Deep Learning Standalone

–

for Chemistry

1. ML Basic
2. Pytorch Basic
3. MLP with Fingerprint Representation
4. CNN with SMILES Representation
5. GNN with Graph Representation
6. Experiment Management and Hyperparameter Tuning with Tensorboard
7. Practical Tips

Topics to learn in this session

1. What is Graph?

Graph Structure

2. Graph Convolutional Network (GCN)

Basic of GCN

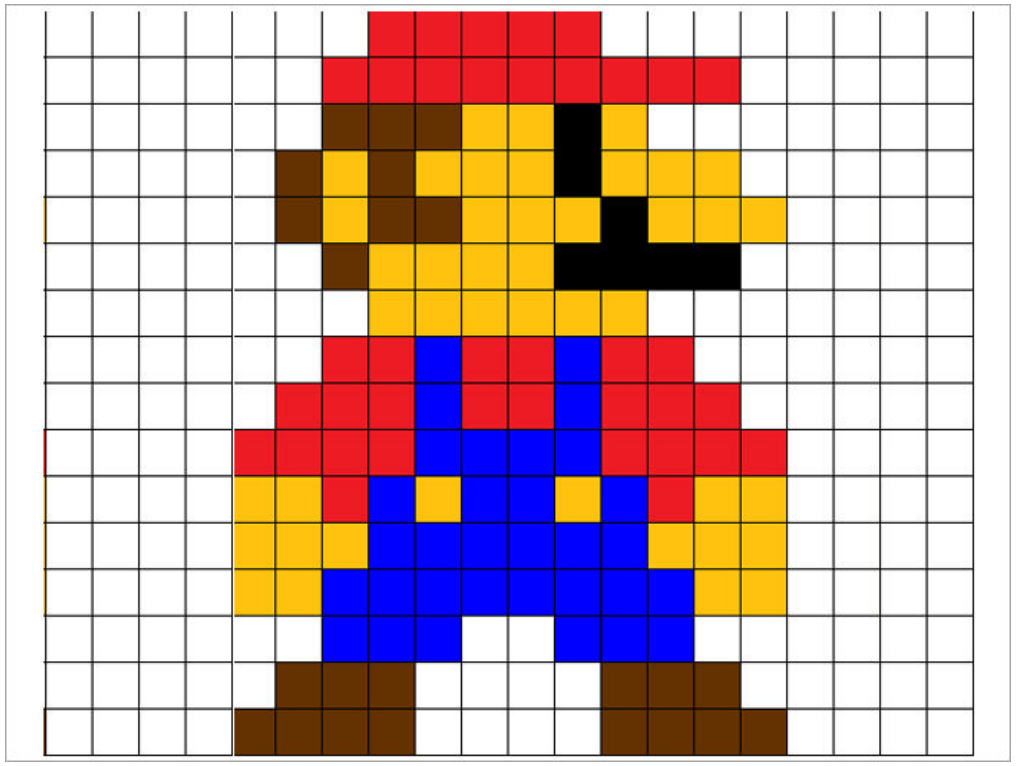
Advanced Techniques of GCN

3. Implementing GCN with pytorch

Predicting lipophilicity value of molecules

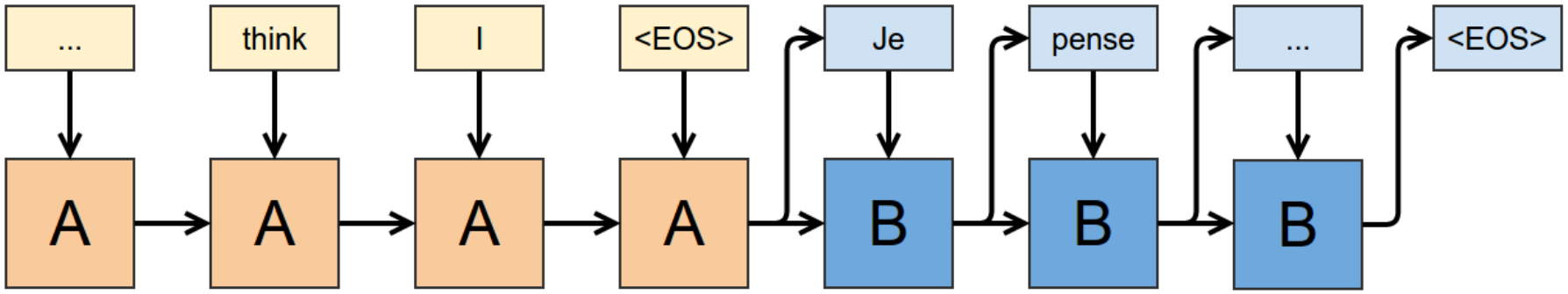
What is Graph?

Data Representation – Image and Sentence

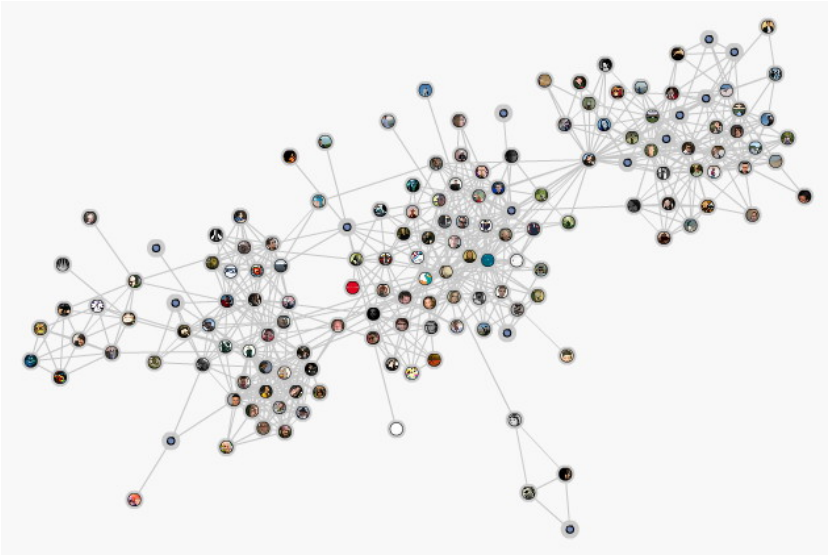


Images are represented with values on each pixel

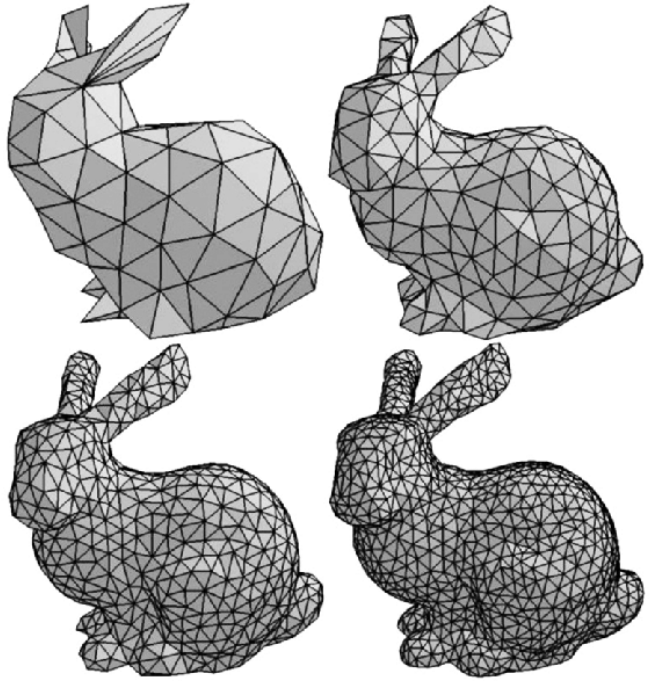
Sentences are represented with character values



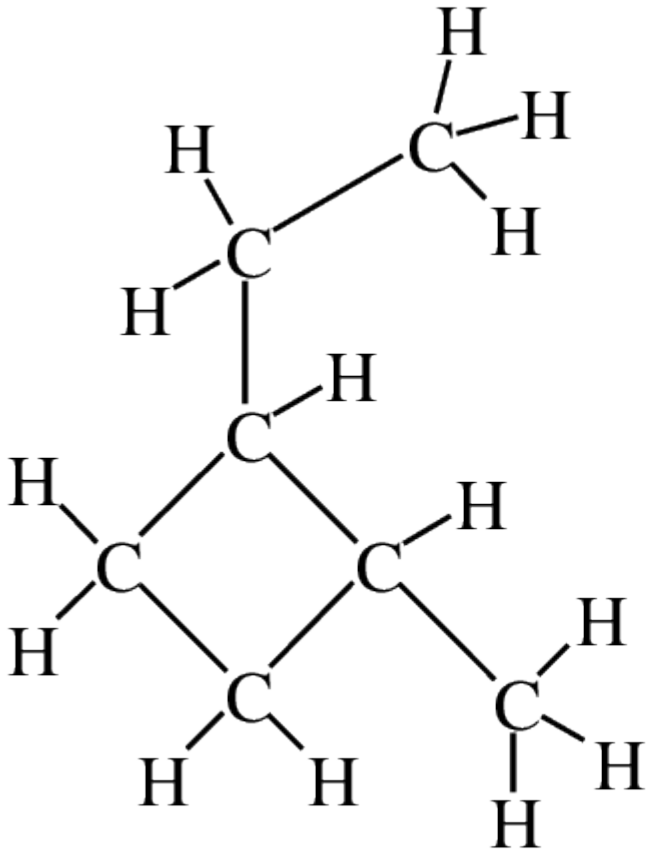
Data Representation – Graph



Social Graph

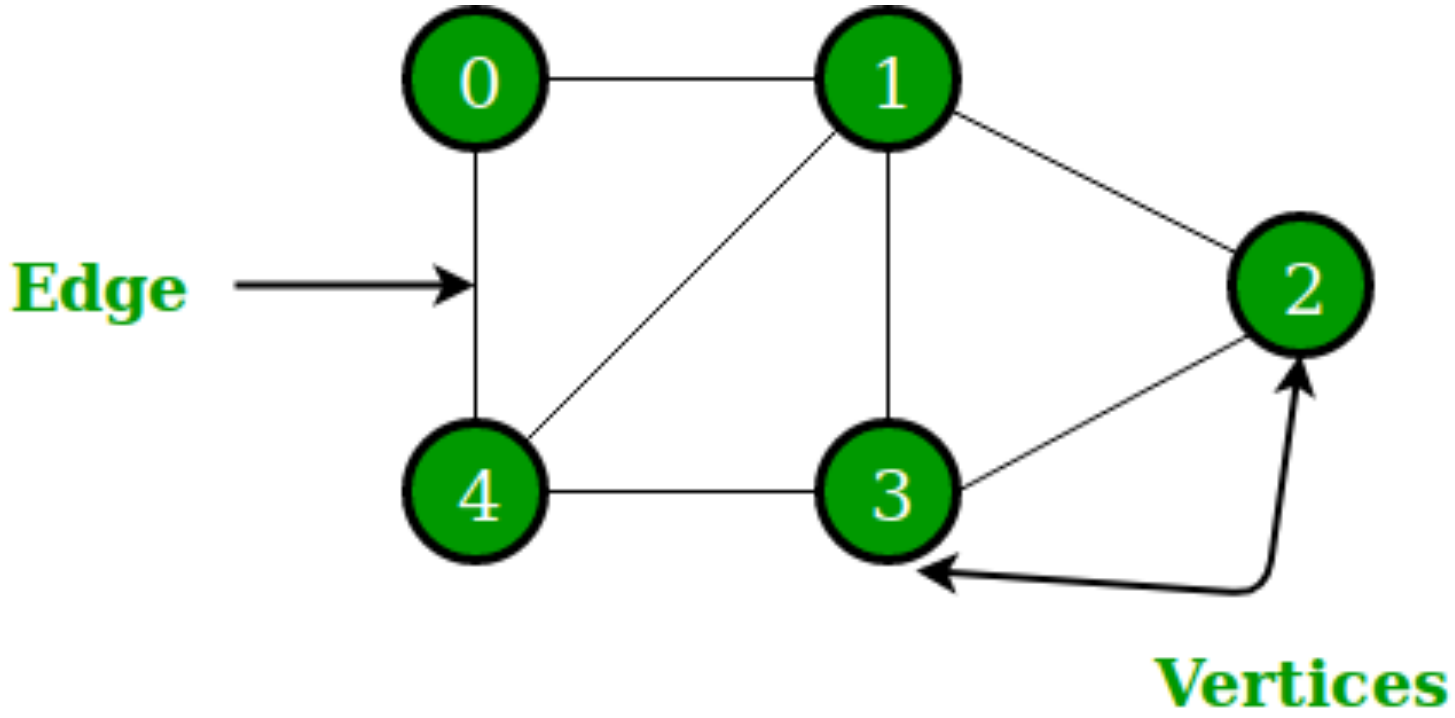


3D Mesh



Molecular Graph

Graph Structure



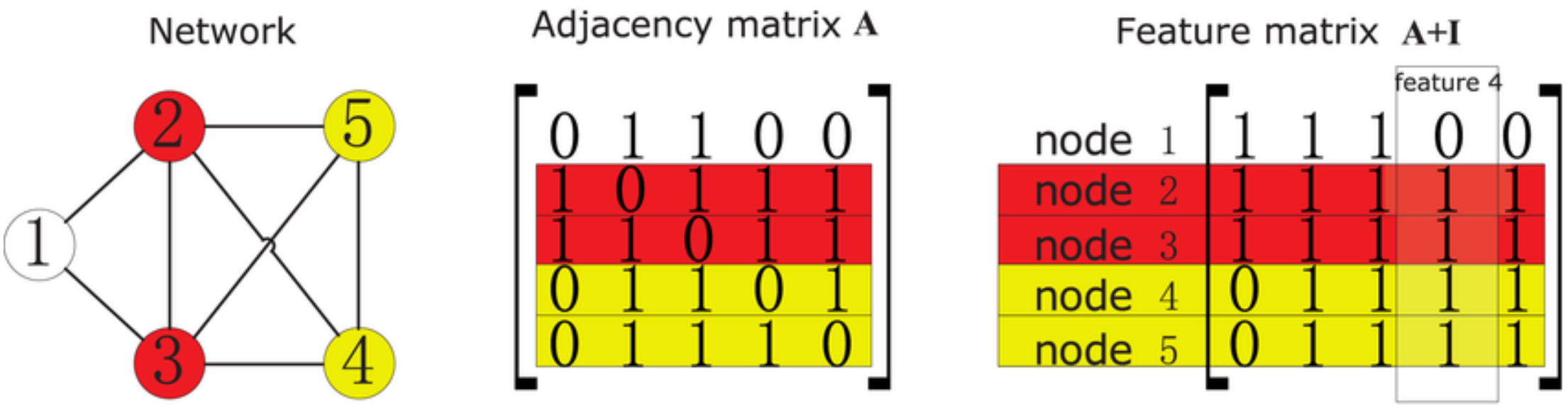
Vertex (Node)

Edge

Graph Structure

Vertex (Node)
Node Feature Matrix

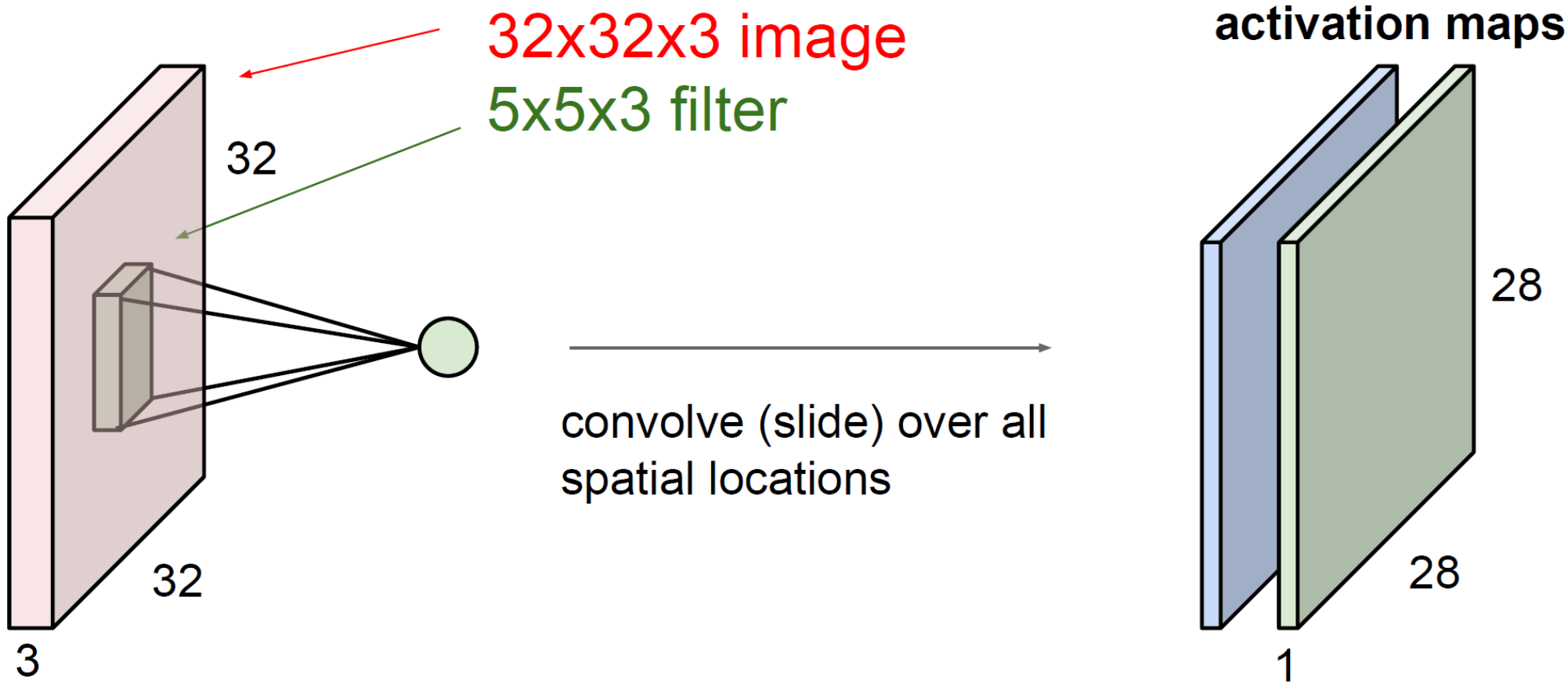
Edge
Adjacency Matrix



Graph Convolution Network

Convolution Layer

: Preserve the spatial structure



Weight sharing

Reduce the number of parameters
–〉 less overfitting, low computational cost

Learn local features

Translation invariance

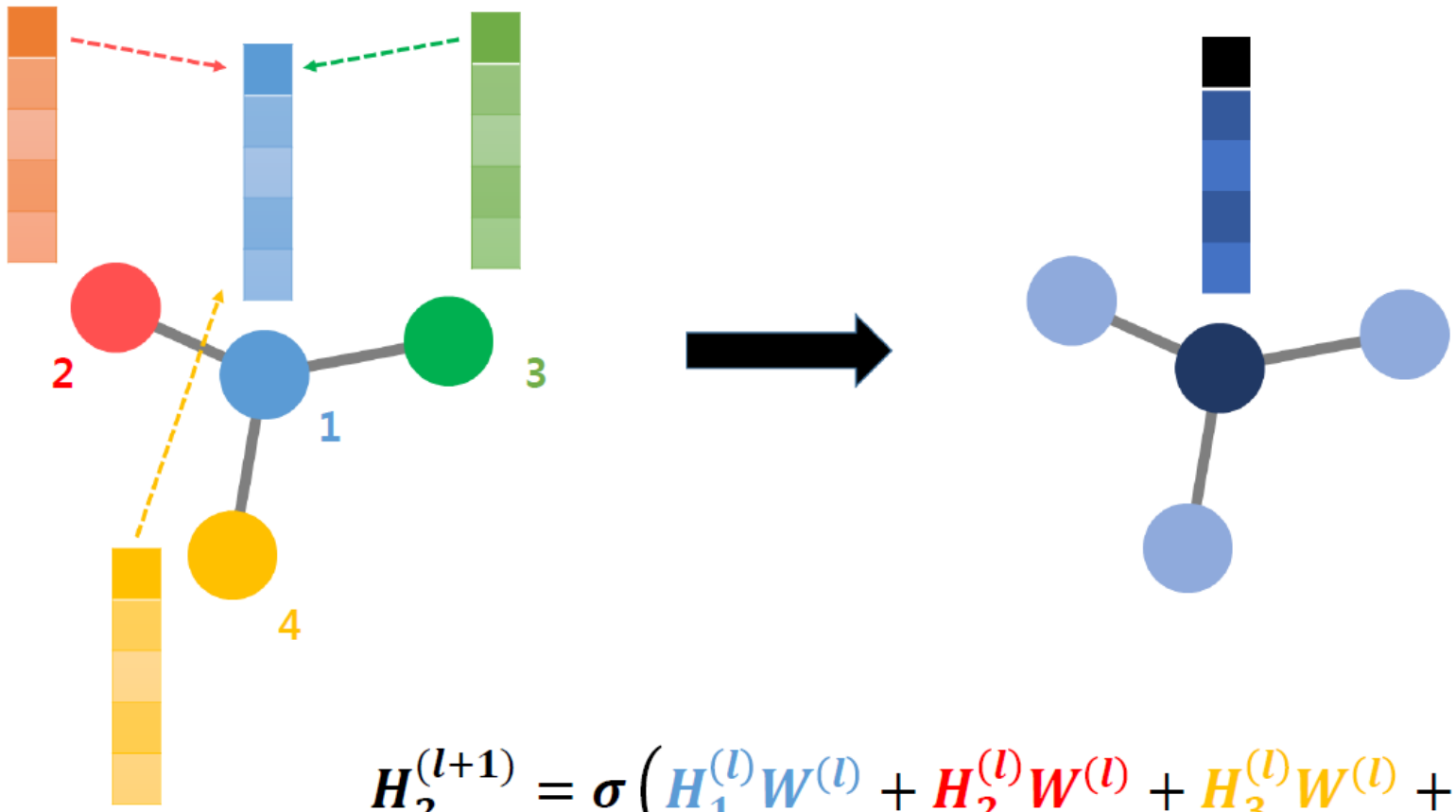
What should we update?

CNN updates values in activation map in each layer.
Values of activation map determine the state of image.

Values of each node feature determine the state of graph.

–〉 Make each layer of network update values of each node feature

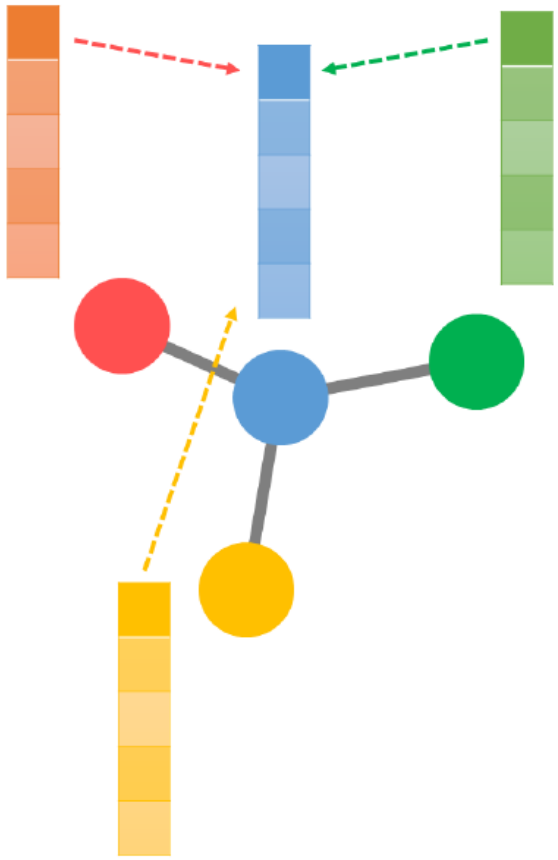
How to update hidden states in GCN

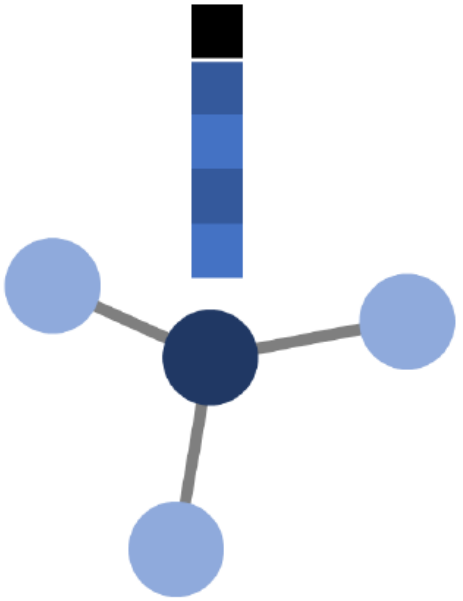


$$H_2^{(l+1)} = \sigma\left(H_1^{(l)}W^{(l)} + H_2^{(l)}W^{(l)} + H_3^{(l)}W^{(l)} + H_4^{(l)}W^{(l)} + b^{(l)}\right)$$

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

How to update hidden states in GCN

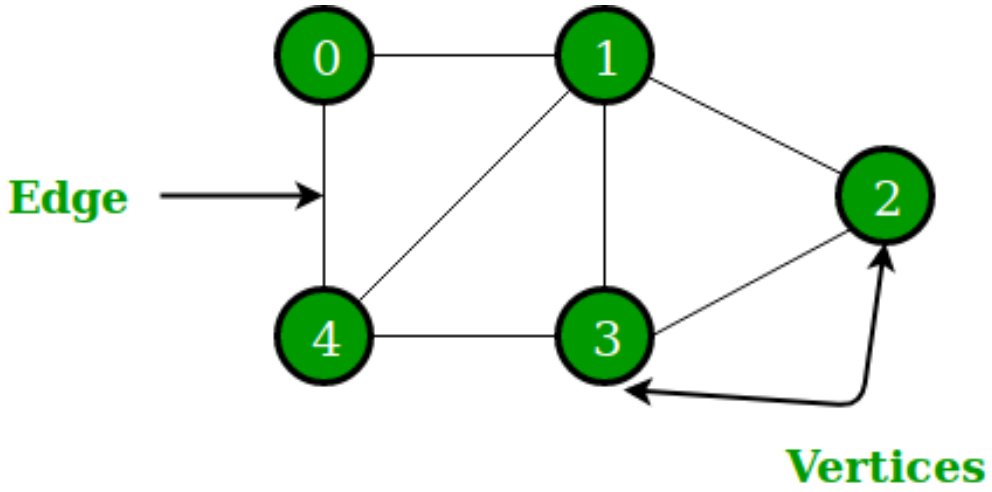




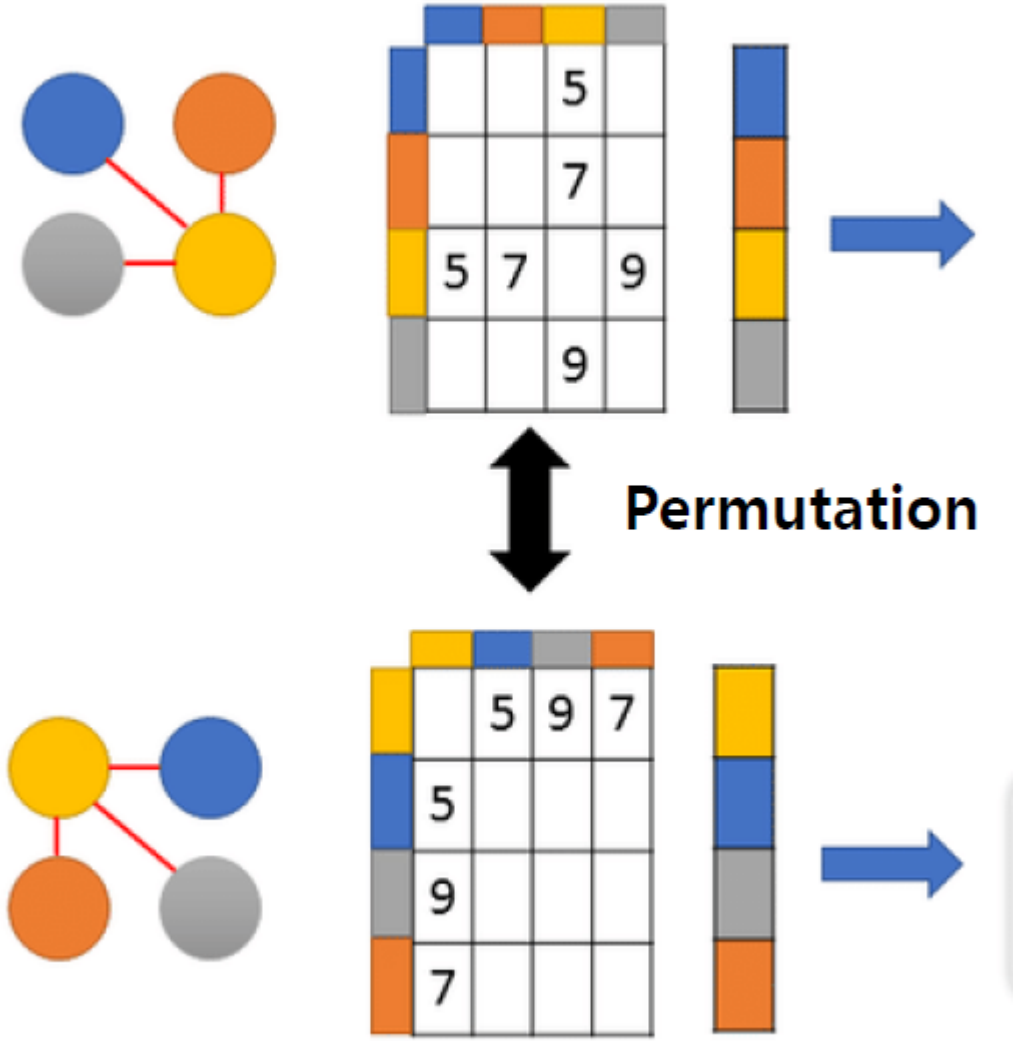
$$H^{(l+1)} = \sigma\left(AH^{(l)}W^{(l)} + b^{(l)}\right)$$

learnable parameters are shared

How to update hidden states in GCN



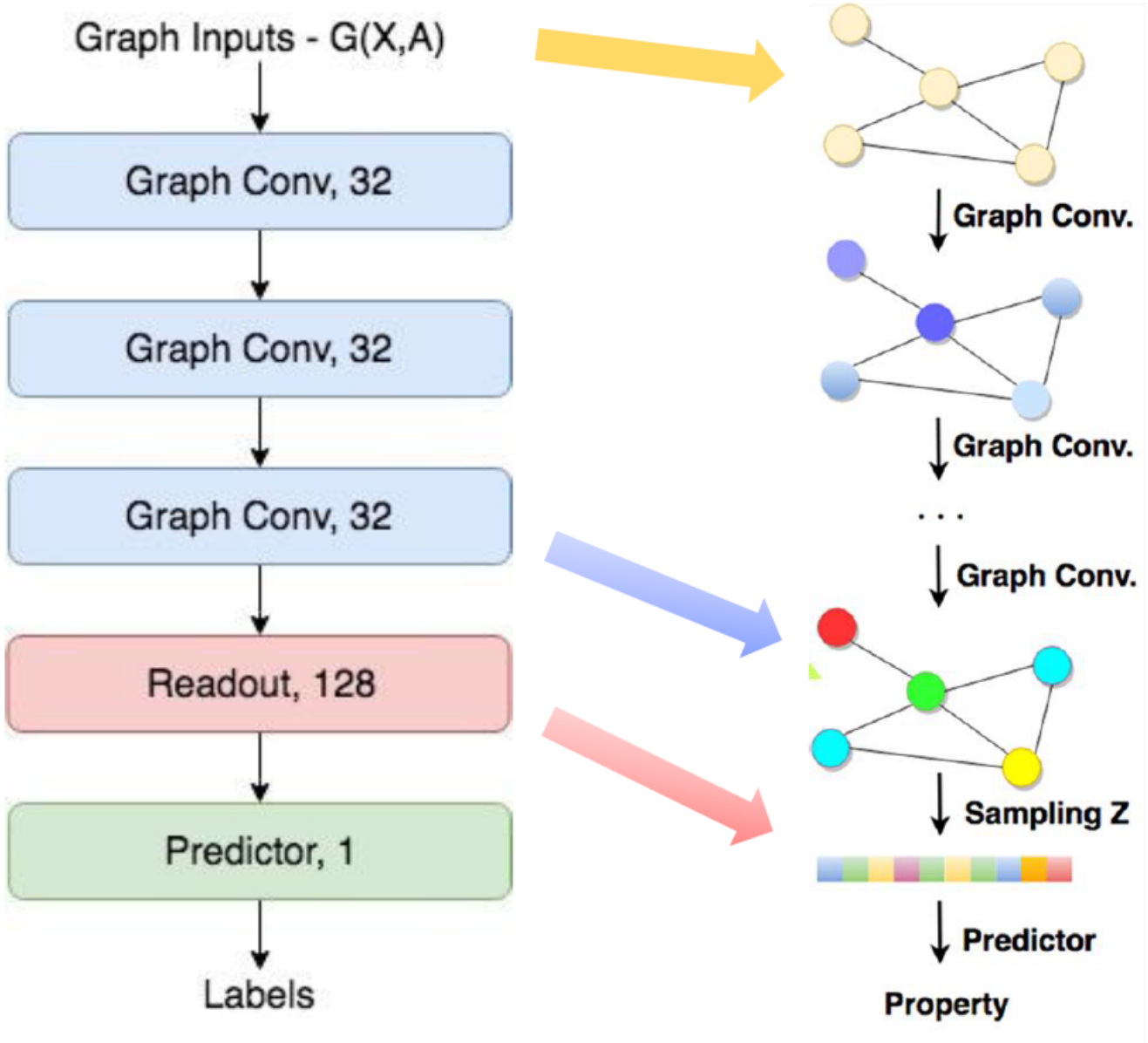
Readout – Permutation Invariance



Node-wise summation

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

Overall Structure of GCN



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

Raw node information

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

Graph features, Z

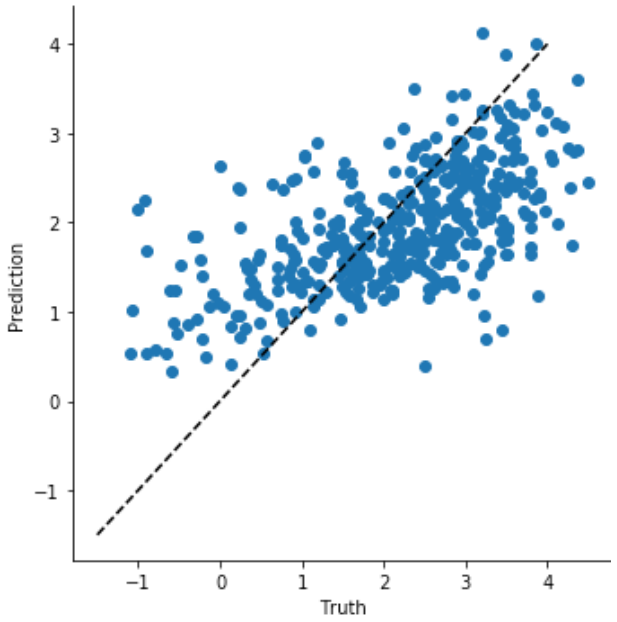
Implementing Vanilla GCN

1. Dataset and DataLoader
→ Return node matrix X and adjacency matrix A and lipophilicity y

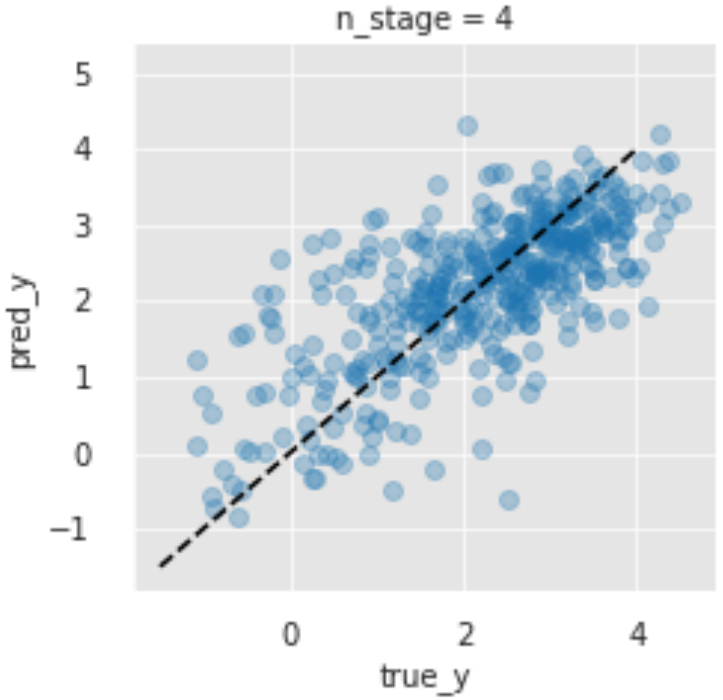
2. GCN architecture
→ Embed index matrix to one-hot-encoding
→ Graph convolution layer
→ 1D Batch Normalization for GCN
→ Readout Layer

3. Hyperparameter Tuning
→ Tweak hyperparameter to maximize the performance

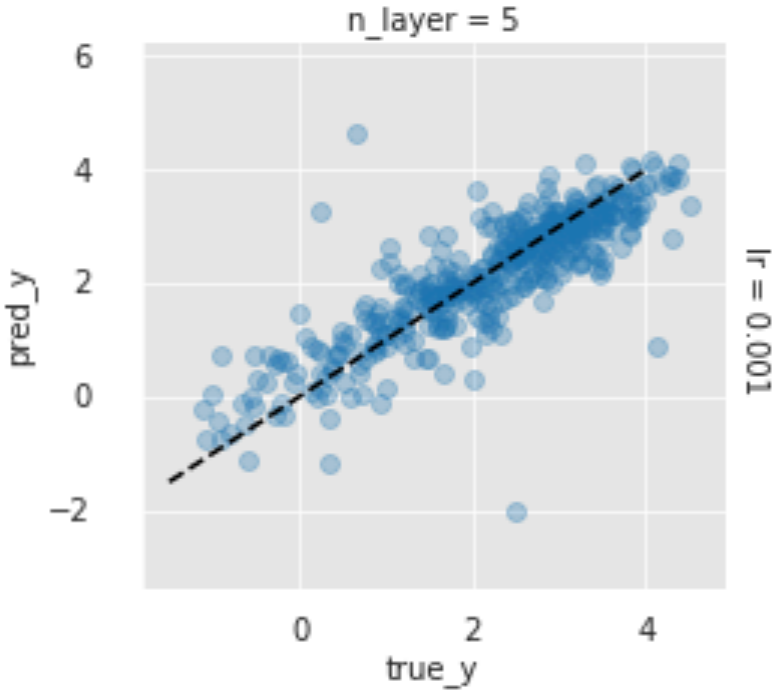
Results



2 hidden Layer MLP
(0.5 dropout rate)
MAE : 0.774



4 Layer CNN
(0.3 dropout rate)
MAE : 0.643



5 Layer GCN
(without dropout)
MAE : 0.436

It is worthy to employ GCN architecture!

Frustrating Implementation

PyTorch geometric

In detail, the following methods are currently implemented:

- **SplineConv** from Fey *et al.*: SplineCNN: Fast Geometric Deep Learning with Continuous B-Spline Kernels (CVPR 2018)
- **GCNConv** from Kipf and Welling: Semi-Supervised Classification with Graph Convolutional Networks (ICLR 2017)
- **ChebConv** from Defferrard *et al.*: Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering (NIPS 2016)
- **NNConv** from Gilmer *et al.*: Neural Message Passing for Quantum Chemistry (ICML 2017)
- **CGConv** from Xie and Grossman: Crystal Graph Convolutional Neural Networks for an Accurate and Interpretable Prediction of Material Properties (Physical Review Letters 120, 2018)
- **ECConv** from Simonovsky and Komodakis: Edge-Conditioned Convolution on Graphs (CVPR 2017)
- **GATConv** from Veličković *et al.*: Graph Attention Networks (ICLR 2018)
- **SAGEConv** from Hamilton *et al.*: Inductive Representation Learning on Large Graphs (NIPS 2017)
- **GraphConv** from, *e.g.*, Morris *et al.*: Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks (AAAI 2019)
- **GatedGraphConv** from Li *et al.*: Gated Graph Sequence Neural Networks (ICLR 2016)
- **GINConv** from Xu *et al.*: How Powerful are Graph Neural Networks? (ICLR 2019)
- **ARMAConv** from Bianchi *et al.*: Graph Neural Networks with Convolutional ARMA Filters (CoRR 2019)
- **SGConv** from Wu *et al.*: Simplifying Graph Convolutional Networks (CoRR 2019)
- **APPNP** from Klicpera *et al.*: Predict then Propagate: Graph Neural Networks meet Personalized PageRank (ICLR 2019)
- **AGNNConv** from Thekumparampil *et al.*: Attention-based Graph Neural Network for Semi-Supervised Learning (CoRR 2017)
- **TAGConv** from Du *et al.*: Topology Adaptive Graph Convolutional Networks (CoRR 2017)
- **RGCNConv** from Schlichtkrull *et al.*: Modeling Relational Data with Graph Convolutional Networks (ESWC 2018)
- **SignedConv** from Derr *et al.*: Signed Graph Convolutional Network (ICDM 2018)
- **DNAConv** from Fey: Just Jump: Dynamic Neighborhood Aggregation in Graph Neural Networks (ICLR-W 2019)
- **EdgeConv** from Wang *et al.*: Dynamic Graph CNN for Learning on Point Clouds (CoRR, 2018)
- **PointConv** (including **Iterative Farthest Point Sampling**, dynamic graph generation based on **nearest neighbor** or **maximum distance**, and **k-NN interpolation** for upsampling) from Qi *et al.*: PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation (CVPR 2017) and PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space (NIPS 2017)
- **XConv** from Li *et al.*: PointCNN: Convolution On X-Transformed Points (official implementation) (NeurIPS 2018)
- **PPFConv** from Deng *et al.*: PPFNet: Global Context Aware Local Features for Robust 3D Point Matching (CVPR 2018)
- **GMMConv** from Monti *et al.*: Geometric Deep Learning on Graphs and Manifolds using Mixture Model CNNs (CVPR 2017)
- **FeaStConv** from Verma *et al.*: FeaStNet: Feature-Steered Graph Convolutions for 3D Shape Analysis (CVPR 2018)
- **HypergraphConv** from Bai *et al.*: Hypergraph Convolution and Hypergraph Attention (CoRR 2019)
- A **MetaLayer** for building any kind of graph network similar to the TensorFlow Graph Nets library from Battaglia *et al.*: Relational Inductive Biases, Deep Learning, and Graph Networks (CoRR 2018)
- **GlobalAttention** from Li *et al.*: Gated Graph Sequence Neural Networks (ICLR 2016)
- **Set2Set** from Vinyals *et al.*: Order Matters: Sequence to Sequence for Sets (ICLR 2016)
- **Sort Pool** from Zhang *et al.*: An End-to-End Deep Learning Architecture for Graph Classification (AAAI 2018)
- **Dense Differentiable Pooling** from Ying *et al.*: Hierarchical Graph Representation Learning with Differentiable Pooling (NeurIPS 2018)
- **Graclus Pooling** from Dhillon *et al.*: Weighted Graph Cuts without Eigenvectors: A Multilevel Approach (PAMI 2007)
- **Voxel Grid Pooling** from, *e.g.*, Simonovsky and Komodakis: Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs (CVPR 2017)
- **Top-K Pooling** from Gao and Ji: Graph U-Nets (ICML 2019), Cangea *et al.*: Towards Sparse Hierarchical Graph Classifiers (NeurIPS-W 2018) and Knyazev *et al.*: Understanding Attention and Generalization in Graph Neural Networks (ICLR-W 2019)
- **SAG Pooling** from Lee *et al.*: Self-Attention Graph Pooling (ICML 2019) and Knyazev *et al.*: Understanding Attention and Generalization in Graph Neural Networks (ICLR-W 2019)
- **Edge Pooling** from Diehl *et al.*: Towards Graph Pooling by Edge Contraction (ICML-W 2019) and Diehl: Edge Contraction Pooling for Graph Neural Networks (CoRR 2019)
- **Local Degree Profile** from Cai and Wang: A Simple yet Effective Baseline for Non-attribute Graph Classification (CoRR 2018)
- **Jumping Knowledge** from Xu *et al.*: Representation Learning on Graphs with Jumping Knowledge Networks (ICML 2018)
- **Node2Vec** from Grover and Leskovec: node2vec: Scalable Feature Learning for Networks (KDD 2016)
- **Deep Graph Infomax** from Veličković *et al.*: Deep Graph Infomax (ICLR 2019)
- All variants of **Graph Auto-Encoders** from Kipf and Welling: Variational Graph Auto-Encoders (NIPS-W 2016) and Pan *et al.*: Adversarially Regularized Graph Autoencoder for Graph Embedding (IJCAI 2018)
- **RENet** from Jin *et al.*: Recurrent Event Network for Reasoning over Temporal Knowledge Graphs (ICLR-W 2019)
- **GraphUNet** from Gao and Ji: Graph U-Nets (ICML 2019)
- **NeighborSampler** from Hamilton *et al.*: Inductive Representation Learning on Large Graphs (NIPS 2017)
- **GDC** from Klicpera *et al.*: Diffusion Improves Graph Learning (NeurIPS 2019)

Open Graph Benchmark

Open Graph Benchmark (OGB) is a collection of benchmark datasets, data-loaders and evaluators for graph machine learning in PyTorch.

Data-loaders are fully compatible with PyTorch Geometric (PYG) and Deep Graph Library (DGL). The goal is to have an easily-accessible standardized large-scale benchmark datasets to drive research in graph machine learning.

Name	Size	Description	Task
ogbg-mol-bace	1513	Human β-secretase 1 (BACE-1) inhibition data for a set of molecules	Binary classification
ogbg-mol-bbbp	2039	Blood-brain barrier penetration data for a set of molecules	Binary classification
ogbg-mol-clintox	1477	Clinical trial toxicity and FDA approval status for a set of molecules	Multi-label (2) binary classification
ogbg-mol-esol	1128	Water solubility data for a set of molecules	Regression
ogbg-mol-freesolv	642	Hydration free energy data for a set of molecules	Regression
ogbg-mol-hiv	41127	HIV replication inhibition data for a set of molecules	Binary classification
ogbg-mol-lipo	4200	Octanol/water distribution coefficient (logD) data for a set of molecules	Regression
ogbg-mol-muv	93087	A subset of molecules and their biological properties from the PubChem BioAssay database selected using a refined nearest neighbor analysis	Multi-label (17) binary classification
ogbg-mol-pcba	437929	A subset of molecules and their biological properties from the PubChem BioAssay database	Multi-label (128) binary classification
ogbg-mol-sider	1427	Side effect data grouped into organ classes for a set of molecules	Multi-label (27) binary classification
ogbg-mol-tox21	7831	"Toxicology in the 21st Century" dataset, containing measurements on diverse toxicity targets for a set of molecules	Multi-label (12) binary classification
ogbg-mol-toxcast	8576	High-throughput in vitro toxicology data for a set of molecules	Multi-label (617) binary classification

How to implement all kinds of GNN family and benchmark dataset?

Pytorch Geometric will save you.

Implemented GCN, GatedGCN, GIN architectures, including several pooling modules!

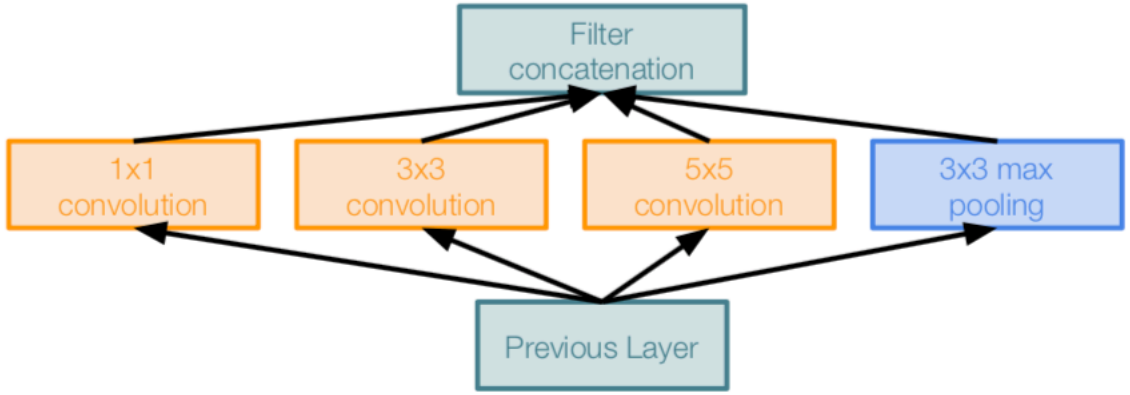
Stanford-OGBG will also save you.

Implemented MoleculeNet benchmark dataset with pytorch geometric

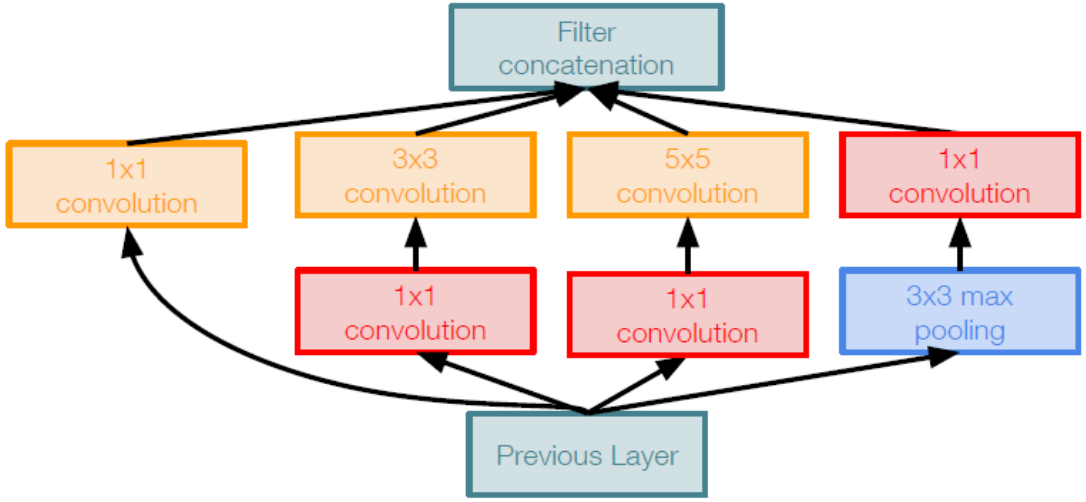
https://github.com/rusty1s/pytorch_geometric https://ogb.stanford.edu/ogbg-mol.html

Advanced Techniques of GCN

Inception



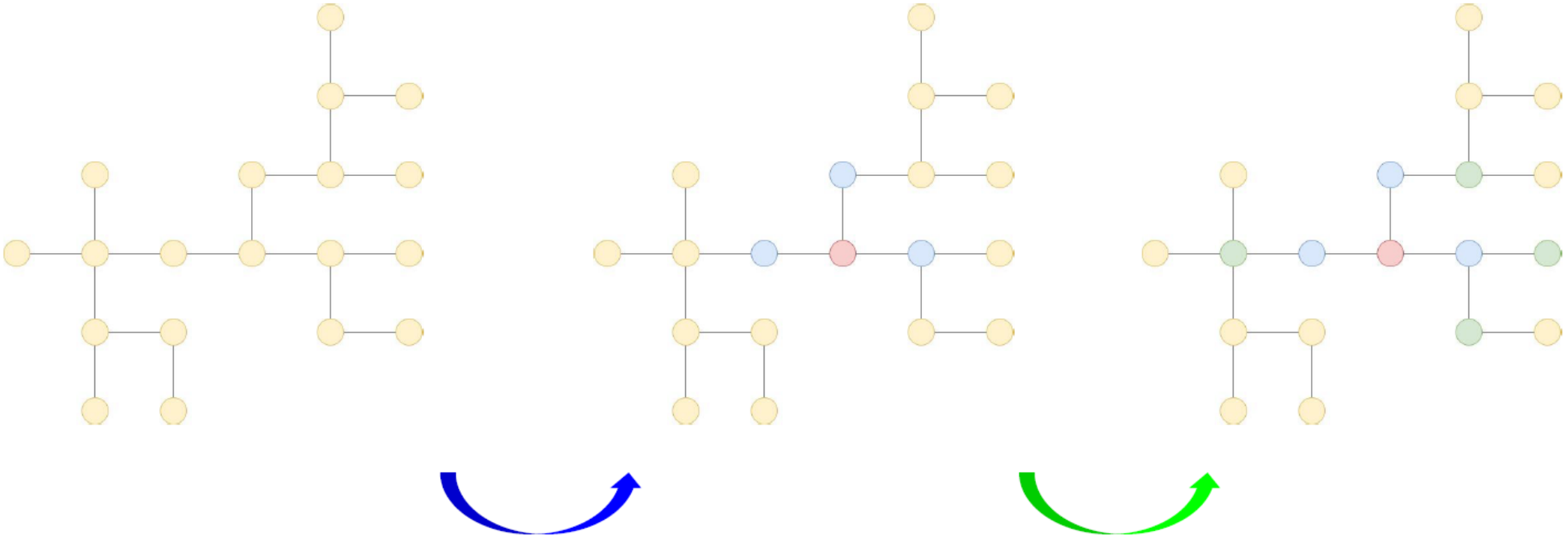
Naive Inception module



Inception module with dimension reduction

Inception

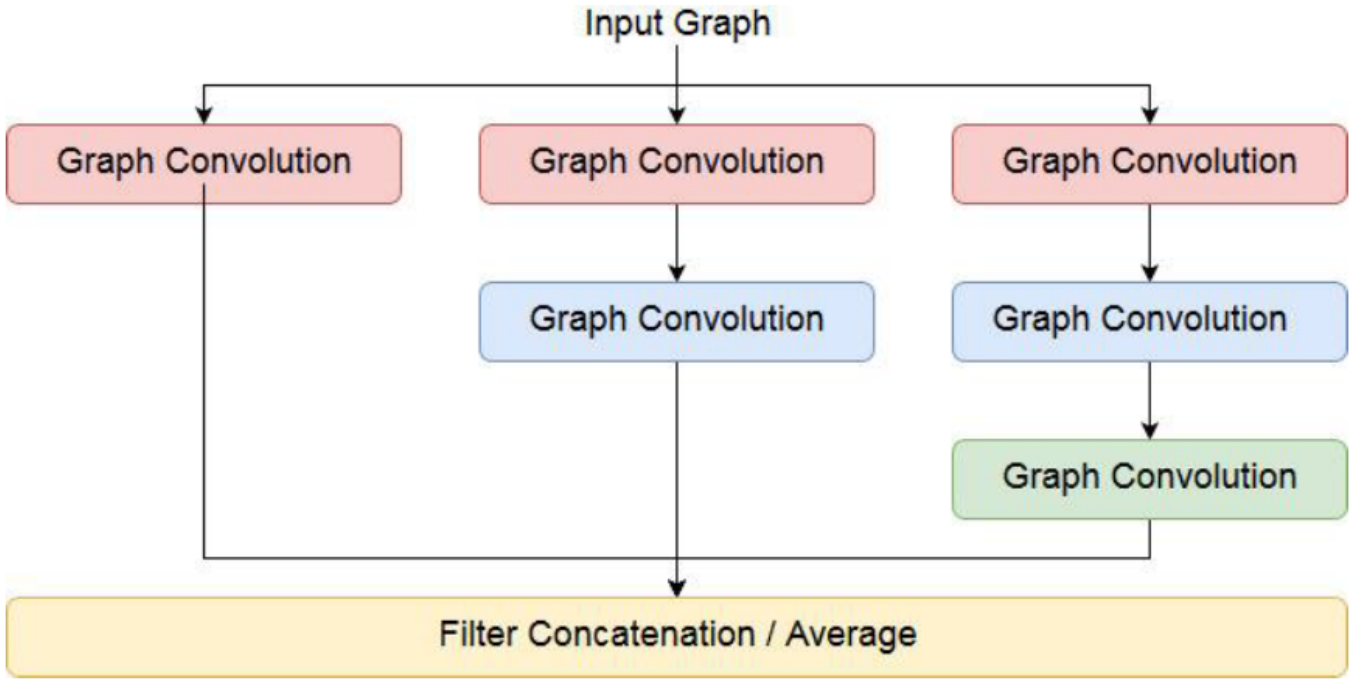
Single graph convolution reflects the first nearest neighbor information and subsequent **multiple graph convolution can deliver information of distant atoms.**



$$H^{(1)} = \sigma\left(AH^{(0)}W^{(0)}\right) \qquad H^{(2)} = \sigma\left(AH^{(1)}W^{(1)}\right)$$

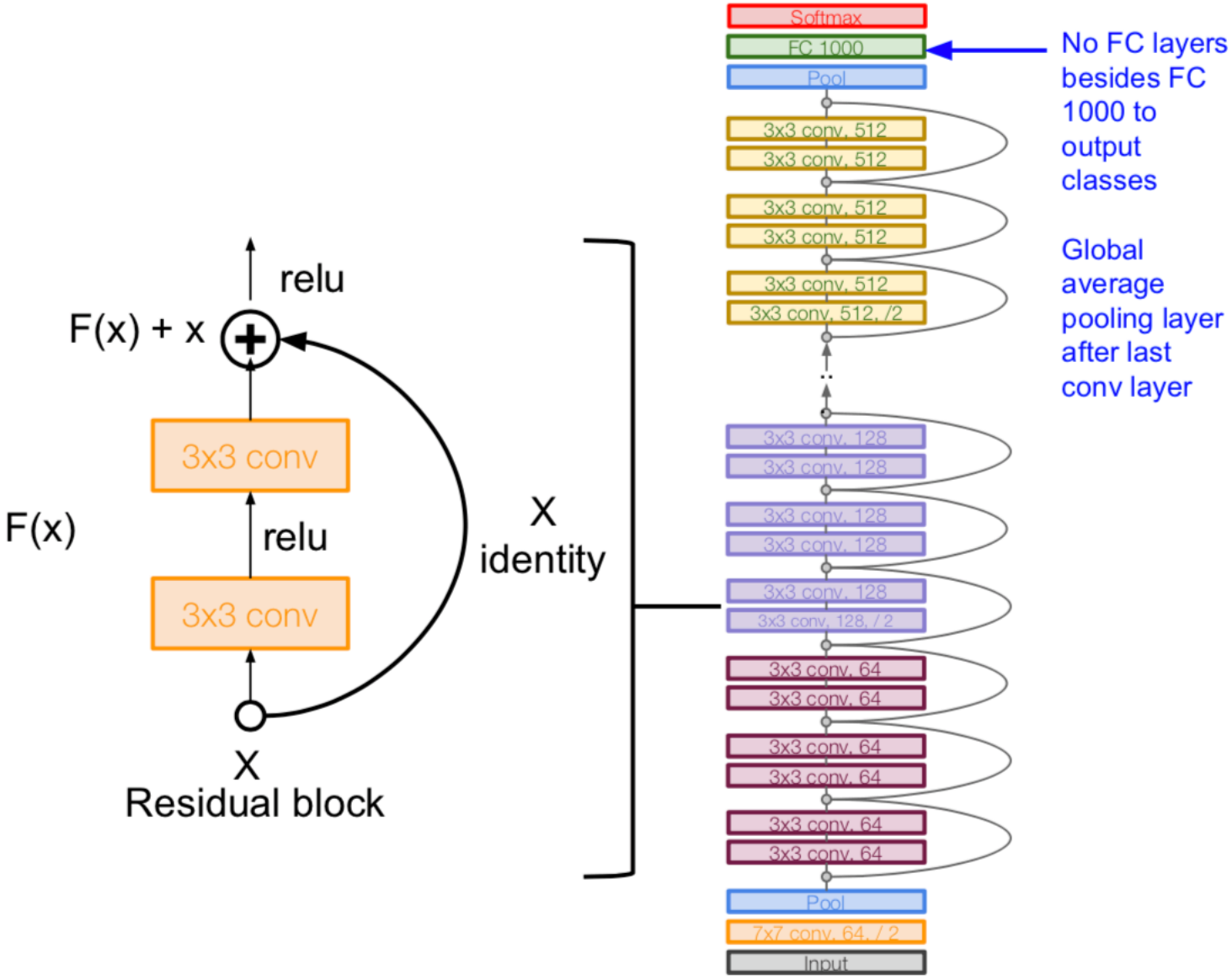
Inception

Inception module in GCN



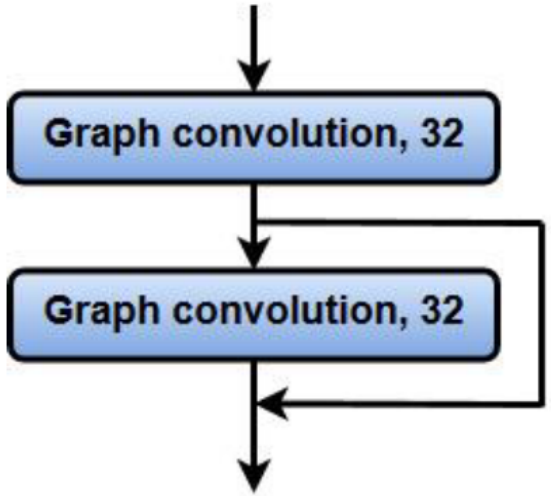
- Make network **wider**
- Avoid **vanishing gradient**

Skip Connection



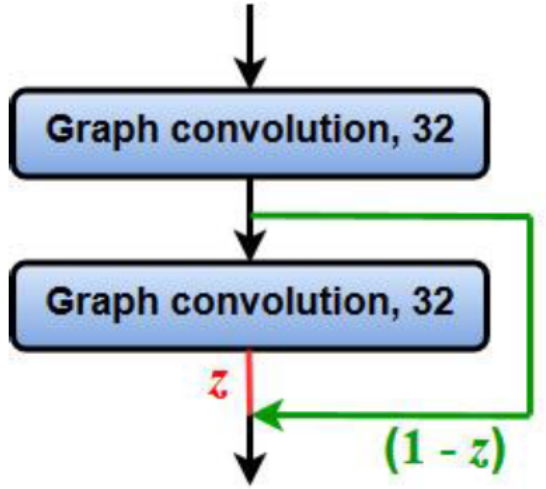
Gated Skip Connection

However, naïve skip-connection unintentionally mix the information.



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

Instead, one may use a gated-skip connection, which mixes the information with appropriate ratio, z.



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

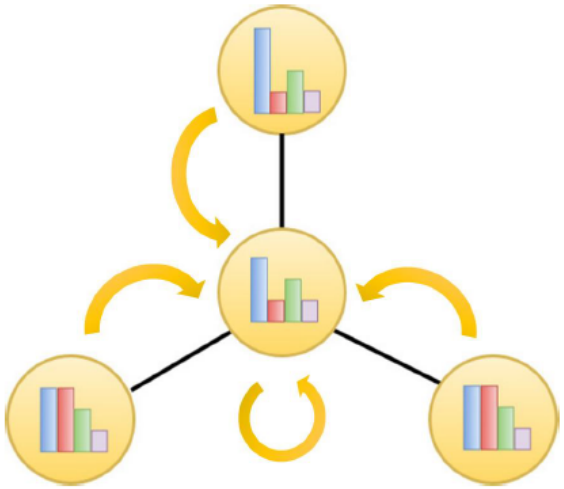
$$z_i = \sigma\left(U_{z,1} H_i^{(l+1)} + U_{z,2} H_i^{(l)} + b_z\right)$$

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).

KAIST

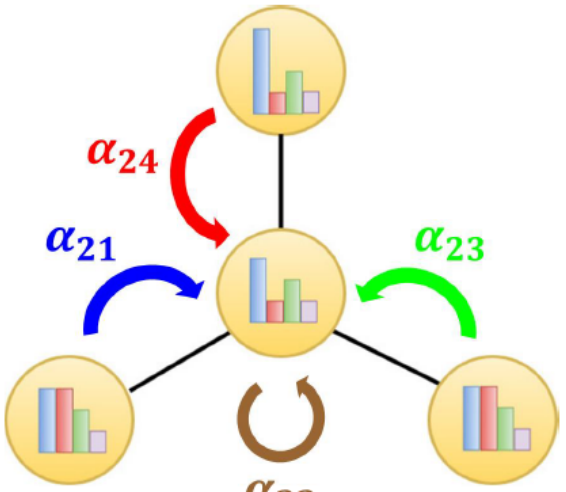
Attention

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



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

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



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

KAIST

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).

Attention

Learnable parameters : Convolution weight and attention coefficient

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

- Velickovic, Petar, et al. – network analysis

$$\alpha_{ij} = softmax(e_{ij}) = \frac{e_{ij}}{exp\left(\sum_{k \in N(i)} e_{ik}\right)} \qquad e_{ij} = LearkyReLU(a^T[H_i W, H_j W])$$

Velickovic, Petar, et al.
"Graph attention networks." *arXiv preprint arXiv:1710.10903* (2017).

- Seongok Ryu, et al. – molecular applications

$$\alpha_{ij} = \tanh\left((H_i W)^T C(H_j W)\right)$$

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).

KAIST