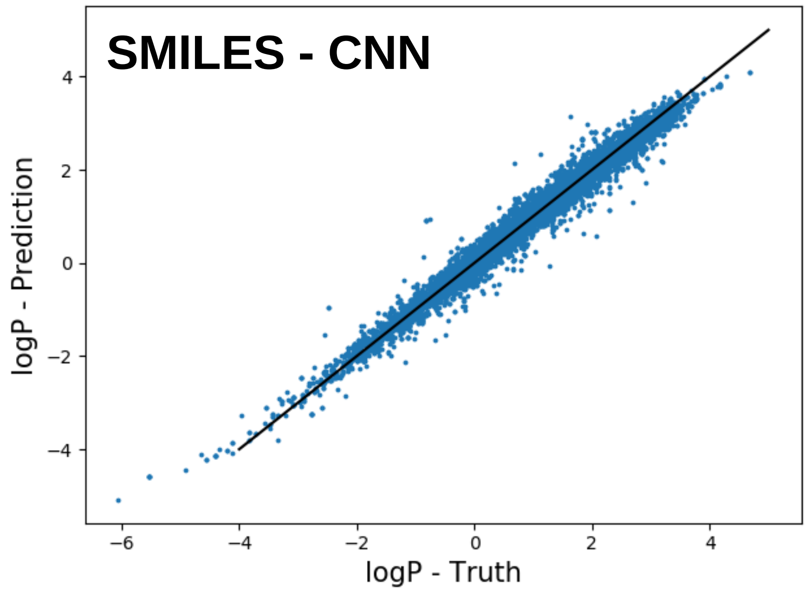
Final node states, $\left\{H_i^{(L)}\right\}$

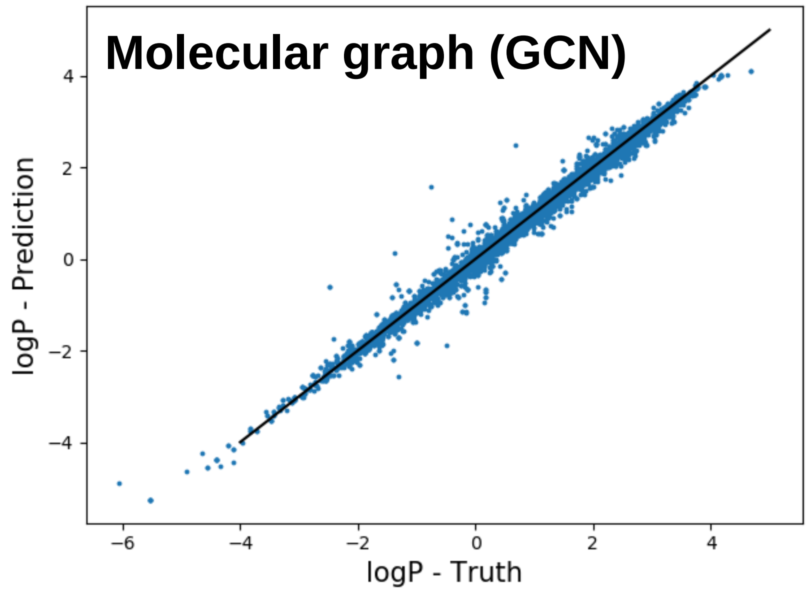
Graph features, *Z*

Graph Convolutional Network (GCN)

Comparison: (input data representation, model architecture)







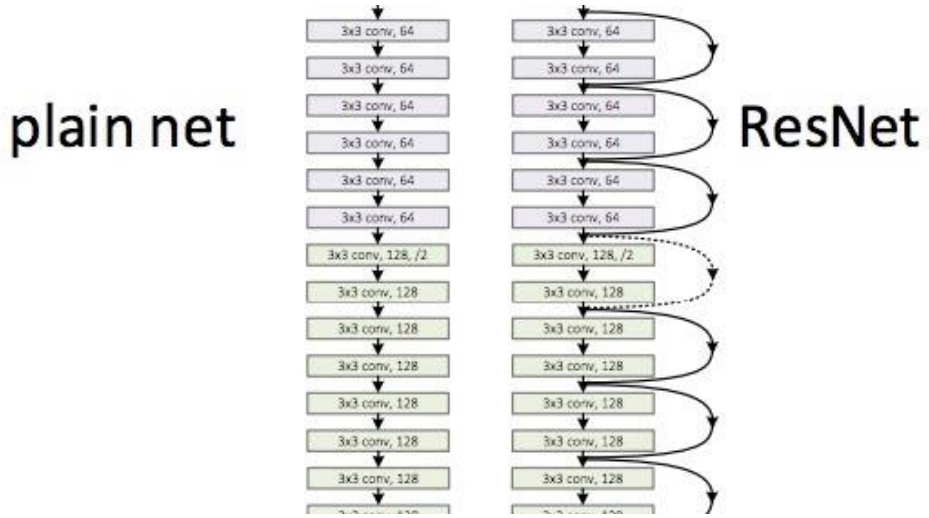
GCN setup

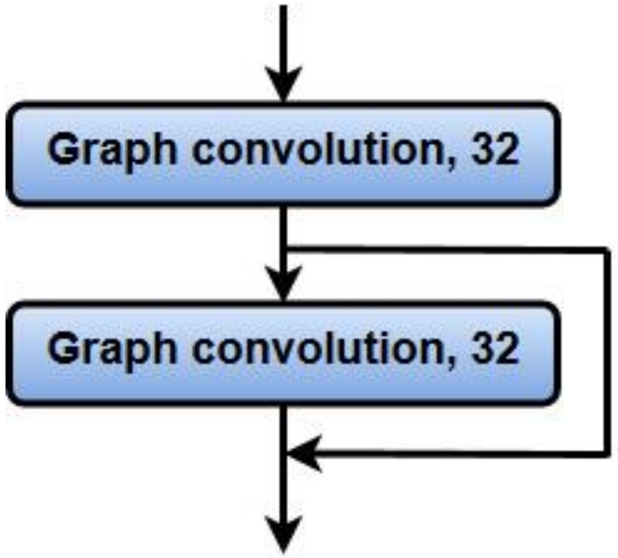
- ✓ Graph convolution layer dim : 64
- ✓ Readout, MLP dim : 256
- ✓ No dropout
- ✓ Regularization lambda = 0.001
- ✓ Adam optimizer, init_lr = 0.001

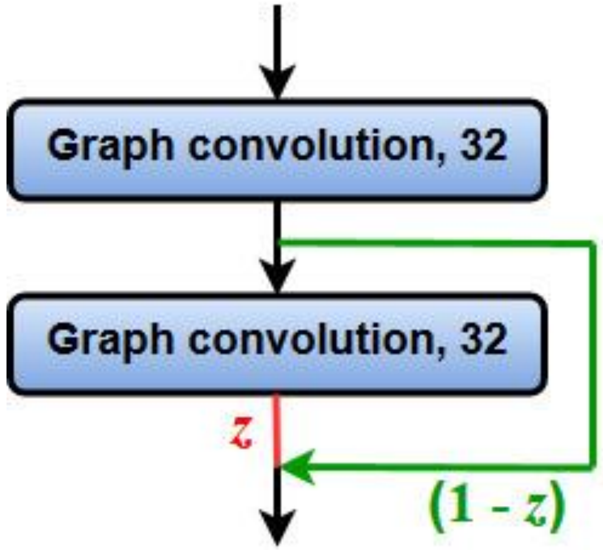
	Fingerprint - MLP	SMILES - CNN	Graph - GCN
MAE	0.31	0.15	0.088
Std. dev	0.42	0.20	0.137

Graph Convolutional Network (GCN)

Effect of using the skip-connection and gated-skip connection







Inspired from **ResNet**, which is one of the most successful NN **in vision recognition**

$$H_{i,sc}^{(l+1)} = H_i^{(l+1)} + H_i^{(l)}$$

$$H_{i,gsc}^{(l+1)} = z_i \odot H_i^{(l+1)} + (1 - z_i) \odot H_i^{(l)}$$

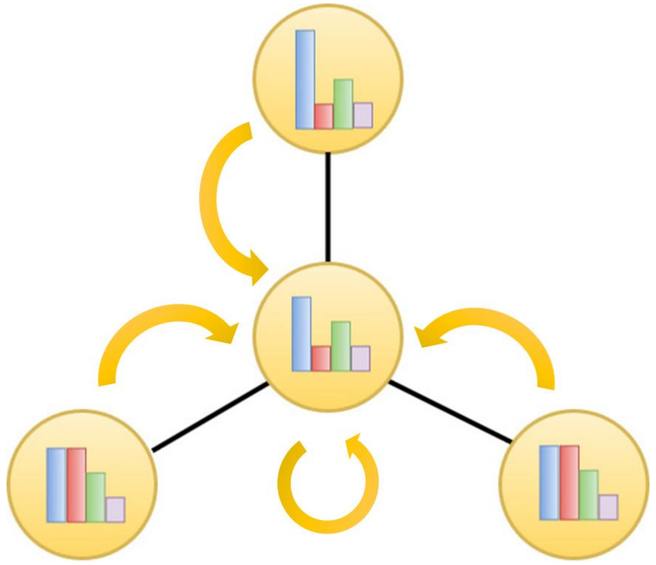
$$z_i = \sigma\left(U_{z,1} H_i^{(l+1)} + U_{z,2} H_i^{(l)} + b_z\right)$$

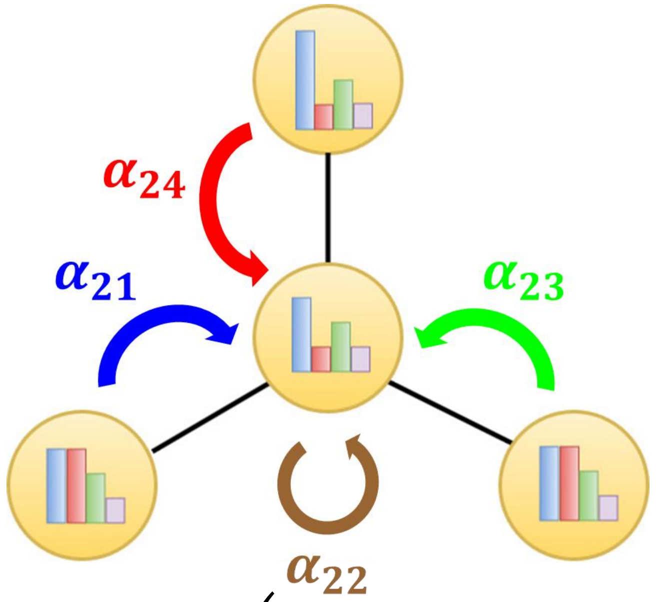
Graph Convolutional Network (GCN)

Effect of using the attention mechanism

Vanilla GCN updates information of neighbor atoms **with same importance**.

Attention mechanism enables it to update nodes **with different importance**



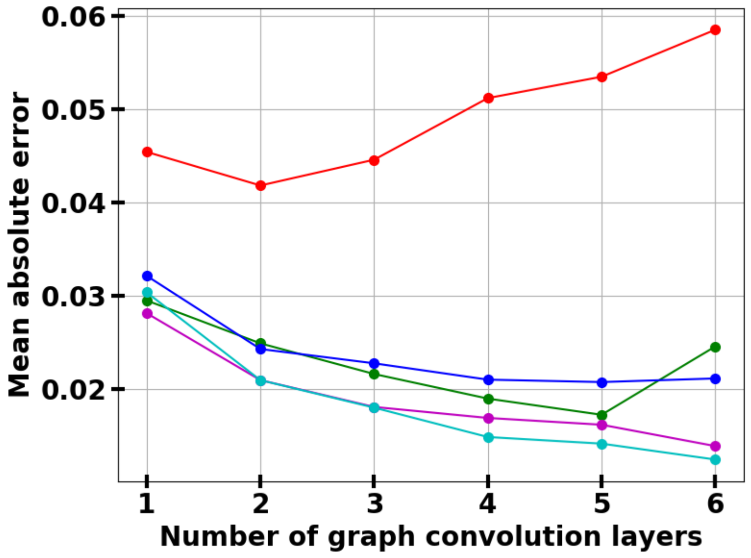


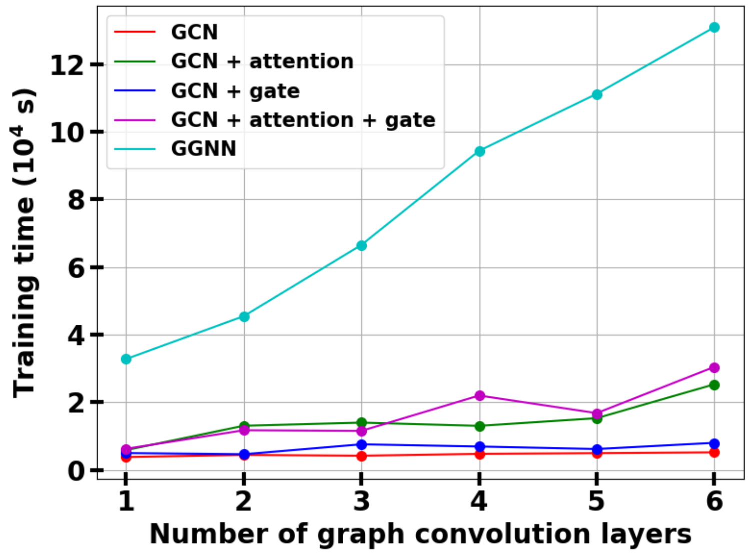
$$H^{(l+1)} = \sigma\left(\sum_{j\in N(i)} H_j^{(l)} W^{(l)}\right)$$

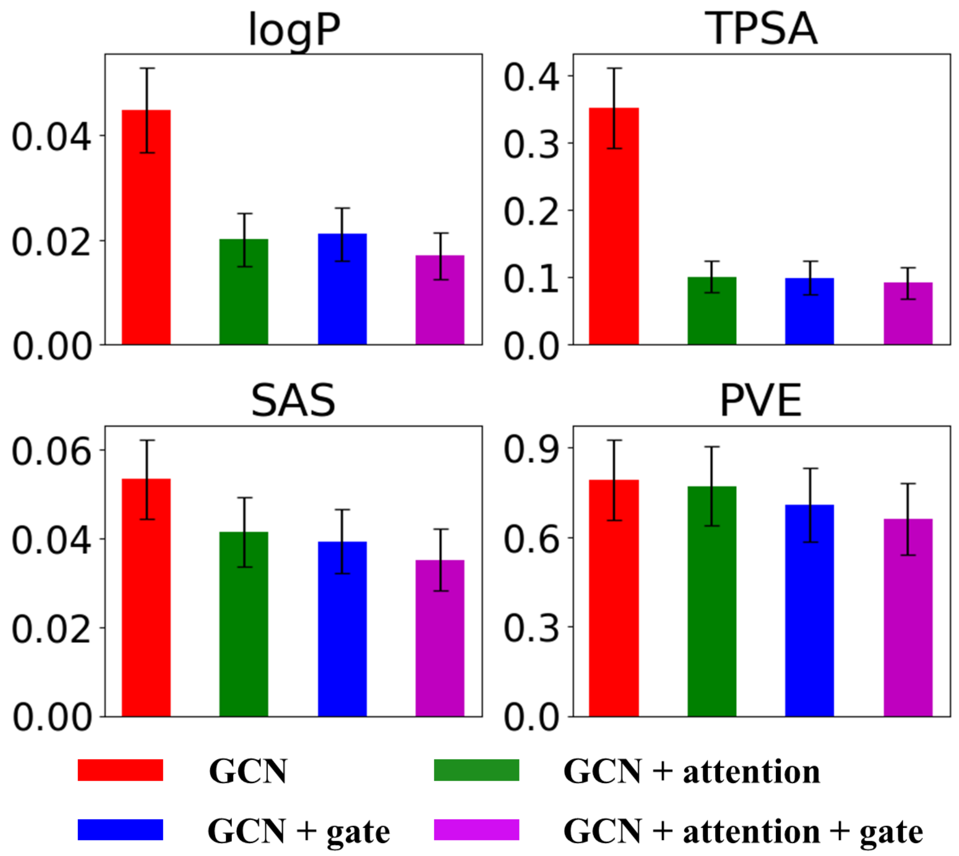
$$H^{(l+1)} = \sigma\left(\sum_{j\in N(i)} \alpha_{ij} H_j^{(l)} W^{(l)}\right)$$

Ryu, Seongok, Jaechang Lim, Seung Hwan Hong and Woo Youn Kim. "Deeply learning molecular structure-property relationships using attention- and gate-augmented graph convolutional network." *arXiv preprint arXiv:1805.10988* (2018).

Graph Convolutional Network (GCN)







- The GCN+attention+gate improves the vanilla GCN best.
- It shows comparable results and requires much lower computational costs than GGNN.

Ryu, Seongok, Jaechang Lim, Seung Hwan Hong and Woo Youn Kim. "Deeply learning molecular structure-property relationships using attention- and gate-augmented graph convolutional network." *arXiv preprint arXiv:1805.10988* (2018).

Assignment #5

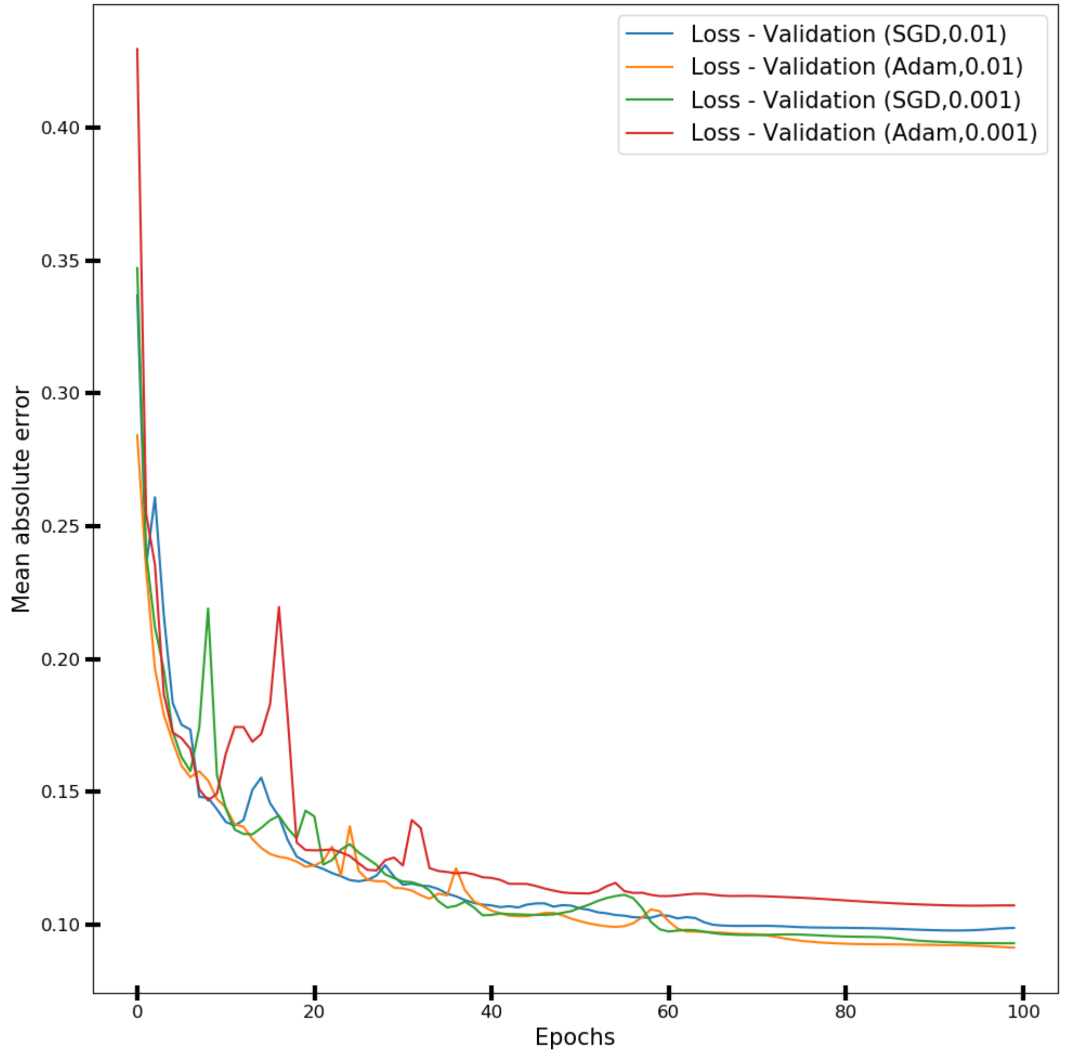
Improve the vanilla GCN model

- In this class, TA showed the vanilla GCN model – which is consisted of three graph convolution layers, a readout layer and a predictor composed of three dense layers.
- Also TA compared the performance of the vanilla GCN, GCN w/ skip connection and GCN w/ gated-skip connection
- In this week, we learned i) the attention mechanism, ii) gated-skip connection, and iii) inception module.
- Therefore, **improving the vanilla GCN is an objective of this assignment.**
- **Report your results - MAE, std. dev, and truth-prediction plot.**
- **In this assignment, you must also report the how the script provides initial graph inputs.**

Bonus assignment

In preparation of this material, I observed that…

- ✓ Stochastic gradient desent (SGD) and Adam optimizers show slightly (sometimes quite) different performances.
- ✓ Of course, the performances depend on learning rates as well.
- ✓ Therefore, I will give additional points to who submit survey on optimizers in deep learning.
- ✓ Survey will greatly help you to understand numerical optimization processes in deep learning.



References

- https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f
- Kingma, D. P., & Ba, J. L. (2015). Adam: a Method for Stochastic Optimization. International Conference on Learning Representations
- Li, Hao, et al. "Visualizing the loss landscape of neural nets." *arXiv preprint arXiv:1712.09913* (2017).