Graph neural networks for traffic modeling
Project Laboratory 1.

Benedek Máth, Ádám Nemes
Supervisor: Dr. Varga Balázs

***Abstract*—The present study experimentally investigated the efficiency of Graph Neural Networks (GNNs) in the context of traffic flow prediction. We generated a dataset comprising 1,358 data points using a microscopic traffic simulation tool - SUMO on the traffic network of Győr. Our focus was on car counting data, with partial coverage of the graph's edges. The Graph Neural Network was trained to estimate the values on the covered edges using data from the remaining edges, simulating the presence of car-counting sensors (e.g.:inductive loop detectors, cameras) in a real-world scenario. We used graph convolutional layers to effectively capture the structure of the graph in our model, allowing us to make accurate predictions based on the connections within the traffic network. We successfully trained the relatively simple model to estimate the traffic volume on the covered edges based on the data from the other edges.**

I. Introduction

Graph Neural Networks are becoming increasingly popular across several fields because they are better at capturing certain features of data structures described by graphs than the usual dense or even convolutional networks. Such fields involve anything where graphs are a good choice of describing a phenomena for example: chemistry, [1] for instance, one might want to classify the role of a protein within a biological interaction graph, and physics [8] analyzes interactions in a physical system, social sciences [6] predict an individual's role in a collaboration network, and recommend new friends to a user in a social network. Our aim is to analyse traffic modelling with graph neural network and determine if a graph neural network can capture the flow of traffic from a limited amount of information, in our case a partition of the roads traffic density.

II. Introduction to SUMO

Sumo, short for "Simulation of Urban Mobility," is an open-source traffic simulation package designed to handle large road networks. Developed by the German Aerospace Center (DLR), Sumo is widely used in both academic research and industry to simulate and analyze various transportation scenarios. It provides a comprehensive set of tools to model the behavior of vehicles, pedestrians, and public transportation systems within urban environments.

Sumo has many positive features: It is Open-source and extensible, it is a Multi-Modal traffic simulation thus it support many forms of transport such as cars, bicyles, buses, and pedestrians. Sumo is scalable thus it can handle complex road networks and hundreds of cars. Sumo includes tools for visualizing vehicle movement and traffic flow in 2D.

Sumo provides a wide range of data output options, allowing users to perform in-depth analysis of traffic dynamics and performance metrics. This detailed data collection capability is essential for understanding traffic patterns, evaluating the impact of traffic management strategies, and conducting various transportation research studies.

Sumo creates a traffic road map with road junctions traffic lights that can be set by the user. SUMO is a microscopic traffic simulator, meaning that each vehicle is modeled as individual agents with their own -simplified- dynamics, called the car following model. The movements of vehicles on a lane is continous in space and discrete in time. Then when the simulation start it slowly fills the roads with vehicles up to the set traffic scale. SUMO simulates every car individually with a route to go along obeying traffic laws, and saves numerous data from the simulation such as emission data, incident data, vehicle-specific data for example speed and acceleration, travel times, and traffic flow data. To train our model we used vehicle count data, specifically the count of cars that left a road segment.

We used the road map and traffic light system of Győr to train our model. The required files to create the road map were provided by our project laboratory supervisor. One simulation generates data for a four hour long traffic flow. We set the period of car counting to fifteen minutes thus creating sixteen data point with one run. The data from the initial two time periods were excluded from our analysis. Since in the first half an hour the simulation loads the road network with cars, we are only interested in the states where the traffic flow is relatively stable.

Fig. 1. A section of Győr's map in sumo simulation.

To create the training data we ran several simulations on Győr's roadmap with multiple modifications to acquire a diverse training data set. We mainly modified the global seed of the simulation and the traffic's scale. On Figure 1. Győr's part around coordinates $47.688059, 17.644442$ is shown.

In SUMO seeds are used to initialize the random number generators that control various stochastic processes within the simulation e.g.: veichle insertion, driver imperfection. Global seed is the primary seed that influences all random processes within the simulation. By setting the global seed, users ensure that the same sequence of random events occurs each time the simulation is run, which is crucial for reproducibility. In our case it was necessary to change the seed in every run to obtain a variety of data. We used 25 different traffic seed from 38-62.

The traffic scale parameter uniformly scales vehicle inflows to the network. As we set a traffic scale higher the number of cars are increasing in the simulations, more traffic jams occur the speed of the movement slows down.

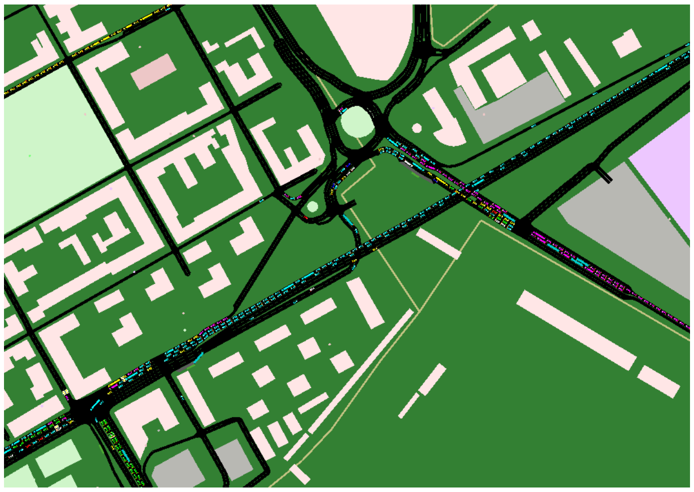

Fig. 2. Sumo simulation with 1.8 traffic scale.

As depicted in Figure 2, a notable increase in traffic volume is observed on the road network simulated using Sumo, leading to significant congestion patterns. We used four distinct traffic scale parameters to generate our data $\{1, 1.1, 1.2, 1.4\}$ thus we obtained in total a hundred different simulations with four types of traffic cale and 25 global seeds. [5]

III. Introduction to Graph Neural Networks

Graph Neural Networks(GNN) are a type of neural network designed to perform inference on data described by graphs. GNN's input are X the matrix of node feature vectors X_i. $G = (V, E)$ with $v_i \in V$ edges, N nodes, $(v_i, v_j) \in E$, an adjacency matrix $A \in \mathbb{R}^{N\times N}$.

In our case the nodes of the graph are road intersection and the edges are roads connecting them, also there are longer roads that we broke into multiple sections of edges and nodes. GNNs are using the adjacency matrix of the graph to capture the properties of the system. Nodes that are connected by edges and edges connected by nodes have greater impact on each other than those which are not. GNNs are using graph convulotional layers to pass information through the graph. In the first layer the edge/node gets information from its direct neighborhood that what we call it's first neighbor, in the second layer it gets information from the first neighbors' direct neighborhood and so on. From k'th layer it will get information from edges/nodes which can reach it in exactly k steps.

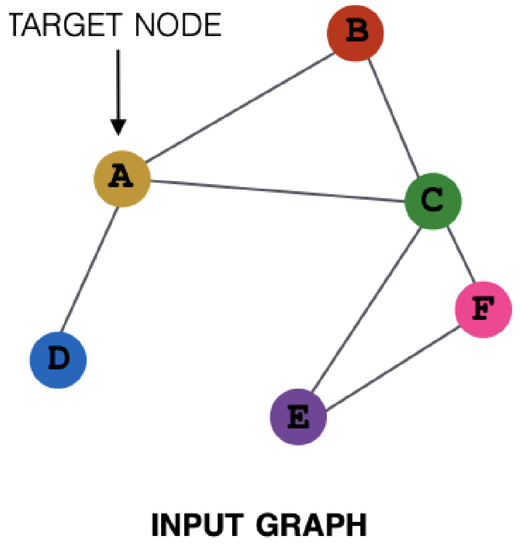

Fig. 3. Example graph

In this case shown in Figure 3 the target node A gets information from nodes: {B, C, D} in the first layer, in the second layer from B, C, D nodes neigborhoods: from {A (itself), C} these are the neighbors of B, from {A, B, E, F} C's neighbors and from the only neighbor of D {A}. This information passing method can be seen on Figure 4

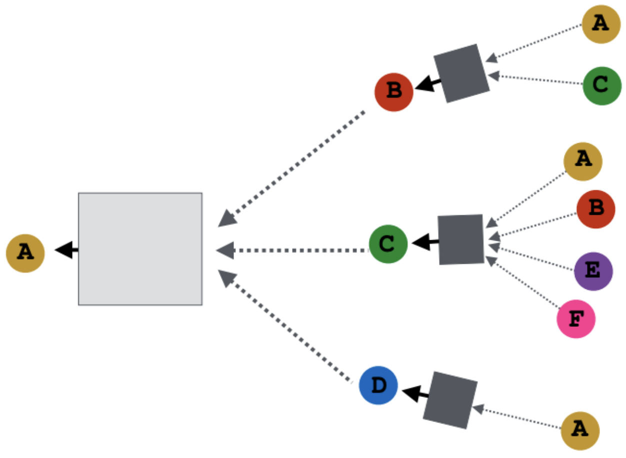

Fig. 4. Example graph

We used a multi-layer Graph Convolutional Network (GCN) layers with the following layer-wise propagation rule:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right). \quad (1)$$

Here,I_N is the identity matrix, $\tilde{A} = A + I_N$ is the adjacency matrix of the undirected graph G with added self-connections. $\tilde{D}$ is the degree matrix defined as $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. $\sigma(\cdot)$ denotes an activation function, such as ReLU ($\mathrm{ReLU}(x) = \max(0, x)$), and $W^{(l)}$ is a layer-specific trainable weight matrix. $H^{(l)} \in \mathbb{R}^{N\times D}$ is the matrix of activations in the l-th layer; $H^{(0)} = X$. [4]

In our project we used Adam optimiser. Adam is a popular optimizer in deep learning due to its adaptive learning rate approach, which helps in faster convergence and better handling of complex models and datasets compared to simpler optimization methods. It achieves this by maintaining per-parameter learning rates that are dynamically adjusted based on the first and second moments of the gradients. This adaptive behavior is particularly beneficial for models with high-dimensional parameter spaces or non-linear activation functions, where traditional SGD may struggle to find optimal solutions efficiently. We set the learning rate to 0.005, which is dictating the magnitude of parameter updates at each iteration. [2]

IV. Modelling traffic

A. The model

Our goal for this semester was to create a graph neural network that is capable of estimating the traffic volume on certain roads, that do not have sensors to measure it, based on the data measured on the known roads that have sensors on them. In our work, the traffic volume is defined as the number of cars that have left the given road.

We decided to have 70% of roads have unknown traffic volume, and the remaining 30% known. The motivation for this was that in real world cases not more than 10% of roads are covered with sensors, however, aiming for that percentage and successfully teaching a model with that little amount of data was out of reach for us in just one semester, therefore we settled for the percentages written above. We selected the known roads randomly, with uniform distribution. In order to make sure, that the selection isn't concentrated on one part of the city, we plotted the selected roads, these are on Figure5. Selecting the roads this way is not the most scientific or optimal way, and one of our goals for next semester is to work on this, and find a better strategy for choosing the known roads.

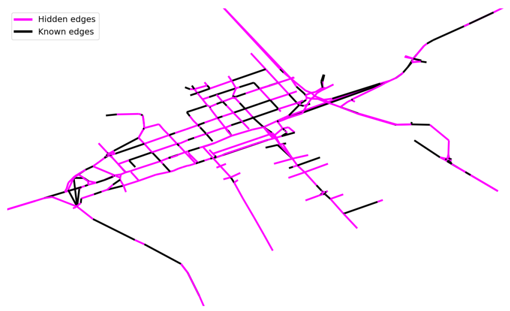

Fig. 5. In this graph the sensorless roads -these have unknown traffic volume- are coloured magenta, the roads with sensors are coloured black.

Our model is fairly simple, we used 8 Graph Convolutional Network Layer (GCN), and a dense layer. The latter is the one, that gives the final output, this is common practice. The reason for the number of GCN layers, is that we had to make sure, that every unknown road is within the reach of at least one known road, otherwise, we couldn't make a prediction on the given road. Since the edge values cannot be negative, after each GCN layer we used a ReLu activation function on the output. One interesting technical detail we ran into was, that if we put a ReLu function on the output of the final, dense layer, the model fails to learn. Therefore, we had to put the ReLu function by hand on the output of the model. Obviously, we can work on a more complex model, including normalization and pooling layers, and our other goal for next semester is to experiment with more complex models. [7]

B. Training of the model

In the training of the model we used the data generated using SUMO. In this subsection we will call data points the members of the generated data set, that describes what was the traffic volume on a given road, in the last 15 minutes. For the training of the model, we used all 1358 data points. During the data preparation, we normed all the traffic volumes with their maximum, as neural networks generally work better if the data is between certain bounds, in our case these are numbers between 0 and 1. Admittedly, for first glance this does seem a somewhat questionable practice, as we can not guarantee, that the road with the highest traffic volume is a known road. However with a better choosing strategy -which we will do next semester-, we can make sure, that the busiest road is among the known roads, therefore this assumption is valid even now, also engineering judgement can be made to find the normalizing factor too. As a rule of thumb an arterial road (with no traffic lights) has a peak capacity of 1800veh/hour/lane. We wanted to make a clear distinction between the known and unknown roads. We had 3 options for this:

1) Set the traffic volume of the unknown roads to some non-negative number
2) Set the the traffic volume of the unknown roads to NaN
3) Set the traffic volume of the known roads to some negative number

The first option does not work, as the positive numbers have real meaning for our problem. The second option also does not work, as the GCN does not work well with NaN values, therefore we settled with option 3, and set the traffic volumes (edge attributes) to -1.

For the training of the model we created a training dataset, where we hid the traffic volumes of the unknown roads in the way written above, and a corresponding validation dataset, where we preserved all the edge features. Then we generated an output based on the input coming from the training dataset, and compared it with the corresponding element of the validation dataset. It is important to note, that the training and validation data should come from the same data point, as in our current goal it does not make sense to make an estimation for time t_2 based on the observations of t_1. We used Mean Squared Error as loss function and backpropagation for correcting the model.

For the training in order to avoid overfitting for a given datapoint we decided to shuffle the data. Our first attempt was that we selected a data point randomly, ran the training cycle for some epochs, and then select a new one. The loss-epoch graph for this method is on Figure 6. It is obviously not a good method, as the model is overfitting for each data point, and fails to learn some general characteristics of the problem.

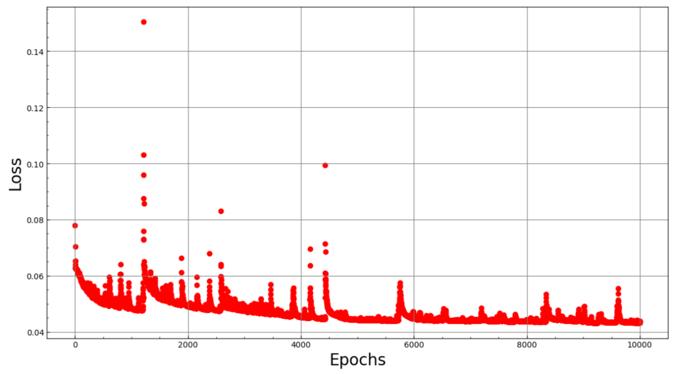

Fig. 6. Loss-epoch graph for the first attempt on teaching the model. It is quite obvious, that the model does not learn something general about the problem, instead it overfits for a datapoint throughout the epochs, and once a new data point is selected it starts again. In this case the number of epochs for which we did not change the training data point was 100.

In order to teach the model in a better way, we decided that for every epoch, we chose the teaching data point randomly. In this case we avoided overfitting, however we did not get a significantly better result after many epochs. The loss-epoch graph is on Figure 7. In order to determine our model's performance and to see whether it learns or not, we used a benchmark. Our benchmark was the average loss that we would get, if the model does not work, and produces only zeros as output. This means that our benchmark is essentially the mean squared average of the 1358 data points we are working with. We can see that on average, our model produced 50% decrease compared to this benchmark, however the losses vary a lot, depending on which data point we chose for the given epoch.

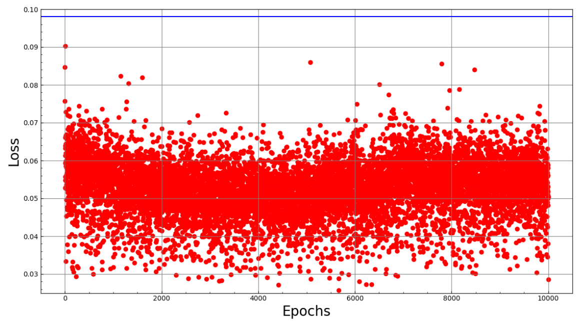

Fig. 7. Loss-epoch graph for the second attempt to teach the model. On the graph the red dots are the loss function's values comparing the validation data points with the output of the model. The blue line shows what the average loss would be, if the model's output are only zeros. We used this as benchmark to determine our model's performance. The model's performance does not get significantly better as we run more epochs on it, in fact, it reaches its minimum after around 4000 epochs, and then it overfits in a way.

In order to capture the average performance better, we calculated after every 100 epoch the average Mean Squared Error for every data point. In order to do this, we created the output of the model for every data point, calculated the mean squared error for them, and calculated an average of that. This is on Figure 8. On this graph we can see more clearly, that our model's performance is around 50% better than the benchmark, and also, after approximately 4000 epochs, the performance gets worse, the total loss grows. One of the potential goals for next semester is enhancing the performance, and achieving smaller losses. A more complex model or implementing a dynamic learning rate for the teaching could be the solution for this problem. In the future, it could also be interesting to compare our model's outputs with other benchmarks.

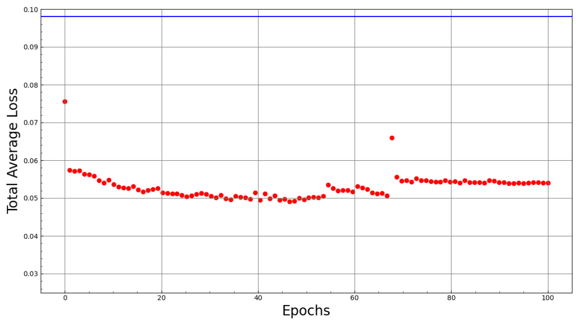

Fig. 8. The average mean squared error-epoch graph for the teaching. The blue line is again showing what the average loss would be, if the model's output are only zeros.

V. RESULTS

In this section we delve deeper into the investigation of the performance of our model. This is justified mainly by the variance of losses in Figure 7. and the usage of 4 different scaling of traffic during the SUMO simulations. Also, we are interested in seeing if there is a road dependency on the accuracy of the model's output.

In order to determine if there are differences between the different scalings, we chose 4 data points, one from each scalings, and created a histogram with the absolute value of the difference between the model's output and the simulated validation data. The histogram is on Figure 9. There are only slight differences between the different scalings, such as the bigger differences are more frequent with scaling 1.4, which makes sense, since in this case a traffic jam is more probable. It is interesting to note, that the model performs the best with scaling 1.1. Acknowledging these differences, the main characteristics are similar, which is something we wanted to see, because this means, the model managed to learn some general nature of traffic volume.

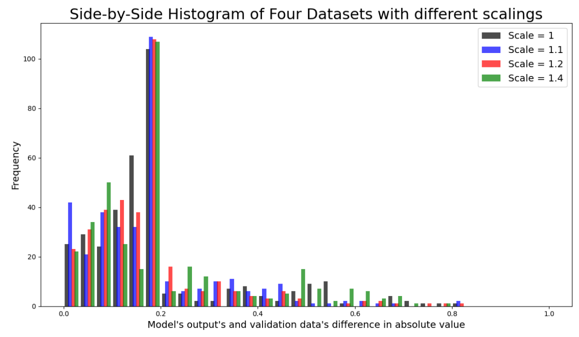

Fig. 9. Histogram with the absolute value of the difference between the model's output and the simulated validation data. As we can see, in this case there is not too much of a difference between the frequencies, the main characteristics are similar, however in the 1.4 scale the frequencies of the larger differences are higher, and the model performs the best with scaling 1.1.

We also wanted to examine what are the spatial nature of the model's output's accuracy. In order to do this, using the same datapoints that we used in Figure 9. we coloured the graph of Győr based on the absolute value of the differences between the model's output and the validation data. The graphs are on Figures 10-13. We can see that the main roads of the city have larger differences, whereas the model managed to learn the smaller streets fairly well. This is a very important result for one of our future goals, which is creating a strategy for placing the sensors in a way that results in a better performance, because this means that instead of a uniform distribution between streets, we should focus on the busier, larger roads.

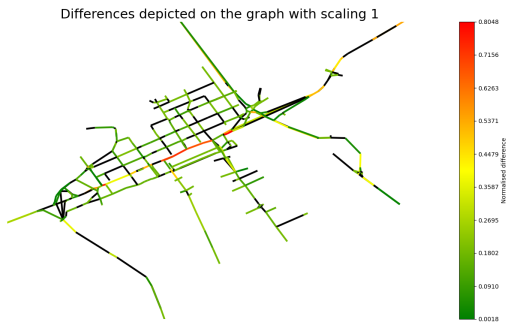

Fig. 10. Coloured graph of Győr based on the the absolute value of the differences between the model's output and the validation data with scaling 1

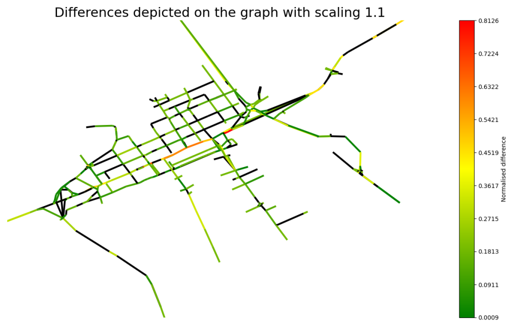

Fig. 11. Coloured graph of Győr based on the the absolute value of the differences between the model's output and the validation data with scaling 1.1

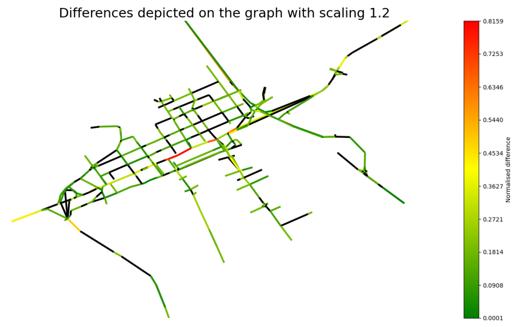

Fig. 12. Coloured graph of Győr based on the the absolute value of the differences between the model's output and the validation data with scaling 1.2

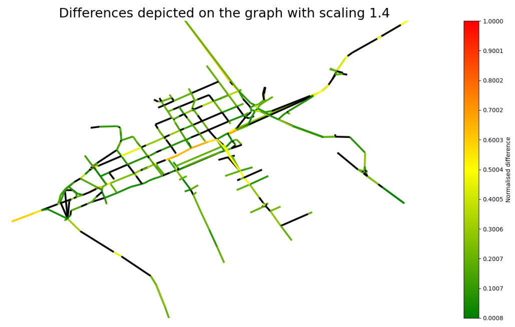

Fig. 13. Coloured graph of Győr based on the the absolute value of the differences between the model's output and the validation data with scaling 1.4

We also used another metric to take a look at the spatial accuracy of the model's output. This is GEH statistics which is used in traffic modelling, traffic forecasting and traffic engineering to compare two sets of traffic volumes. The formula of the metric :

$$\text{GEH} = \sqrt{\frac{2(M-C)^2}{M+C}} \tag{2}$$

Where in our equation M is the quarter hourly traffic volume from the model and C is the volume from the simulation. In this statistics the roads with higher traffic volume will be more sensitive the error term will be scaled with the "importance" of the road. The values in general should be somewhere between 3 and 10, ideally under 5.

These weighted errors are shown on graphs on Figure 14.-17. These graphs strengthens the results from the previous figures, that the model managed to capture the smaller roads' traffic volume fairly well, but lacks on the larger ones. It is quite obvious, that for the main roads the GEH value does not get under 5, in some cases it is also larger than 10, however for the side roads usually does get better than that. [3]

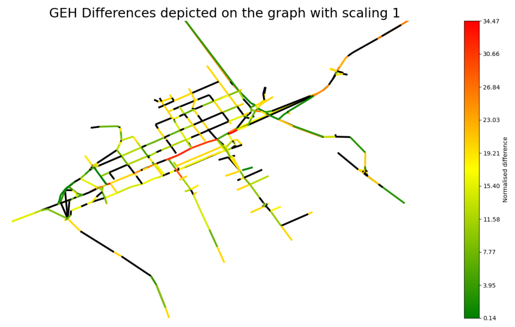

Fig. 14. Coloured graph of Győr based on the the GEH values calculated from the model's output and the validation data with scaling 1

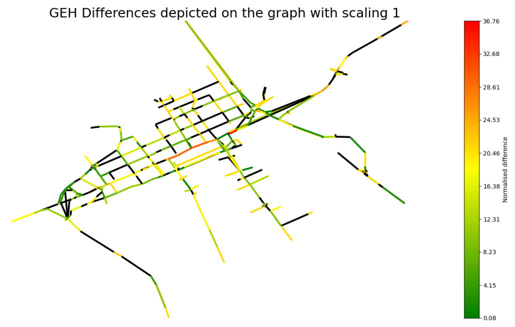

Fig. 15. Coloured graph of Győr based on the the GEH values calculated from the model's output and the validation data with scaling 1.1

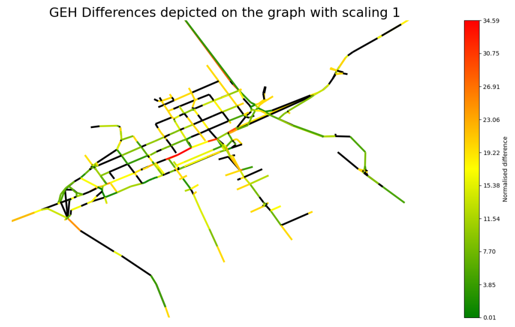

Fig. 16. Coloured graph of Győr based on the the GEH values calculated from the model's output and the validation data with scaling 1.2

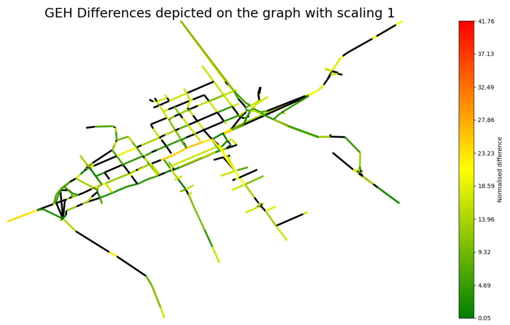

Fig. 17. Coloured graph of Győr based on the the GEH values calculated from the model's output and the validation data with scaling 1.4

VI. Summary and future plans

We successfully created and trained a Graph Neural Network using simulated data from SUMO and proved that despite the simplicity of the model, it works fairly well, it produces results that are significantly better than the benchmark we introduced. Also, we managed to acquire important technical skills such as using pytorch, working with Graph Neural Networks, and using SUMO for traffic simulations, which will help us in the next semester to work on the future plans. In this document we presented our main results, further information and the code we used can be seen at the GIT repository.

Our future plans involve creating a more complex model to achieve better results, working out an advanced strategy for choosing roads to gather data from, and possibly introducing traffic forecasting, in which case we will attempt to forecast the traffic volume at time t_2 based on the data at time t_1, $t_2 > t_1$. Additionally, we plan to conduct a comprehensive sensitivity analysis to understand the impact of various parameters on our model's performance and robustness.

References

[1] Milad Besharatifard and Fatemeh Vafaee. A review on graph neural networks for predicting synergistic drug combinations. *Artificial Intelligence Review*, 57, 02 2024.

[2] Dami Choi, Christopher J. Shallue, Zachary Nado, Jaehoon Lee, Chris J. Maddison, and George E. Dahl. On empirical comparisons of optimizers for deep learning, 2020.

[3] Olga Feldman. The geh measure and quality of the highway assignment models. 10 2012.

[4] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv e-prints*, page arXiv:1609.02907, September 2016.

[5] Daniel Krajzewicz. *Traffic Simulation with SUMO – Simulation of Urban Mobility*, pages 269–293. Springer New York, New York, NY, 2010.

[6] Xiao Li, Li Sun, Mengjie Ling, and Yan Peng. A survey of graph neural network based recommendation in social networks. *Neurocomputing*, 549:126441, 2023.

[7] Amit Roy, Kashob Kumar Roy, Amin Ahsan Ali, M Ashraful Amin, and A K M Mahbubur Rahman. Sst-gnn: Simplified spatio-temporal traffic forecasting model using graph neural network, 2021.

[8] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, pages 8459–8468. PMLR, 2020.