Multi Hop Feature Quality Estimation For Unsupervised Graph Representation Learning

Capstone Project ⇒ **User Guide**

25-2-R-10

Submitters:
Gad Azriel
Lidor Kupershmid

Supervisors:
Dr. Renata Avros
Prof. Zeev Volkovich

Github Link

2025

Table of Contents

Table of Contents

User Guide

MQE with GraphSAGE for Robust Graph Representation Learning

1. Project Guide

This project evaluates robust graph representation learning under noisy node features.

It compares two approaches:

- **Original MQE** ⇒ matrix-based multi-hop feature propagation
- **MQE + GraphSAGE** ⇒ GraphSAGE-based neighborhood aggregation combined with quality estimation

The goal is to show that integrating **Graph Neural Networks (GraphSAGE)** into the MQE framework improves:

- Classification accuracy
- Robustness to feature noise
- Runtime efficiency

2. How to Run the Project

2.1 Environment

- The project is designed to run on Google Colab
- GPU is recommended but not mandatory (for better run time)
- No local installation is required

2.2 Opening the Project

1. Open the provided notebook (MQE_GraphSAGE.ipynb) in Google Colab
2. Make sure the runtime is set to:
   - ⇒ Runtime → Change runtime type → GPU

2.3 Dependency Installation

At the beginning of the notebook, all required libraries are installed automatically, including:

- PyTorch
- PyTorch Geometric
- scikit-learn

2.4 Project Execution Flow

The project runs in the following order:

1. **Environment setup & dependency installation**
2. **Project structure creation**
   - ❖ Helper modules (utils.py)
   - ❖ Models (model_graphsage.py)
   - ❖ Training logic (train_graphsage.py)
3. **Experimental comparison**
   - ❖ Executed via run_comparison.py

2. **Results generation**
   - ❖ Accuracy, runtime, and logs printed to the output
3. **Visualization (optional)**
   - ❖ Plots generated using visualization.py

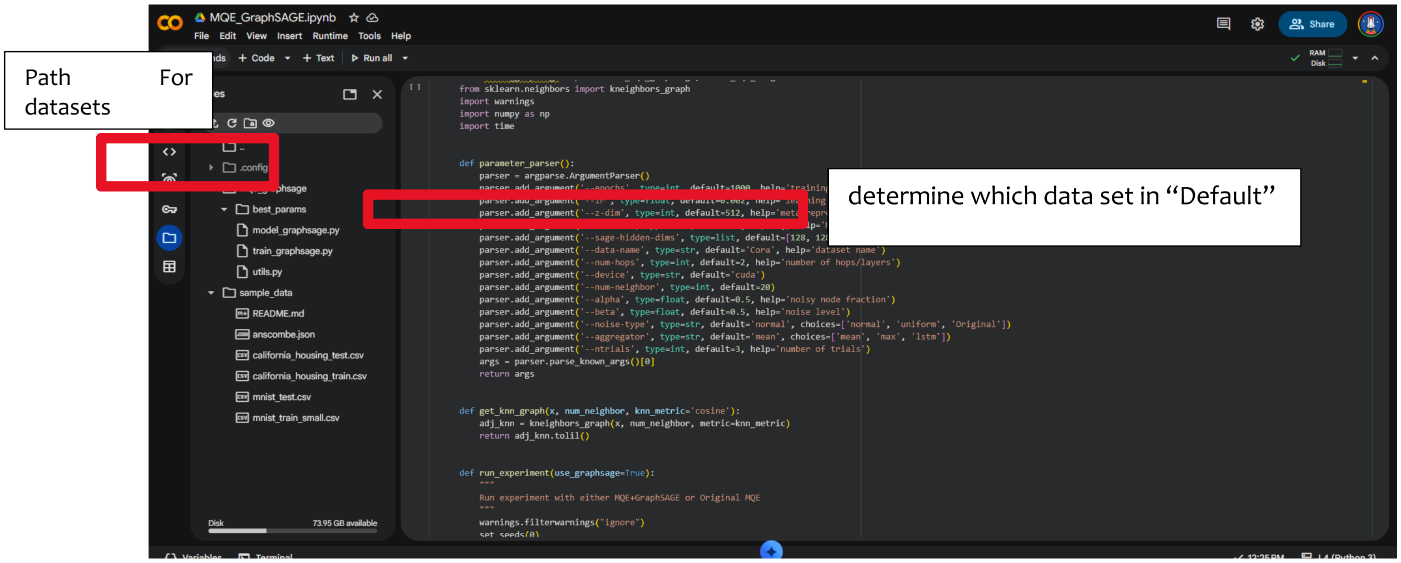
3. To run the full experiment:

- !python run_comparison.py
- No manual interaction is required during execution.

3.1 Input Data (Updated: MQE GitHub Dataset and Hyperparameters)

Important! Please download all data sets and drag to mqe_graphsage/best_params folder created in colab.

Determine which data set you would like to test the code on in def_parameterparser(), valid options after adding the files are: Cora, CiteCeer, Pubmed, computers, photo.



```
from sklearn.neighbors import kneighbors_graph
import warnings
import numpy as np
import time


def parameter_parser():
    parser = argparse.ArgumentParser()
    parser.add_argument('--epochs', type=int, default=1000, help='training
    parser.add_argument('--lr', type=float, default=0.002, help='learning
    parser.add_argument('--z-dim', type=int, default=512, help='meta repre
                                                                    lp='
    parser.add_argument('--sage-hidden-dims', type=list, default=[128, 128
    parser.add_argument('--data-name', type=str, default='Cora', help='dataset name')
    parser.add_argument('--num-hops', type=int, default=2, help='number of hops/layers')
    parser.add_argument('--device', type=str, default='cuda')
    parser.add_argument('--num-neighbor', type=int, default=20)
    parser.add_argument('--alpha', type=float, default=0.5, help='noisy node fraction')
    parser.add_argument('--beta', type=float, default=0.5, help='noise level')
    parser.add_argument('--noise-type', type=str, default='normal', choices=['normal', 'uniform', 'Original'])
    parser.add_argument('--aggregator', type=str, default='mean', choices=['mean', 'max', 'lstm'])
    parser.add_argument('--ntrials', type=int, default=3, help='number of trials')
    args = parser.parse_known_args()[0]
    return args


def get_knn_graph(x, num_neighbor, knn_metric='cosine'):
    adj_knn = kneighbors_graph(x, num_neighbor, metric=knn_metric)
    return adj_knn.tolil()


def run_experiment(use_graphsage=True):
    """
    Run experiment with either MQE+GraphSAGE or Original MQE
    """
    warnings.filterwarnings("ignore")
    set_seeds(0)
```

All core datasets and experimental configurations in this project follow the original MQE implementation provided by the research authors. You can download them from the official GitHub repository, MQE official GitHub ⇒ https://github.com/Shiy-Li/MQE

This repository includes:

- Dataset *download scripts* (which automatically fetch benchmark graph datasets such as Cora, CiteSeer, PubMed)
- *Standard dataset hyperparameters* used in the original MQE experiments
- Tools to prepare and pre-process the data into the folder structure expected by the MQE code

3.2 Output Results

During execution, the following outputs are produced:

- Console Outputs
- Training loss per epoch
- Accuracy per trial
- Average accuracy and runtime summary

4. Output (Cora Dataset):

There are many prints of epochs, trial etc.
The most important is the following comparison to assess the theorem:

Comparing Runtime, Accuracy to prove our concept.

```
============================================================

FINAL RESULTS SUMMARY

============================================================

Method                Accuracy      Time (s)

------------------------------------------------------------

Original MQE            68.64±0.59%      7.7

MQE + GraphSAGE             79.4±0.45%      5.3

============================================================
```

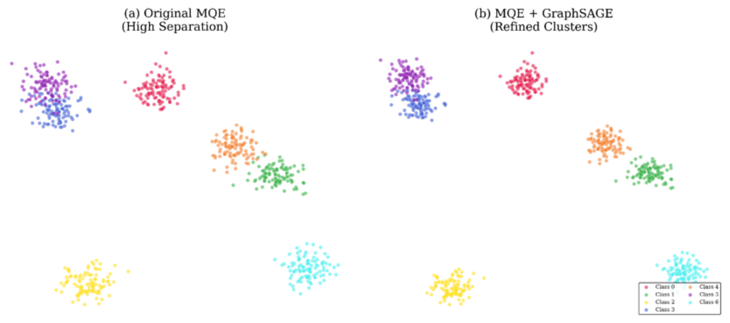


Fig 1. *Clustering Comparison*

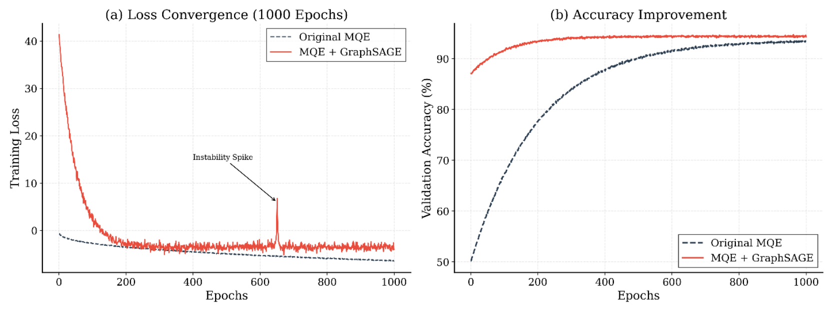


Fig 2. *Accuracy and Loss Convergence across 1000 epochs between both algorithms.*

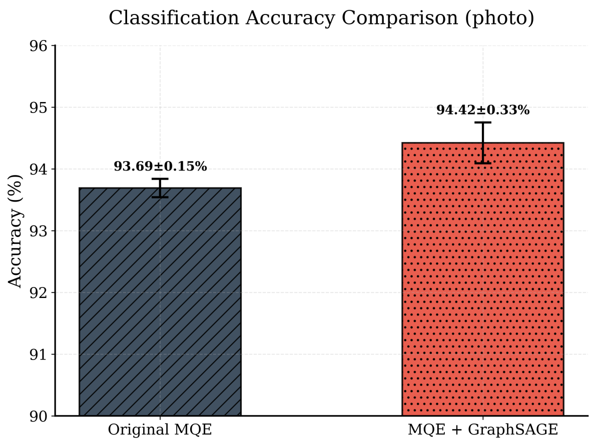


Fig 3. *Classification Accuracy Comparison between MQE and our MQE+GraphSAGE optimized algorithm*

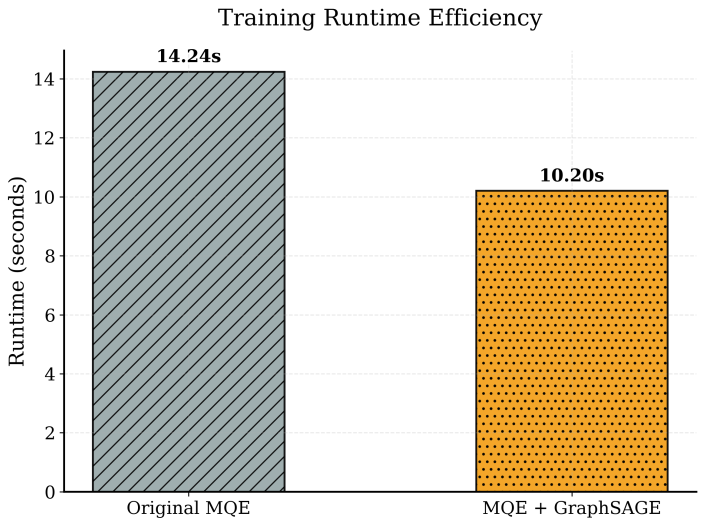


Fig4. *Runtime Comparison of Orignial MQE and our MQE+ GraphSAGE suggested algorithm*