Assignment 2

Tingjun Yuan

***Abstract*—This assignment is mainly based on the knowledge of Attention Mechanisms and Graph Neural Networks (GNN). The dataset is generally based on a pedestrian trajectory recorded in a mall. The models trained in the assignment predict the positions of the pedestrians. This assignment first trains a GNN model according to a tutorial, but reimplemented with PyTorch. After training, the model is evaluated with the main squared error (MSE), the mean Euclidean distance, and the plot graphs. Next, the model is reexamined by tuning hyperparameters, trying a deeper embedding, and replacing the learned attention mechanism. This report shows the results of the model evaluation.**

I. Result and Evaluation

Task 1

This task preprocesses the data, defines the classes for layers and training logic, trains the model, and evaluates it by calculating the mean squared error (MSE) and visualizing the differences between prediction and real future positions. This step follows the tutorial [1] but rewrites the code using PyTorch, another deep learning library which is more compatible with the experiment environment.

The hyperparameters for the training is as follows:

```
hidden_units=100,
num_heads=8,
num_layers=3,
output_dim=2,
num_epochs=100,
learning_rate=1e-3,
```

After training the model for 100 epochs, the training MSE is **0.4871**, and the test mean Euclidean distance is **0.2455**. In addition, one of the scenes is randomly chosen from the test set to perform evaluation by visualizing the coordinates. The result is shown in Figure 1.

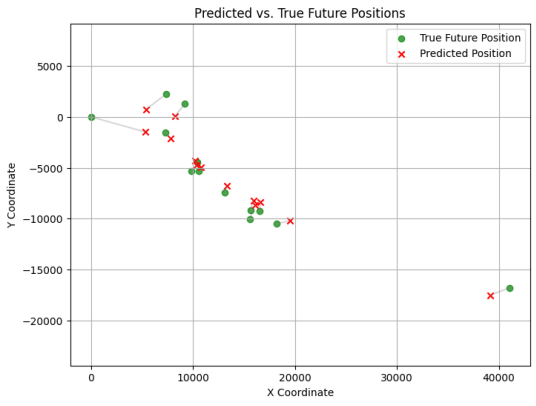

Fig. 1. Visualization on the Original Model

Task 2

This task consists of two parts: performing hyperparameter tuning of the number of attention heads, and trying a deeper embedding of the node features. Compared with the code provided in Task 1, these parts tried different numbers of heads: one of them is `num_heads=32` (greater than the original), the other is `num_heads=4` (less than the original). The result is shown in Table I, the visualization that uses the same scene described above but applies for `num_heads=16` is shown in Figure 2. More figures can be seen in the Notebook.

TABLE I
Evaluation Result of Different Numbers of Heads

Num of Heads	Train MSE	Test Mean Euclidean Dist
4	0.2523	0.2159
8*	0.4871	0.2455
16	0.5456	0.1362
32	0.8346	0.1375

* Already done in Task 1, repeated here for comparasion

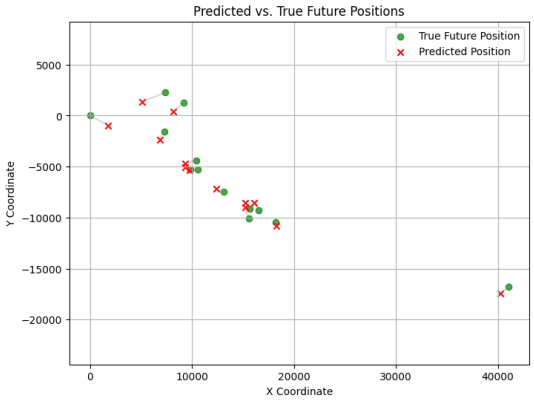

Fig. 2. Visualization with num_heads = 16

In addition to this task, the `GraphAttention` class is also modified so that it uses multi-layer perceptron (MLP) with `ReLU` activation. This generally improves the quality of the model. After modifying the model structure, the model is retrained with the same hyperparameters as used in Task 1 (i.e. `num_heads=8`). The training MSE becomes **0.1949**, and the test mean Euclidean distance is **0.1436**. The plot of the positions is shown in Figure 3.

Task 3

This task replaces the learned attention mechanism with an attention mechanism based on the Cosine similarity between

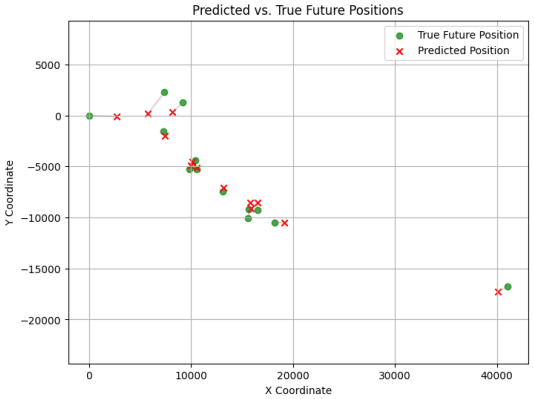

Fig. 3. Visualization with MLP including ReLU

node vectors. The modification is based on the end of Task 2. After training the modified model with the same hyperparameters as those in Task 1, the training MSE becomes **0.1885**, and the test mean Euclidean distance becomes **0.1078**. The visualization for the scene that applies this graph is shown in Figure 4.

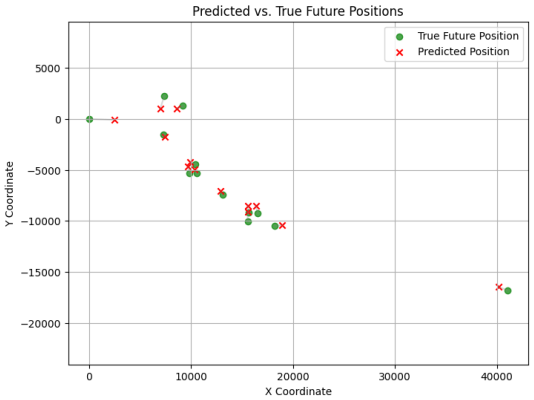

Fig. 4. Visualization with Cosine similarity-based attention mechanism

II. Discussion and Conclusion

As shown in the results, we can find that there has many limitations on the naïve single-layer perception with linear activation, even when I tuned the number of heads. However, with the MLP with `ReLU` activation, even with the same hyperparameters, the performance of the model is far more better.

To make our model more efficient and accurate, the learned attention mechanism of the model is consequently replaced with the Cosine similarity-based attention. This makes the model more accurate.

In conclusion,

References

[1] A. Kensert, "Graph attention network (GAT) for node classification," 2021, Keras. [Online]. Available: https://keras.io/examples/graph/gat_node_classification/

[2] S. Jaiswal, "Multilayer Perceptrons in Machine Learning: A Comprehensive Guide," 2025, Datacamp. [Online]. Available: https://www.datacamp.com/tutorial/multilayer-perceptrons-in-machine-learning

[3] P. Miesle, "What is Cosine Similarity: A Comprehensive Guide," 2025, DataStax. [Online]. Avaliable: https://www.datastax.com/guides/what-is-cosine-similarity