CHALMERS
UNIVERSITY OF TECHNOLOGY

Graph Neural Networks for Traffic Flow Prediction

A Physics-Informed Prediction of Equilibrium Traffic Redistribution under Network Reconfiguration without OD Demand

Master's thesis in Data Science and AI

Xinwei Wu

Department of Electrical Engineering

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

Graph Neural Networks for Traffic Flow Prediction

A Physics-Informed Prediction of Equilibrium Traffic Redistribution under Network Reconfiguration without OD Demand

Xinwei Wu

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2026

Graph Neural Networks for Traffic Flow Prediction
A Physics-Informed Prediction of Equilibrium Traffic Redistribution under Network Reconfiguration without OD Demand
Xinwei Wu

Supervisor: Filip Rydin, Department of Electrical Engineering
Co-supervisor: Alvin Combrink, Department of Electrical Engineering
Co-supervisor: Sten Elling Tingstad Jacobsen, Department of Electrical Engineering

Examiner: Balazs Kulcsar, Department of Electrical Engineering

Master's thesis 2026
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover: AI-assisted author-generated illustration of equilibrium traffic redistribution under road network reconfiguration.

Typeset in LaTeX
Gothenburg, Sweden 2026

Graph Neural Networks for Traffic Flow Prediction
A Physics-Informed Prediction of Equilibrium Traffic Redistribution under Network Reconfiguration without OD Demand
Xinwei Wu
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Network reconfigurations, including closures, new road segments, capacity changes, and speed changes, can redistribute traffic flows. Predicting this redistribution is important for rapid assessment of infrastructure changes and operational interventions. Conventional Stochastic User Equilibrium (SUE) methods can compute the resulting equilibrium state, but they require Origin-Destination (OD) demand and must be solved again for each reconfigured network. This thesis studies a more constrained setting: predicting post-edit equilibrium flows from the pre-edit network, old flows, and the reconfigured network, without OD demand as input.

A physics-informed graph learning framework, called ST-PINN GatedGCN, is developed for this task. First, edge alignment maps old-flow information onto the edge set of the reconfigured graph. The GatedGCN then learns flow propagation and topology changes through graph message passing. The training objective combines edge-flow regression with flow-conservation regularization. It fits SUE-generated labels while reducing node-level physical inconsistency. Inference is completely OD free as the demand is used only to generate labels.

Experiments use the Sioux Falls and EMA networks, with 10,000 network-pair samples for each network. The model obtains 12.92% WMAPE on Sioux Falls and 7.84% WMAPE on EMA, which is lower than the tested baselines. Retained edges are predicted more accurately than newly added edges, since new edges have no historical flow and may introduce new route choices. Ablations show that old-flow information, residual physical injection, the global message channel, and recurrent pressure update affect error and conservation behavior. Inference is about 197 times faster than SUE on Sioux Falls and about 1896 times faster on EMA.

These results suggest that old flows and graph structure contain useful information for fast scenario screening under network reconfiguration. The remaining errors reflect the uncertainty caused by unobserved OD demand, especially for newly added links.

Keywords: Traffic flow prediction; Network reconfiguration; Graph neural networks; Physics-informed learning; OD-demand-free prediction; Flow conservation.

Acknowledgements

I first would like to thank my supervisor Filip for his guidance, support, and encouragement during the past five months. Whenever my research lost direction, he was able to give me useful ideas and clear guidance. I also learned a great deal from him in the writing process. His advice helped me think more rigorously, build a clearer academic argument, and tell the story of the thesis in a better way.

I would also like to thank my co-supervisor Alvin. In our weekly meetings, he often raised sharp and interesting questions. These discussions helped me look beyond the surface of the results. They also helped me think more carefully about why a method works, why it fails, and what different methods can be used in the project.

I am grateful to my examiner Balazs for his guidance and suggestions. His comments helped me improve the thesis and made this thesis journey more valuable.

Finally, I would like to thank my family and friends for their company and encouragement during this period. I also thank Chalmers for providing technical support. During my two years at Chalmers, I gained solid knowledge, learned a rigorous academic attitude, and met many kind classmates and friends. It was a new and meaningful journey.

Xinwei Wu, Gothenburg, June 2026

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

BPR	Bureau of Public Roads
EMA	Eastern Massachusetts Network
FFN	Feed-Forward Network
GatedGCN	Gated Graph Convolutional Network
GNN	Graph Neural Network
LHS	Latin Hypercube Sampling
LWR	Lighthill-Whitham-Richards
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MHA	Multi-Head Attention
MLP	Multi-Layer Perceptron
MSA	Method of Successive Averages
MSE	Mean Squared Error
OD	Origin-Destination
PINN	Physics-Informed Neural Network
RelCon	Relative Conservation Violation
RMSE	Root Mean Squared Error
RMSNorm	Root Mean Square Layer Normalization
STGCN	Spatio-Temporal Graph Convolutional Network
ST-PINN	Spatio-Topological Physics-Informed Neural Network
SUE	Stochastic User Equilibrium
SwiGLU	Swish-Gated Linear Unit
WMAPE	Weighted Mean Absolute Percentage Error

Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables that have been used throughout this thesis.

Indices

e	Indices for directed graph edges
u, v	Indices for graph nodes, intersections, or centroids
k	Index for pseudo-time diffusion step
p	Index for path
r, s	Indices for origin and destination

Sets

$G = (V, E, A)$	Pre-edit road network
$G' = (V, E', A')$	Post-edit road network
V	Shared node set
E, E'	Directed edge sets before and after network reconfiguration
A, A'	Edge-attribute collections before and after network reconfiguration
E_{ret}	Set of retained edges, $E_{\text{ret}} = E \cap E'$
E_{add}	Set of newly added edges, $E_{\text{add}} = E' \setminus E$
$\mathcal{P}_{rs}$	Set of feasible paths between origin r and destination s
$\mathcal{N}(v)$	Neighborhood of node v
$\text{In}_G(v)$	Set of incoming edges to node v in graph G
$\text{Out}_G(v)$	Set of outgoing edges from node v in graph G
S	Set of edges used for metric evaluation, with $S \subseteq E'$

Parameters

K	Number of pseudo-time diffusion steps
d_h	Hidden feature dimension
d_{ff}	Feed-forward network dimension
α, β	BPR congestion-function parameters
t_e^0	Free-flow travel time on edge e
κ_e	Capacity of edge e
λ_{con}	Weight of the flow-conservation regularization term
ϵ	Small positive constant used for numerical stability
μ_f	Mean used for flow normalization
σ_f	Standard deviation or scale used for flow normalization
B'	Directed node-edge incidence operator of the reconfigured graph
ϕ_m	Message function in a graph neural network layer
ϕ_h	Node-update function in a graph neural network layer
ψ	Edge-flow readout or decoder function

Variables

f^{old}	Observed old flows on G
f^{new}	Post-edit flows on G'
f_e	Real-scale target flow on edge e
$\hat{f}_e$	Real-scale predicted flow on edge e
$\tilde{f}_e$	Normalized target flow on edge e
$\hat{\tilde{f}}_e$	Normalized predicted flow on edge e
$\hat{\tilde{f}}_e^{(k)}$	Normalized flow estimate on edge e at pseudo-time step k
$\Delta \hat{\tilde{f}}_e^{(k)}$	Normalized flow correction on edge e at pseudo-time step k
a_e	Edge-attribute vector on the pre-edit graph
a'_e	Edge-attribute vector on the post-edit graph
a_e^{align}	Aligned old-new edge descriptor for edge $e \in E'$
z_e	Aligned edge input feature used by the learning model
d_v	Node-level net-demand proxy derived from old flows
ρ_v	Node-level flow-imbalance residual at node v
Q	OD demand matrix used for SUE label generation
q_{rs}	OD demand from origin r to destination s
δ_{ep}	Indicator equal to one if edge e belongs to path p

$t_e(f_e)$	Congestion-dependent travel cost on edge e
$h_v^{(k)}$	Node hidden state at pseudo-time step k
$g_e^{(k)}$	Edge hidden state at pseudo-time step k
$m_{u \to v}$	Message sent from node u to node v
η_e	Learned edge gate in the GatedGCN layer
$\mathcal{L}_{\text{sup}}$	Supervised edge-flow regression loss
$\mathcal{L}_{\text{con}}$	Flow-conservation regularization loss
$\mathcal{L}$	Total training loss
$\text{RMSE}_{\text{norm}}$	Root mean squared error in normalized space
$\text{RMSE}_{\text{real}}$	Root mean squared error in real flow units
WMAPE	Weighted mean absolute percentage error
RelCon	Relative node-level flow-conservation violation

Contents

List of Acronyms **ix**

Nomenclature **xi**

List of Figures **xix**

List of Tables **xxi**

1 Introduction **1**
1.1 Background . . . 1
1.2 Problem Statement . . . 2
1.3 Aim and Research Questions . . . 3
1.4 Contributions . . . 4
1.5 Thesis Outline . . . 4

2 Related Work on Traffic Assignment and Graph-Based Learning **5**
2.1 Traffic Assignment . . . 5
2.2 Graph Neural Networks . . . 6
2.3 Physics-Informed Learning on Graphs . . . 7
2.4 Research Gap . . . 8

3 Physics-Informed Graph Learning Framework for Flow Prediction **9**
3.1 Ground-Truth Generation through Traffic Assignment . . . 9
3.1.1 Generation of Pre-edit Traffic Scenarios . . . 9
3.1.2 Network Reconfiguration Operators . . . 10
3.1.3 Equilibrium Flow Computation . . . 10
3.1.4 Construction of Network-Pair Samples . . . 12
3.2 Graph-Based Surrogate Model . . . 13
3.2.1 Design Principle . . . 13
3.2.2 Edge Alignment Module . . . 14
3.2.3 GatedGCN-Based Pseudo-Time Diffusion . . . 15
3.2.4 Global Interaction Modeling via Virtual Routing . . . 17
3.3 Structure-Aware Learning Objective . . . 18
3.3.1 Supervised Regression Loss . . . 18
3.3.2 Equilibrium Flow Conservation . . . 18
3.4 Well-Posedness of the OD-Free Flow-to-Flow Problem . . . 20

3.4.1 Observable Inputs and Latent Demand . . . 20
3.4.2 Non-Uniqueness of the Flow-to-Flow Map . . . 20
3.4.3 Statistical Formulation . . . 21
3.4.4 Irreducible Error and Meaning for This Thesis . . . 22

4 Implementation **25**
4.1 Training Setup . . . 25
4.2 Baselines . . . 27
4.2.1 Old-Flow Baseline . . . 27
4.2.2 MLP Baseline . . . 29
4.2.3 Single-Topology GatedGCN Baseline . . . 29
4.2.4 Node-Centric GNN Baseline . . . 29
4.2.5 SUE Solver . . . 30
4.3 Ablation Study Design . . . 30
4.3.1 Removal of Old-Flow Information . . . 30
4.3.2 New-Attribute-Only Input . . . 31
4.3.3 Removal of Physical Residual Injection . . . 31
4.3.4 Removal of Global Message Channel . . . 31
4.3.5 Unshared Diffusion Cells . . . 32
4.3.6 Removal of Recurrent Pressure Update . . . 32

5 Results **33**
5.1 Experimental details . . . 33
5.1.1 Datasets and Network Reconfiguration Scenarios . . . 33
5.1.2 Evaluation Metrics . . . 35
5.2 Overall Prediction Performance . . . 36
5.2.1 Performance on Sioux Falls . . . 36
5.2.2 Performance on EMA Network . . . 37
5.2.3 Performance by Edge Type . . . 39
5.3 Physical Consistency of Predicted Flows . . . 42
5.3.1 Flow-Conservation Violation . . . 43
5.3.2 Accuracy-Conservation Trade-Off . . . 43
5.3.3 Qualitative Examples of Flow Redistribution . . . 45
5.4 Ablation Studies . . . 46
5.4.1 Ablation Setup . . . 46
5.4.2 Overall Ablation Results . . . 46
5.4.3 Impact on Prediction Accuracy and Physical Consistency . . . 48
5.5 Failure Cases . . . 50
5.6 Summary . . . 50

6 Discussion **53**
6.1 Main Findings . . . 53
6.1.1 RQ1: Methodological Gap . . . 53
6.1.2 RQ2: OD-Free Flow-to-Flow Formulation . . . 53
6.1.3 RQ3: Predictive Performance . . . 53
6.1.4 RQ4: Error and Physical Consistency . . . 54
6.2 Implications . . . 54

6.3 Limitations 55
6.4 Future Work 55

Bibliography **57**

List of Figures

3.1 Overview of the ST-PINN GatedGCN surrogate for OD-free traffic flow prediction under network reconfiguration. The model takes the pre-edit network, observed old flows, and the reconfigured network as inputs. Edge alignment transfers old-flow information onto the new edge set, while recurrent physics-informed GatedGCN diffusion updates edge and node states through residual injection, message passing, and flow correction. OD demand is used only by the SUE solver to generate training labels. The final output is the predicted post-edit flow field on E', evaluated by WMAPE, RMSE, and RelCon. 13

4.1 Dataset generation workflow. For each sampled OD demand Q, f^{old} is solved on the pre-edit network G by SUE. The network is then reconfigured, and SUE is solved again on G' with the same Q to obtain the training label f^{new}. The OD demand is used only for label generation and is not part of the model input. 25
4.2 Implemented ST-PINN GatedGCN architecture. Edge alignment maps information from G to the new edge set E'. Retained edges can inherit old-flow information, while newly added edges are marked because they have no old flow. The recurrent GatedGCN diffusion then updates the flow state through residual injection, message passing, and flow correction. 26
4.3 Training and evaluation workflow. Graph-pair samples are split into training, validation, and test sets. Flow values and edge attributes are normalized before training. The model is optimized with supervised edge-flow loss and conservation loss. The final checkpoint is selected on the validation set and then evaluated on the test set using WMAPE, RMSE, RelCon, edge-type errors, and runtime. 27

5.1 Example reconfiguration sample for Sioux Falls. 34
5.2 Example reconfiguration sample for the EMA network. 34
5.3 Overall training loss on Sioux Falls. 37
5.4 Overall validation RMSE on Sioux Falls. 38
5.5 Overall training loss on the EMA network. 39
5.6 Overall validation RMSE on the EMA network. 40
5.7 Relative retained-edge WMAPE reduction of ST-PINN GatedGCN over the strongest baseline. 41

5.8 New-edge prediction penalty of ST-PINN GatedGCN, measured as the ratio between new-edge WMAPE and retained-edge WMAPE. . . 42
5.9 Accuracy-conservation trade-off under different values of λ_{con}. 45
5.10 Qualitative example of flow redistribution on a Sioux Falls test graph. The first two pictures compare observed and predicted flow redistribution after reconfiguration, while the third one shows the absolute prediction error. Dashed red links indicate removed links. 45
5.11 Ablation impact on WMAPE and RelCon relative to the full ST-PINN GatedGCN model. Positive values indicate worse performance than the full model. 49

List of Tables

4.1 Training hyperparameters of ST-PINN GatedGCN. 28
4.2 Ablation variants used to isolate the main components of ST-PINN GatedGCN. 32

5.1 Overall prediction performance on Sioux Falls. 36
5.2 Overall prediction performance on the EMA network. 38
5.3 Relative retained-edge improvement of ST-PINN GatedGCN over the strongest baseline. 40
5.4 New-edge penalty and relative improvement of ST-PINN GatedGCN. 41
5.5 Prediction accuracy and flow-conservation violation on the test set. RelCon measures the relative violation of the node-level flow-conservation constraint. 43
5.6 Accuracy-conservation trade-off under different conservation weights. 44
5.7 Ablation results of ST-PINN GatedGCN on Sioux Falls and EMA. . . 47
5.8 Summary of prediction accuracy, physical consistency, and computational efficiency. 51
5.9 Training time of ST-PINN GatedGCN. 52

1

Introduction

1.1 Background

Traffic flow prediction is a central problem in transportation system analysis as it supports monitoring, control, and planning decisions across modern road networks [1, 2]. The increasing availability of loop detectors, wireless sensing infrastructure, mobile traces, and large-scale trajectory data has substantially expanded the empirical basis of traffic modelling [2, 3]. As a result, a large number of the recent literature has focused on temporal forecasting, where the objective is to predict future traffic speed, flow, or travel time from historical observations [1, 4]. These models are highly valuable for short-term operations and real-time traffic management. However, they address only one part of the broader problem faced by transportation agencies. In many planning and operational settings, the key question is not only how traffic evolves over time, but also how traffic redistributes when the network itself is modified.

This setting can be described as traffic flow prediction under network reconfiguration. In such problems, the aim is to estimate the equilibrium flow pattern after a structural change to the network, such as a road closure, the addition of a new road link, a lane-capacity adjustment, or a change in speed limits. The task is therefore counterfactual: the target flow field must be inferred from the currently observed flows together with the proposed reconfiguration. This type of analysis is essential in infrastructure planning, disruption management, and rapid what-if scenario screening, where decision makers need reliable estimates of post-edit traffic information before the intervention is implemented.

Within conventional transportation planning, these questions are typically addressed through origin-destination (OD)-based traffic assignment. Starting from Wardrop's equilibrium principle [5], practical workflows generally require an OD demand matrix together with a traffic assignment model, such as user equilibrium or stochastic user equilibrium, in order to compute stationary link flows [6, 7]. Although this framework is theoretically well established, its practical use in rapid scenario analysis is constrained by two major difficulties. First, OD demand is not directly observable and must instead be obtained from surveys, traffic counts, or heterogeneous proxy data, making the estimation problem underdetermined and sensitive to modelling assumptions [8, 9, 10]. Second, equilibrium assignment usually has to be solved repeatedly for each candidate intervention, which creates a substantial computa-

tional burden in large networks or in applications that require many counterfactual evaluations [6, 11].

These limitations motivate the development of learned surrogate models that can approximate post-edit flow redistribution directly from network structure, edge attributes, and observed traffic states. Graph neural networks (GNNs) are a natural candidate for this purpose because they have a structure similar to that of road systems: nodes represent intersections or centroids, whereas edges represent directed road segments with physical and operational attributes. GNNs have already shown strong performance in traffic forecasting by learning spatial dependencies on road networks [4, 10], and recent work has begun to extend graph-based learning to traffic assignment itself [11]. Based on this emerging direction, this thesis investigates whether graph-based and physics-informed learning can predict equilibrium link flows under network reconfiguration without requiring explicit OD information. Such a model would offer both practical value, and methodological value, by connecting transportation equilibrium modelling with modern graph-based machine learning.

1.2 Problem Statement

In this thesis, the problem is defined as predicting post-edit stationary link flows, given the pre-edit network, the modified network, and observed pre-edit link flows, but without access to the OD demand matrix. This problem is closely related to equilibrium-based traffic assignment, where the objective is to determine link flows and route-choice behaviour in a steady state [5].

Let the pre-edit road network be denoted by $G = (V, E, A)$, where V is the node set, E is the directed edge set, and A collects the edge attributes. After network reconfiguration, the new network is written as $G' = (V, E', A')$. The modification may involve adding or removing links, or changing key edge attributes such as capacity or free-flow travel time. The node set remains unchanged before and after the reconfiguration. The objective is not to model the short transient state immediately after the intervention. Instead, the goal is to predict the link flows after traffic has redistributed and the network has reached a new equilibrium.

The prediction target is

$$f^{\text{new}} = \{f_e^{\text{new}} : e \in E'\} \in \mathbb{R}_+^{|E'|}. \tag{1.1}$$

where f^{new} denotes the stationary flow on edge e in the modified network. We assume that the structure and edge attributes of both the pre-edit and post-edit networks are available, together with the observed pre-edit link flows

$$f^{\text{old}} = \{f_e^{\text{old}} : e \in E\} \in \mathbb{R}_+^{|E|}. \tag{1.2}$$

The task is to learn a mapping

$$F : (G, G', f^{\text{old}}) \mapsto \hat{f}^{\text{new}}. \tag{1.3}$$

such that $\hat{f}^{\text{new}}$ approximates the true post-edit equilibrium flow f^{new}.

A central assumption is that the latent OD demand remains fixed before and after the network reconfiguration. The intervention changes the network structure rather than the travel demand itself. Thus, differences between pre-edit and post-edit flows are attributed to route redistribution induced by the modified network. However, the OD matrix is not observed and is not provided to the model during the training.

This problem is difficult for several reasons. First, changes in topology or edge attributes affect feasible routes, travel costs, and flow distribution at the network level. Even a local intervention may therefore have non-local consequences. Post-edit flow cannot, in general, be inferred from a purely local correction of its pre-edit value. Second, once OD information is removed, the inverse problem becomes underdetermined. Different latent demand patterns may generate similar pre-edit observations f^{old}, yet produce different equilibrium flows on the modified network G'. Therefore, the mapping from (G, G', f^{old}) to f^{new} is, in general, not unique. This non-uniqueness is a fundamental difficulty of the OD-free setting. The learning problem is thus to infer a practically useful approximation of post-edit equilibrium flows from structural and observational information alone.

1.3 Aim and Research Questions

The aim of this thesis is to study OD-free prediction of equilibrium link flows under road network reconfiguration. The input consists of the pre-edit graph $G = (V, E, A)$, the observed pre-edit flow f^{old}, and the reconfigured graph $G' = (V, E', A')$. The target is the post-edit equilibrium flow f^{new} on G'. The OD demand matrix Q is used only to generate SUE labels and is not provided for the learning model.

The thesis first examines the methodological gap in existing approaches. It then formulates the prediction task as an OD-free flow-to-flow problem and evaluates a physics-informed graph surrogate for this setting. The research questions are:

RQ1 What methodological gap remains when existing traffic assignment, OD estimation, graph forecasting, and graph surrogate are compared for OD-free equilibrium flow prediction under network reconfiguration?

RQ2 How can post-edit equilibrium flow prediction be formulated as an OD-free flow-to-flow learning problem using G, f^{old}, and G', while treating the OD demand Q as latent?

RQ3 Can a physics-informed graph surrogate predict post-edit link flows on different road networks more accurately than learning-based baselines?

RQ4 How do prediction error and physical consistency vary across network scales, edge types, and model components, especially for retained edges, newly added edges, and ablated model variants?

1.4 Contributions

This thesis makes three main contributions.

First, it formulates traffic flow prediction as an OD-free learning problem. The task is defined as predicting stationary link flows under network reconfiguration, without using the OD demand matrix as an input. This formulation reflects a practically relevant setting in which reliable OD information is unavailable or too costly to estimate.

Second, it develops a data generation pipeline for supervised learning in this setting. The pipeline constructs paired pre-edit and post-edit traffic networks, applies controlled network modifications, and computes post-edit equilibrium flows using traffic assignment. This makes it possible to train and evaluate learning-based models on counterfactual flow redistribution tasks with ground truth.

Third, it proposes a graph-based prediction model for post-edit traffic flow estimation. The model is designed to use both network topology and edge attributes, together with observed pre-edit flows, to predict stationary flows on the modified network. The design is motivated by the relational structure of road systems and by the need to capture non-local effects induced by network changes.

1.5 Thesis Outline

The rest of this thesis is organized as follows. Chapter 2 reviews related work on traffic assignment and graph neural networks. It also clarifies the research gap addressed in this thesis. Chapter 3 presents the proposed physics-informed graph learning framework for OD-free flow-to-flow prediction. It describes the generation of network-pair dataset, the graph-based surrogate model, the learning objective, and the well-posedness of the prediction problem. Chapter 4 describes the implementation, including the training setup, baseline models, and ablation studies. Chapter 5 reports the experimental results on the Sioux Falls and EMA networks. It discusses prediction accuracy, physical consistency, edge-type behavior, computational efficiency, and ablation results. Chapter 6 concludes the thesis by interpreting the main findings, discussing limitations, and outlining directions for future work.

2

Related Work on Traffic Assignment and Graph-Based Learning

2.1 Traffic Assignment

Traffic assignment studies how travel demand is distributed over a transportation network. The network is usually represented as a directed graph, with nodes corresponding to intersections, centroids, or terminals, and links corresponding to road segments. Travel demand is specified by an OD matrix. In static traffic assignment, the objective is to determine stationary route and link flows for a given network, a given OD matrix, and a set of link cost functions. The problem is therefore concerned with equilibrium flow patterns rather than short-term traffic dynamics.

The classical starting point is Wardrop's first principle, which states that, at equilibrium, all used routes between an OD pair have equal and minimal travel cost, and that no traveller can reduce travel time by switching routes unilaterally [5]. This condition defines the user equilibrium. A related concept is the system optimum, in which total network travel cost is minimized. The user equilibrium formulation was given a precise mathematical structure by Beckmann, McGuire, and Winsten [12], who showed that, under separable and monotone link cost functions, the problem can be written as a convex optimization problem.

Deterministic user equilibrium assumes that travellers perceive route costs exactly. This assumption is often too strong in practice. Travellers may differ in information, perception, and preferences, and they may not respond identically to the same network conditions. For this reason, stochastic traffic assignment models were introduced. In particular, stochastic user equilibrium allows route choice to depend on perceived rather than exact costs [7]. This formulation is widely used when route choice uncertainty must be taken into account.

Static traffic assignment therefore requires three central parts: a network representation, an OD matrix, and a specification of link cost functions. Once these are given, equilibrium flows are computed by iterative numerical methods. Standard references include Sheffi [6] and Patriksson [13], who describe both the theoretical basis and the main solution methods used in practice, including convex combinations and

Frank–Wolfe-type algorithms. These models remain a core part of transportation analysis and provide the standard framework for estimating stationary link flows under congestion.

2.2 Graph Neural Networks

GNNs are learning models for graph-structured data. A graph consists of a node set V and an edge set E. In networked physical systems, nodes represent physical entities and edges represent interactions or couplings between them. This representation is useful when the system is governed by local interactions but exhibits global collective behaviour. Examples include molecules, particle systems, power networks, and transportation networks. In such settings, graph structure provides a suitable inductive bias for learning relational dependencies [14].

A common GNN formulation is message passing. Let $G = (V, E, A)$ be a graph. For a node $v \in V$, let $h_v^{(\ell)}$ denote its hidden representation at layer ℓ. For an edge $e = (u, v) \in E$, let a_e denote its edge attributes. A message passing layer computes messages along edges and then updates node representations by aggregation:

$$m_{u \to v}^{(\ell)} = \phi_m\Big(h_u^{(\ell)},\, h_v^{(\ell)},\, a_e\Big)\,, \tag{2.1}$$

$$h_v^{(\ell+1)} = \phi_h\Big(h_v^{(\ell)},\, \mathrm{AGG}_{u \in \mathcal{N}(v)} m_{u \to v}^{(\ell)}\Big)\,. \tag{2.2}$$

Here, $\mathcal{N}(v)$ denotes the neighborhood of node v, that is, the set of nodes connected to v. The function ϕ_m is a learnable message function. It maps the source-node state, the target-node state, and the edge attributes to an edge-wise message. The function ϕ_h is a learnable update function. It maps the previous node state and the aggregated incoming message to the next node representation. In practice, ϕ_m and ϕ_h are parameterized by neural-network modules, multi-layer perceptrons with nonlinear activations. The trainable parameters of the message-passing layer are contained in these two functions. The aggregation operator AGG is permutation-invariant, such as summation, mean, or max. By stacking several layers, information can propagate beyond immediate neighbors. This allows the model to capture long-range dependencies through repeated local interactions. A general message passing framework was formalized by Gilmer et al. [15].

GNNs have been used in several classes of physical problems. In molecular modelling, they learn from atoms and bonds while respecting the underlying interaction graph [15]. In physical simulation, graph-based models have been used to represent object-centric systems and to learn dynamics directly from relational states [16]. More broadly, Battaglia et al. [14] argue that graph-based architectures are well suited to domains in which entities interact through structured relations. This is important in networked physical systems because the state of one component is rarely independent of its neighbors. Instead, system behaviour emerges from many coupled local interactions.

One useful GNN architecture for such problems is the Gated Graph Convolutional Network (GatedGCN) proposed by Bresson and Laurent [17]. In this architecture, messages are modulated by learned edge-dependent gates, and node and edge representations are updated jointly. This is useful when interactions have different strengths or different functional roles. In many physical networks, not all adjacent connections contribute equally to the final response. Edge gating allows the model to learn this heterogeneity directly from data. Residual connections are also included to support deeper message passing and more stable optimization. For these reasons, GatedGCN has become a practical architecture for graph learning tasks in which edge attributes carry substantial information.

2.3 Physics-Informed Learning on Graphs

Physics-informed learning incorporates known physical structure into the learning process. In the classical PINN framework, this is done by penalizing violations of governing equations, boundary conditions, or conservation laws during training rather than learning solely from data [18]. The main idea is that predictions should not only fit observations, but also remain consistent with the underlying physical system. This often improves data efficiency, interpretability, and generalization. A broader review is given by Karniadakis et al. [19].

In many applications, the governing structure is not expressed most naturally on a regular spatial grid. Instead, the system is better described by interacting entities and relations. In such cases, graph-based models provide a convenient representation. Physics-informed learning on graphs combines this representation with structural constraints derived from the domain. These constraints may take the form of conservation laws, symmetry conditions, constitutive relations, or balance equations. The purpose is not to replace graph learning, but to restrict it to physically meaningful solution spaces.

For networked systems, conservation is often the most basic structural prior. In a flow network, for example, predictions should satisfy balance relations at nodes and remain compatible with edge-dependent costs or interactions. Similar ideas appear in graph-based physical modelling, where relational architectures are combined with physical constraints to improve stability and consistency [16]. The same principle has also been explored in physics-informed graph neural networks for dynamical systems, where conservation laws are enforced directly in the learned interaction model [20]. These examples illustrate a general point: in graph domains, physical knowledge is often expressed through relations between nodes and edges rather than through pointwise residuals alone.

This perspective is relevant in transportation as well. Traffic flow on a road network is not an arbitrary edge-level signal. It is shaped by network connectivity, conservation of flow, congestion-dependent costs, and route-choice equilibrium. For that reason, graph-based surrogates for traffic assignment can benefit from physically meaningful constraints in addition to data fitting. Recent work has also begun

to study graph neural networks for traffic assignment directly, which further supports the use of graph-based learning in this setting [11]. Physics-informed learning on graphs can therefore be understood as a natural extension of data-driven graph modelling to domains where structural consistency matters.

2.4 Research Gap

Classical traffic assignment provides a well-established framework for analysing equilibrium flow distribution in road networks. Standard formulations assume a known OD matrix and compute equilibrium flows by solving an assignment problem on a given network [5, 13]. The same general framework is also used when candidate network changes are evaluated. In that case, a new assignment problem must be solved for each modified network. This remains effective, but it becomes restrictive when OD information is uncertain and when many scenarios must be screened quickly.

Graph-based learning has become common in transportation research, but most existing work addresses a different task. Representative models such as DCRNN [4] and STGCN [21] are designed for spatio-temporal forecasting on a fixed graph. Their objective is to predict future traffic states from historical observations while the network topology remains unchanged. This work has been highly successful, and recent reviews [22] confirm its central role in transportation applications of graph neural networks. However, forecasting on a fixed topology does not address how equilibrium flows change after links are added, removed, or modified.

More direct work on traffic assignment with graph neural networks has only appeared recently. Liu and Meidani [11] proposed an end-to-end heterogeneous GNN for static traffic assignment. A later extension considered multi-class traffic assignment in the same general framework [23]. These studies show that graph-based surrogates can approximate assignment outputs with good accuracy. They are an important step forward. At the same time, they remain OD-based formulations. OD demand is provided explicitly, and the task is still posed as assignment under known demand rather than inference from observed pre-edit flows alone.

The literature therefore still contains a clear gap. There is extensive work on OD-based traffic assignment. There is also extensive work on graph-based traffic forecasting on fixed networks. More recently, there are graph neural network surrogates for traffic assignment itself. What is still missing is a method that combines these works in the specific setting of network reconfiguration and OD-free information. In particular, the literature lacks graph-based models that use the pre-edit network, the modified network, and observed pre-edit link flows to predict post-edit equilibrium link flows without explicit OD input.

3

Physics-Informed Graph Learning Framework for Flow Prediction

3.1 Ground-Truth Generation through Traffic Assignment

The supervised dataset used in this thesis is generated by combining benchmark road networks, controlled network edits, and equilibrium traffic assignment. Each training instance is built as a paired pre-edit and post-edit network state. The pre-edit state provides the baseline structure and flow pattern. The post-edit state provides the counterfactual target that the learning model is asked to predict. This design makes the data generation process explicit and keeps the prediction task closely aligned with the thesis objective.

3.1.1 Generation of Pre-edit Traffic Scenarios

Each sample starts from a benchmark road network represented as a directed graph with a fixed node set and a fixed base topology. In this thesis, the graph edges carry physical attributes that describe the operating state of the network. These attributes are capacity, speed, and length. Free-flow travel time is then computed from edge length and speed. A pre-edit traffic scenario is defined before any structural modification is applied. It therefore consists of the base graph together with scenario-specific edge attributes and an OD demand matrix used by the assignment solver.

To generate multiple pre-edit states on the same base network, link-level operating conditions are varied across scenarios. In particular, capacities and operating speeds are sampled over predefined feasible ranges by Latin Hypercube Sampling (LHS) [24]. This produces a diverse set of baseline conditions while keeping the sampling procedure systematic. Once these quantities have been sampled, the corresponding free-flow travel times are recomputed. The result is a collection of pre-edit network states that share the same topology but differ in operational conditions.

Demand is also generated by LHS and then kept fixed within each old–new pair. In all cases, the demand information is used only for label generation through traffic assignment. It is not part of the predictor input at inference time. This distinction is central to the thesis. The baseline scenario must contain enough information to

define a physically meaningful pre-edit flow pattern, but the later learning problem remains OD-free.

The output of this stage is a collection of pre-edit network states. Each state includes the original graph, scenario-specific edge attributes, and a demand matrix that is hidden from the predictor. These baseline states are necessary because counterfactual prediction is defined relative to an observed pre-edit condition rather than from topology alone.

3.1.2 Network Reconfiguration Operators

After the pre-edit scenario has been defined, an edited network is created by applying explicit reconfiguration operators. The node set is kept unchanged throughout. Reconfiguration acts only on the edge set and on edge attributes. The purpose of this step is to construct a post-edit network that differs from the baseline in a controlled but non-trivial way.

The reconfiguration strategy combines topology change and attribute perturbation. Topology change is carried out in two stages. First, new edges are added from nodes with relatively high baseline flow that is obtained by the SUE solver. For each selected source node, a small number of outgoing connections is introduced toward previously unconnected nodes. Second, edges are removed from nodes with relatively low baseline flow. A deletion is accepted only if the edited graph remains strongly connected [25]. Here, strongly connected means that for every ordered pair of nodes, there exists a directed path from the first node to the second. This condition is imposed so that feasible routes remain available in the later assignment step.

After topology change, the edited network is further modified through edge-attribute perturbation. A subset of edges is selected, and their capacities and speeds are rescaled within admissible bounds. The corresponding free-flow travel times are then recomputed. This step allows the counterfactual network to reflect both structural edits and operational changes. The resulting reconfiguration operators are therefore aligned with standard categories in transportation network design, such as edge addition, edge removal, and capacity or performance adjustment [26].

3.1.3 Equilibrium Flow Computation

Once both the pre-edit and edited networks have been defined, ground-truth edge flows are generated through stochastic user equilibrium assignment. Actually, in a deterministic user equilibrium, all used routes between an origin and a destination have equal and minimal travel cost. In a SUE, travellers are allowed to have imperfect perception of route costs. Lower-cost routes are more likely to be chosen, but routes with higher perceived cost may still carry some flow.

Congestion is represented through flow-dependent edge travel times. For each directed edge e, the travel time is given by a Bureau of Public Roads (BPR) type

function,

$$t_e(f_e) = t_e^0 \left[1 + \alpha \left(\frac{f_e}{\kappa_e} \right)^{\beta} \right], \tag{3.1}$$

where t_e^0 is the free-flow travel time, κ_e is edge capacity, and α and β are congestion parameters. The travel-time vector $t(f)$ therefore changes as traffic is assigned to the network.

For a given graph $G = (V, E, A)$, OD demand matrix Q, and travel-time vector t, stochastic network loading is formulated through a Markov-logit route-choice model. For each destination s, a value function V_i^s represents the expected downstream travel cost from node i to destination s. It satisfies the recursive logit relation

$$V_i^s = -\frac{1}{\theta} \log \sum_{(i,j)\in E} \exp\left[-\theta \left(t_{ij} + V_j^s \right) \right], \qquad V_s^s = 0, \tag{3.2}$$

where θ is the logit dispersion parameter. This gives the destination-specific transition probability

$$p_{ij}^s = \frac{\exp\left[-\theta \left(t_{ij} + V_j^s \right) \right]}{\sum_{(i,k)\in E} \exp\left[-\theta \left(t_{ik} + V_k^s \right) \right]}. \tag{3.3}$$

The probability p_{ij}^s describes the share of traffic heading to destination s that moves from node i to node j. It is larger for downstream alternatives with lower perceived total cost.

Given these transition probabilities, destination-based traffic loading can be written as a balance relation. Let x_i^s denote the amount of traffic at node i with destination s, and let q_i^s denote the corresponding OD demand injected at node i. Then

$$x_i^s = q_i^s + \sum_{(j,i)\in E} x_j^s p_{ji}^s. \tag{3.4}$$

The induced edge flow is

$$y_{ij} = \sum_s x_i^s p_{ij}^s. \tag{3.5}$$

This loading step maps a current travel-time field to an edge-flow field.

The SUE flow is obtained as a fixed point between congestion-dependent travel times and stochastic network loading:

$$f = \mathcal{L}_\theta\big(t(f), Q; G\big), \tag{3.6}$$

where $\mathcal{L}_\theta$ denotes the Markov-logit loading operator. The fixed point is approximated by a method-of-successive-averages update. At iteration k, travel times are evaluated from the current flow $f^{(k)}$, stochastic loading produces an auxiliary flow $y^{(k)}$, and the assigned flow is updated as

$$f^{(k+1)} = (1 - \lambda_k) f^{(k)} + \lambda_k y^{(k)}, \tag{3.7}$$

where λ_k is a bounded step size. The iteration continues until the flow and cost changes are sufficiently small.

For each network-pair sample, the same latent OD demand is used before and after reconfiguration. The pre-edit equilibrium flow is

$$f^{\text{old}} = \Phi_{\text{SUE}}(G, Q), \tag{3.8}$$

and the post-edit equilibrium flow is

$$f^{\text{new}} = \Phi_{\text{SUE}}(G', Q). \tag{3.9}$$

The vector f^{new} is used as the supervision target. The OD demand Q is part of the label-generation process, but it is not provided to the graph learning model. The prediction task is therefore to approximate SUE-generated post-edit link flows from G, G', and f^{old}, without explicit OD demand input.

3.1.4 Construction of Network-Pair Samples

After both equilibrium solves have been completed, the final dataset is seen as a collection of paired samples. Each instance contains a pre-edit graph, its equilibrium edge flows, the edited graph obtained through reconfiguration, and the post-edit equilibrium flows computed on that edited graph. The same latent demand scenario underlies both states in the pair, but this demand is not exposed to the predictor. The pair therefore captures a controlled before–after change in network structure and flow distribution.

From a macro perspective, the label-generation stage can be viewed as

$$(G, Q, \Delta G) \mapsto (G', f^{\text{new}}), \tag{3.10}$$

where G is the pre-edit network, Q is the OD demand matrix used internally by the assignment solver, and ΔG denotes the applied network modification. For the downstream learning task, this is converted into the OD-free supervised mapping

$$\left(G, f^{\text{old}}, \Delta G\right) \mapsto f^{\text{new}}. \tag{3.11}$$

This formulation matches the purpose of the thesis. The model is not asked to recover demand. It is asked to infer how an observed baseline flow pattern redistributes when the network is edited. This paired construction is appropriate for two reasons. First, it preserves the relational structure of the counterfactual problem. The post-edit target is always tied to a specific pre-edit baseline and a specific modification. Second, it makes the later graph-learning task well defined. The predictor receives information about the original network state and the edited network state, and it learns to approximate the stationary post-edit flow field generated by equilibrium assignment. In this way, the dataset serves as a bridge between classical traffic assignment and graph-based surrogate modelling for counterfactual flow prediction.

3.2 Graph-Based Surrogate Model

The proposed model, referred to as ST-PINN GatedGCN, is an edge-centric physics-informed graph surrogate for OD-free flow prediction under network reconfiguration. ST-PINN denotes a spatio-topological physics-informed neural network. The term spatio-topological reflects that the model operates on a pair of graph structures and must account for changes in both edge attributes and connectivity. The term physics-informed reflects the use of node-level flow imbalanced pressure and flow-conservation regularization to guide the learned redistribution process. GatedGCN is used as the local message-passing backbone because it updates node and edge states jointly and uses edge-dependent gates to control how information is transmitted across directed links.

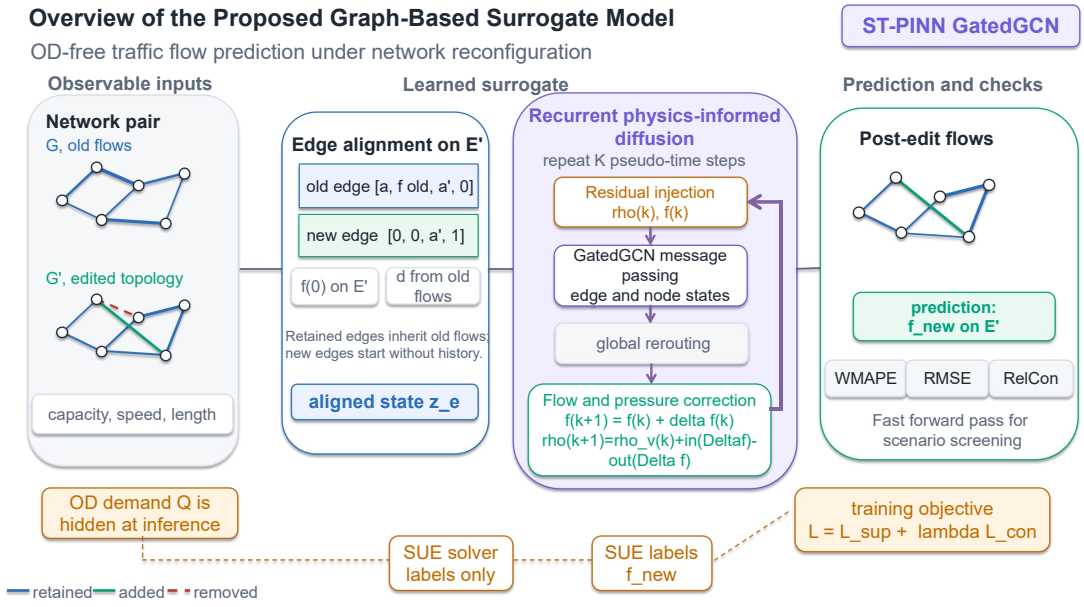

Figure 3.1: Overview of the ST-PINN GatedGCN surrogate for OD-free traffic flow prediction under network reconfiguration. The model takes the pre-edit network, observed old flows, and the reconfigured network as inputs. Edge alignment transfers old-flow information onto the new edge set, while recurrent physics-informed GatedGCN diffusion updates edge and node states through residual injection, message passing, and flow correction. OD demand is used only by the SUE solver to generate training labels. The final output is the predicted post-edit flow field on E', evaluated by WMAPE, RMSE, and RelCon.

3.2.1 Design Principle

Recent work has shown that graph neural networks can act as effective surrogates for traffic assignment and network flow prediction [11, 27]. However, most graph-based formulations assume that prediction is performed on a single fixed network. The present problem is different. The input flow is defined on the pre-edit network, while

the prediction target is defined on the post-edit network. A useful surrogate must therefore solve a paired-graph state transfer problem before any graph diffusion can begin.

This requirement motivates the structure of the model. First, the pre-edit flow state must be aligned with the reconfigured edge set. Second, the aligned state must be propagated through the edited topology in a way that is sensitive to directed edges and edge attributes. This motivates the use of a residual gated graph convolution, which maintains explicit edge states and edge-dependent gates, following the logic of Bresson et al. [28]. Third, the model must account for rerouting effects that are not confined to short graph distances. This motivates a nonlocal interaction module, referred to here as virtual routing. Fourth, the output should be guided by node-level flow conservation so that the predicted flow field remains compatible with the stationary traffic-assignment setting [5].

3.2.2 Edge Alignment Module

The first step of the model is to align the pre-edit edge state with the edited network. This step is necessary because there is no guaranteed one-to-one correspondence between the edges of G and G'. Some edges are retained, some are removed, and some are newly added. Even when an edge with the same ordered endpoints appears in both networks, its attributes may have changed after the edit. A direct transfer of the pre-edit flow vector is therefore not well defined.

The alignment module resolves only the unambiguous correspondences that remain valid after reconfiguration. An edge in the edited network is treated as retained if the same directed edge also exists in the pre-edit network. Otherwise, it is treated as a newly added edge. This rule is used both to initialize the flow state on G' and to construct the edge descriptor for the downstream graph model.

The initial projected flow on the edited network is defined edge-wise as follows:

$$f_e^{(0)} = \begin{cases} f_e^{\mathrm{old}}, & e \in E \cap E', \\ 0, & e \in E' \setminus E. \end{cases} \tag{3.12}$$

In words, retained edges inherit their observed pre-edit flow, whereas newly added edges are initialized with zero flow. This gives a simple and physically interpretable starting point. The model does not assume any immediate flow on links that did not exist before the edit.

The same correspondence rule is used to build the aligned edge descriptor. For each edge $e \in E'$, the descriptor contains old edge attributes, pre-edit flow, new edge attributes, and a binary indicator for whether the edge is new:

$$a_e^{\mathrm{align}} = \begin{cases} [a_e, f_e^{\mathrm{old}}, a'_e, 0], & e \in E \cap E', \\ [0, 0, a'_e, 1], & e \in E' \setminus E. \end{cases} \tag{3.13}$$

Here, a_e denotes the pre-edit edge attributes and a'_e denotes the post-edit edge attributes. In the present implementation, these attributes consist of capacity, speed, and length. The aligned descriptor therefore has eight components in total: three old attributes, one pre-edit flow value, three new attributes, and one binary indicator. This descriptor is then passed through a linear embedding layer in order to initialize the latent edge state used by the diffusion model.

Edge alignment also defines the initial imbalance field on the edited network. First, a node-level demand proxy is computed from the pre-edit stationary flow. For each node v, this quantity is the difference between total inflow and total outflow in the pre-edit network:

$$d_v = \sum_{e \in \mathrm{In}_G(v)} f_e^{\mathrm{old}} - \sum_{e \in \mathrm{Out}_G(v)} f_e^{\mathrm{old}}. \tag{3.14}$$

This quantity is carried into the edited network pair. After the projected flow $f_e^{(0)}$ has been defined on G', the initial imbalance is computed as

$$\rho_v^{(0)} = \sum_{e \in \mathrm{In}_{G'}(v)} f_e^{(0)} - \sum_{e \in \mathrm{Out}_{G'}(v)} f_e^{(0)} - d_v. \tag{3.15}$$

This is the difference between incoming and outgoing projected flow on the edited network, corrected by the node-level demand proxy from the pre-edit state. If $\rho_v^{(0)} \neq 0$, the projected flow is not yet compatible with the edited topology. The model therefore starts from a non-equilibrium state and must redistribute flow over the edited network in the subsequent diffusion steps. This makes the role of the alignment module clear: it does not solve the reconfiguration problem by itself, but it provides a structured and physically meaningful initialization for the later stages of the model.

3.2.3 GatedGCN-Based Pseudo-Time Diffusion

The core of the model is a recurrent diffusion process defined on the edited graph. The recurrence index $k = 0, 1, \ldots, K$ is a pseudo-time variable. It does not represent physical time. Instead, it indexes a sequence of iterative corrections that move the projected post-edit flow state toward a stationary configuration. The purpose of this module is therefore not temporal forecasting. Its purpose is to model flow redistribution after a network edit.

At each pseudo-time step, the model updates both node states and edge states. The edge state carries information about the current link condition, while the node state carries information about the local redistribution context. The current imbalance field and the current flow estimate are injected into these latent states before message passing is performed. In the implemented model, this injection is done by linear

projections. For a directed edge $u \to v$, the injected edge state is written as

$$\bar{g}_{uv}^{(k)} = W_g \begin{bmatrix} g_{uv}^{(k)} \\ \rho_u^{(k)} \\ \rho_v^{(k)} \\ \tilde{f}_{uv}^{(k)} \end{bmatrix} + b_g, \tag{3.16}$$

and the injected node state is

$$\bar{h}_u^{(k)} = W_h \begin{bmatrix} h_u^{(k)} \\ \rho_u^{(k)} \end{bmatrix} + b_h. \tag{3.17}$$

Here, $g_{uv}^{(k)}$ and $h_u^{(k)}$ are the current latent edge and node states, $\rho_u^{(k)}$ is the current node imbalance, and $\tilde{f}_{uv}^{(k)}$ is the current edge flow estimate in normalized space. In this thesis, normalized space refers to the standardized flow representation used by the learning model, whereas conservation updates are later applied in real flow units.

After this physical state has been injected, the model performs local propagation using a residual GatedGCN layer. The purpose of this step is to transmit rerouting information along the edited directed topology. For each directed edge $u \to v$, an edge-dependent gate is computed as

$$q_{uv}^{(k)} = W_D \bar{h}_v^{(k)} + W_E \bar{h}_u^{(k)} + W_C \bar{g}_{uv}^{(k)}, \qquad \eta_{uv}^{(k)} = \sigma\left(q_{uv}^{(k)}\right), \tag{3.18}$$

where $\eta_{uv}^{(k)}$ controls how much information is passed along that edge. The incoming message at node v is then

$$m_v^{(k)} = \frac{\sum_{u \in \mathcal{N}(v)} \eta_{uv}^{(k)} \odot W_B \bar{h}_u^{(k)}}{\sum_{u \in \mathcal{N}(v)} \eta_{uv}^{(k)} + \varepsilon}, \tag{3.19}$$

where $\mathcal{N}(v)$ denotes the set of predecessor nodes connected to v,and ε is a small constant added for numerical stability. The updated node and edge states are

$$h_v^{(k+1)} = \mathrm{Res}\left(W_A \bar{h}_v^{(k)} + m_v^{(k)}\right), \qquad g_{uv}^{(k+1)} = \mathrm{Res}\left(q_{uv}^{(k)}\right). \tag{3.20}$$

Here, $\mathrm{Res}(\cdot)$ denotes the residual update used in the implemented layer, including normalization, activation, and dropout. The same diffusion cell is reused at every pseudo-time step. This weight sharing is important because it makes the rollout behave as a repeated redistribution mechanism rather than as a stack of unrelated graph transformations.

Once the local propagation step has been completed, the updated edge state is decoded into a flow correction:

$$\Delta \tilde{f}_e^{(k)} = \psi\left(g_e^{(k+1)}\right), \qquad \tilde{f}_e^{(k+1)} = \tilde{f}_e^{(k)} + \Delta \tilde{f}_e^{(k)}. \tag{3.21}$$

The readout function $\psi(\cdot)$ is a small multi-layer perceptron. It predicts a correction in normalized space. This prediction is then converted to real flow units for the imbalance update. The imbalance field evolves according to

$$\rho_v^{(k+1)} = \rho_v^{(k)} + \sum_{e \in \mathrm{In}_{G'}(v)} \Delta f_e^{(k)} - \sum_{e \in \mathrm{Out}_{G'}(v)} \Delta f_e^{(k)}, \tag{3.22}$$

where $\Delta f_e^{(k)}$ denotes the real-valued flow correction on edge e. This update has a direct interpretation: if more corrective flow enters a node than leaves it, the imbalance increases; if more leaves than enters, the imbalance decreases.

This recurrence defines a discrete conservation-style redistribution process. It is not intended to be a full macroscopic traffic solver. Its role is more specific. It provides a structured latent evolution in which local flow corrections modify the node imbalance field, and the updated imbalance field then shapes the next round of edge-level corrections. In this way, the model approximates post-edit flow redistribution as an iterative relaxation process on the edited graph. That interpretation is the main reason for introducing pseudo-time diffusion in the first place.

3.2.4 Global Interaction Modeling via Virtual Routing

Local diffusion is necessary, but it is not sufficient. The GatedGCN update described above propagates information only through local graph connectivity. After K pseudo-time steps, information can only travel through a limited K-hop neighborhood. This is adequate for nearby corrections, but it may be insufficient when a local network edit changes route choice over a much larger part of the network. A deleted bridge, a new bypass, or a modified high-capacity corridor may influence flow redistribution far beyond the immediate neighborhood of the edited links.

To address this limitation, the model includes a virtual routing module. The purpose of this module is to introduce nonlocal interaction between nodes that are far apart in graph distance but strongly coupled in the final redistribution pattern. The module operates on the node-state matrix $H^{(k)}$ at pseudo-time step k. It uses multi-head self-attention [29], followed by layer normalization and a residual feed-forward block. Below, MHA denotes multi-head attention and LN denotes layer normalization. Since this is self-attention, the query, key, and value inputs are all taken from the same node-state matrix:

$$\mathbf{Q}_{\text{att}}^{(k)} = H^{(k)} W_Q, \qquad \mathbf{K}_{\text{att}}^{(k)} = H^{(k)} W_K, \qquad \mathbf{V}_{\text{att}}^{(k)} = H^{(k)} W_V. \tag{3.23}$$

At a high level, the module computes a dense interaction pattern within each graph:

$$\mathrm{VR}\Big(H^{(k)}\Big) = \mathrm{LN}\Big(H^{(k)} + \mathrm{MHA}\Big(\mathbf{Q}_{\text{att}}^{(k)}, \mathbf{K}_{\text{att}}^{(k)}, \mathbf{V}_{\text{att}}^{(k)}\Big)\Big), \tag{3.24}$$

followed by a residual feed-forward transformation. In expanded form, the attention weights are

$$\omega_{uv}^{(k)} = \mathrm{softmax}_v \left(\frac{\left(W_Q h_u^{(k)}\right)^\top \left(W_K h_v^{(k)}\right)}{\sqrt{d_{\text{att}}}} \right), \tag{3.25}$$

and the updated node representation is

$$\tilde{h}_u^{(k)} = \sum_v \omega_{uv}^{(k)} W_V h_v^{(k)}. \tag{3.26}$$

Here, $h_u^{(k)}$ is the node embedding of node u, W_Q, W_K, and W_V are learned projection matrices, and d_{att} is the attention dimension used for scaling. $\omega_{uv}^{(k)}$ measures how strongly node u attends to node v at pseudo-time step k.

In the implemented architecture, the virtual routing module is applied at every pseudo-time step. At each step k, the current node and edge states are first updated by local GatedGCN message passing. The global message channel is then evaluated using the updated states and injected back into the diffusion process before the next flow correction is computed.

3.3 Structure-Aware Learning Objective

The surrogate model is trained with a structure-aware objective rather than with point-wise regression alone. This is necessary because the task is to predict post-edit stationary flows on a reconfigured network, not just to match edgewise labels independently. The learning objective therefore combines direct supervision on the target link flows with graph-structured regularization terms that reflect nodal balance and terminal stability. This design is consistent with the role of conservation and equilibrium in static traffic assignment [5, 13], while remaining fully compatible with a learned graph surrogate [11].

3.3.1 Supervised Regression Loss

The first component of the training objective is a supervised regression loss on the post-edit edge flows. Let E' denote the edge set of the edited network. For each edge $e \in E'$, let $\tilde{f}_e$ denote the normalized solver-generated post-edit stationary flow, and let $\hat{\tilde{f}}_e$ denote the corresponding model prediction. The supervised loss is defined as the mean absolute error over the edited edge set:

$$\mathcal{L}_{\text{sup}} = \frac{1}{|E'|} \sum_{e \in E'} \left| \hat{\tilde{f}}_e^{\text{new}} - \tilde{f}_e^{\text{new}} \right|. \tag{3.27}$$

This term is the primary data-fitting signal in the model. It directly trains the surrogate to reproduce the post-edit equilibrium link flows generated by the traffic assignment solver. The loss is evaluated in normalized flow space. This is consistent with the data preprocessing described earlier and improves numerical stability across scenarios with different traffic magnitudes.

In the implemented model, the supervised term is based on mean absolute error (MAE) rather than mean squared error (MSE). The reason is practical. Even after normalization, some edges and some scenarios may produce relatively large residuals. An MSE loss would place disproportionate weight on these cases because the error is squared. MAE is less sensitive to such large deviations and therefore yields a more balanced training signal across the edited network. This is desirable here because the main objective is to learn the overall redistribution pattern of post-edit flows rather than to let a small number of large residuals dominate the optimization.

3.3.2 Equilibrium Flow Conservation

A purely supervised edgewise loss is not sufficient for this task. Two predicted flow vectors may have similar regression error while implying very different network

states. In particular, an edgewise prediction may fit the target approximately but still violate nodal mass balance or remain inconsistent with the stationary structure of the edited network. This issue is especially important under network reconfiguration, where the prediction is intended to represent a redistributed flow pattern rather than an arbitrary perturbation of edge values.

To encode this structure, the model introduces an equilibrium-oriented regularization based on the balance residual on the reconfigured graph. Let $f \in \mathbb{R}^{|E'|}$ be a real-valued flow vector on G'. Let d_v denote the node-level net-demand proxy inherited from the pre-edit stationary state. In this thesis, d_v is defined as the difference between total incoming and total outgoing flow at node v in the pre-edit network. The balance residual at node v is then defined as

$$\rho_v(f) = \sum_{e \in \mathrm{In}_{G'}(v)} f_e - \sum_{e \in \mathrm{Out}_{G'}(v)} f_e - d_v. \tag{3.28}$$

Equivalently, in matrix form,

$$\rho(f) = B'f - d, \tag{3.29}$$

where B' is the directed node–edge incidence operator of the reconfigured network. Since the surrogate model is trained without an explicit OD matrix, the vector d serves as an inherited nodal balance signal extracted from the pre-edit stationary flow state.

Let $\hat{f}^{\mathrm{new}}$ denote the final real-valued predicted flow. The corresponding terminal imbalance, or terminal pressure, at node v is $\rho_v(\hat{f}^{\mathrm{new}})$. The terminal equilibrium regularization is defined as

$$\mathcal{L}_{\mathrm{con}} = \frac{1}{|V|} \sum_{v \in V} \left(\frac{\rho_v(\hat{f}^{\mathrm{new}})}{\sigma_f} \right)^2, \tag{3.30}$$

where σ_f is the global flow scale used for normalization. This term acts as a soft equilibrium regularizer. It does not enforce an exact Wardrop user equilibrium. Instead, it penalizes terminal nodal imbalance and encourages the final prediction to remain closer to a stationary flow state consistent with the traffic assignment solver.

The full training objective is therefore written as

$$\mathcal{L} = \mathcal{L}_{\mathrm{sup}} + \lambda_{\mathrm{con}} \mathcal{L}_{\mathrm{con}}, \tag{3.31}$$

where $\mathcal{L}_{\mathrm{sup}}$ is the supervised regression loss defined in the previous subsection, and λ_{con} controls the strength of the equilibrium regularization. The resulting objective is structure-aware because the regularization term is defined through the directed network balance operator rather than through pointwise edge errors alone. This keeps the learning objective aligned with the physical and assignment-based interpretation of the surrogate model.

3.4 Well-Posedness of the OD-Free Flow-to-Flow Problem

This section clarifies the mathematical status of the OD-free flow-to-flow prediction problem. This task should not be interpreted as a deterministic inverse problem that recovers the hidden OD demand. Instead, it is a statistical prediction problem in which the target is a conditional flow given the observable pre-edit state and the reconfigured graph.

3.4.1 Observable Inputs and Latent Demand

The prediction problem in this thesis is built from a paired traffic-assignment process. $G = (V, E, A)$ denotes the pre-edit graph, and $G' = (V, E', A')$ denotes the reconfigured graph. The node set V is kept fixed in the data. The directed edge set may change from E to E'. The edge attributes A and A' include quantities such as capacity, speed, and free-flow travel time.

Let Q denote the latent OD demand matrix. The same demand is used before and after network reconfiguration, but it is not given to the learning model. If Φ denotes the traffic assignment operator, the generated equilibrium flows are

$$f^{\text{old}} = \Phi(G, Q), \qquad f^{\text{new}} = \Phi(G', Q). \tag{3.32}$$

The OD demand Q is used only inside the SUE solver to generate labels. At prediction time, the observable input is

$$X = (G, G', f^{\text{old}}), \tag{3.33}$$

and the target is

$$Y = f^{\text{new}}. \tag{3.34}$$

Thus, the model predicts post-edit equilibrium flows on the edge set E'. The output dimension is determined by the reconfigured graph. This is important because G' may contain added edges, removed edges, and changed edge attributes.

3.4.2 Non-Uniqueness of the Flow-to-Flow Map

A deterministic inverse formulation would require a single-valued map T that recovers the post-edit flow from the observable input. In this setting, such a map would have to satisfy

$$T\Big(G, G', \Phi(G, Q)\Big) = \Phi(G', Q) \tag{3.35}$$

for every feasible latent demand matrix Q.

This requirement fails if two different OD demand matrices lead to the same old flows but different new flows. Suppose there exist $Q_1 \neq Q_2$ such that

$$\Phi(G, Q_1) = \Phi(G, Q_2), \tag{3.36}$$

while

$$\Phi(G', Q_1) \neq \Phi(G', Q_2). \tag{3.37}$$

Then the same observable input X corresponds to two different valid outputs Y.

In fact, link flows do not generally identify OD demand. The issue has the same origin as the underdetermination in classical OD estimation from link counts. In the Sioux Falls and EMA settings, the OD demand matrix has many degrees of freedom. The observed old flows provide aggregate edge-level information, but they do not fully reveal hidden route demand. Even when G and f^{old} are known, multiple latent demand patterns may remain compatible with the same observed traffic state.

The conclusion is not that labels do not exist. In the generated dataset, each sample has a post-edit label after the SUE solver computes f^{new}. The failure is uniqueness. Since uniqueness is one of the Hadamard requirements for well-posedness, the failure of uniqueness implies that the OD-free flow-to-flow problem is not well posed as a deterministic inverse problem [30].

3.4.3 Statistical Formulation

The learning task can still be defined strictly as a statistical prediction problem. Let (G, G', Q) be drawn from the data-generating distribution used in this thesis. Then (X, Y) are jointly distributed random variables, where $X = (G, G', f^{\text{old}})$ and $Y = f^{\text{new}}$.

Under squared loss, define the population risk as

$$\mathcal{R}_2(g) = \mathbb{E}\left[\|Y - g(X)\|^2_{2,E'}\right], \tag{3.38}$$

where

$$\|z\|^2_{2,E'} = \sum_{e \in E'} z_e^2. \tag{3.39}$$

The edge set E' is part of X. The norm is therefore evaluated on the edge set of the current reconfigured graph.

The population-optimal predictor under squared loss is the conditional expectation

$$g_2^\star(X) = \mathbb{E}[Y \mid X]. \tag{3.40}$$

To see this, consider any measurable predictor g and define

$$\xi = Y - g_2^\star(X), \qquad \zeta = g_2^\star(X) - g(X). \tag{3.41}$$

Then

$$Y - g(X) = \xi + \zeta. \tag{3.42}$$

Using the squared norm on the edge set E' of the current reconfigured graph gives

$$\|Y - g(X)\|^2_{2,E'} = \|\xi + \zeta\|^2_{2,E'}. \tag{3.43}$$

Expanding the squared norm,

$$\|\xi + \zeta\|_{2,E'}^2 = \|\xi\|_{2,E'}^2 + \|\zeta\|_{2,E'}^2 + 2\langle \xi, \zeta \rangle_{E'}. \tag{3.44}$$

Substituting the definitions of ξ and ζ yields

$$\begin{aligned} \|Y - g(X)\|_{2,E'}^2 =& \|Y - g_2^\star(X)\|_{2,E'}^2 + \|g_2^\star(X) - g(X)\|_{2,E'}^2 \\ &+ 2\left\langle Y - g_2^\star(X), g_2^\star(X) - g(X) \right\rangle_{E'}. \end{aligned} \tag{3.45}$$

Taking expectations on both sides gives

$$\begin{aligned} \mathcal{R}_2(g) =& \mathbb{E}\Big[\|Y - g_2^\star(X)\|_{2,E'}^2\Big] + \mathbb{E}\Big[\|g_2^\star(X) - g(X)\|_{2,E'}^2\Big] \\ &+ 2\mathbb{E}[\left\langle Y - g_2^\star(X), g_2^\star(X) - g(X) \right\rangle_{E'}]. \end{aligned} \tag{3.46}$$

The cross term is zero. By the tower property,

$$\begin{aligned} &\mathbb{E}[\left\langle Y - g_2^\star(X), g_2^\star(X) - g(X) \right\rangle_{E'}] \\ &= \mathbb{E}[\mathbb{E}[\left\langle Y - g_2^\star(X), g_2^\star(X) - g(X) \right\rangle_{E'} \mid X]]. \end{aligned} \tag{3.47}$$

Conditioned on X, both $g_2^\star(X) - g(X)$ and E' are fixed. Therefore,

$$\begin{aligned} &\mathbb{E}[\left\langle Y - g_2^\star(X), g_2^\star(X) - g(X) \right\rangle_{E'} \mid X] \\ &= \left\langle \mathbb{E}[Y - g_2^\star(X) \mid X], g_2^\star(X) - g(X) \right\rangle_{E'}. \end{aligned} \tag{3.48}$$

Since $g_2^\star(X) = \mathbb{E}[Y \mid X]$,

$$\mathbb{E}[Y - g_2^\star(X) \mid X] = \mathbb{E}[Y \mid X] - g_2^\star(X) = 0. \tag{3.49}$$

Hence

$$\mathcal{R}_2(g) = \mathcal{R}_2(g_2^\star) + \mathbb{E}\Big[\|g(X) - g_2^\star(X)\|_{2,E'}^2\Big]. \tag{3.50}$$

Equivalently,

$$\mathcal{R}_2(g) = \mathcal{R}_2(g_2^\star) + \mathbb{E}\left[\|g(X) - g_2^\star(X)\|_{2,E'}^2\right]. \tag{3.51}$$

This decomposition shows that $g_2^\star$ is the unique population-optimal predictor in the L^2 sense, up to almost-sure equivalence.

So far, the supervised loss used in the implemented model is of L^1 type. Under an L^1 loss, the corresponding population target is a conditional median or another conditional location functional, not the conditional mean. This does not change the well-posedness argument. In both cases, the target is a statistical functional of $Y \mid X$, not an exact inverse recovery of Q.

3.4.4 Irreducible Error and Meaning for This Thesis

The minimum squared-loss risk is

$$\mathcal{R}_2^\star = \mathbb{E}\left[\|Y - \mathbb{E}[Y \mid X]\|_{2,E'}^2\right]. \tag{3.52}$$

Equivalently, this can be written as

$$\mathcal{R}_2^\star = \mathbb{E}\left[\operatorname{tr}\left(\operatorname{Cov}(Y \mid X)\right)\right]. \tag{3.53}$$

This quantity is zero only when Y is fully determined by X. It is positive when different latent OD demands remain possible under the same observable input. This error is not an optimization failure. It is also not only a limitation of model capacity. It is an information limit caused by the missing OD demand. The old flows provide strong information about the latent OD state, but they do not fully identify Q.

Therefore, the OD-free flow-to-flow problem should be understood as statistical prediction under partial information, rather than as deterministic recovery of the traffic assignment solution from complete demand data.

4

Implementation

4.1 Training Setup

Our model ST-PINN GatedGCN is trained to predict stationary link flows on the reconfigured network G'. For each graph-pair sample, the model receives the pre-edit network G, the reconfigured network G', and the observed pre-edit flows f^{old}. The prediction target is the normalized post-edit flow vector f^{new} on the edge set E'. All experiments are implemented with PyTorch Geometric [31]. Figure 4.1 shows how these graph-pair samples are generated.

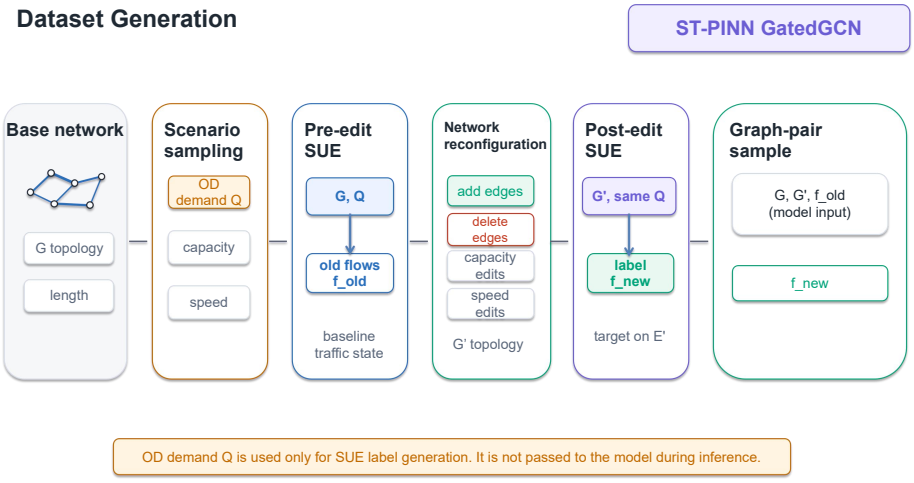

Figure 4.1: Dataset generation workflow. For each sampled OD demand Q, f^{old} is solved on the pre-edit network G by SUE. The network is then reconfigured, and SUE is solved again on G' with the same Q to obtain the training label f^{new}. The OD demand is used only for label generation and is not part of the model input.

The training objective combines supervised edge-flow regression with a physics-informed conservation term [18]. The loss used in training is

$$\mathcal{L} = \mathcal{L}_{\text{sup}} + \lambda_{\text{con}} \mathcal{L}_{\text{con}}. \tag{4.1}$$

Here, $\mathcal{L}_{\text{sup}}$ is mean L1 loss on retained and newly added edges. L1 loss is used as the supervised regression loss because it is less dominated by large edge-level residuals

than a squared loss. The conservation term is computed on the final real-scale prediction. The terminal balance residual is

$$\rho(\hat{f}^{\text{new}}) = B' \hat{f}^{\text{new}} - d. \tag{4.2}$$

Here, B' is the directed incidence operator of G', $\hat{\mathbf{f}}^{\text{new}}$ is the predicted real-scale flow on E', and $\mathbf{d}$ is the node-level net-demand proxy derived from the pre-edit flow. The conservation loss is

$$\mathcal{L}_{\text{con}} = \frac{1}{|V|} \sum_{v \in V} \left(\frac{\rho(\hat{f}^{\text{new}})}{\sigma_f} \right)^2 . \tag{4.3}$$

The division by σ_f keeps this term on the same numerical scale as the normalized flow supervised loss.

Figure 4.2 summarizes the implemented forward pass. The detailed modelling choices are the same as those introduced in Chapter 3.

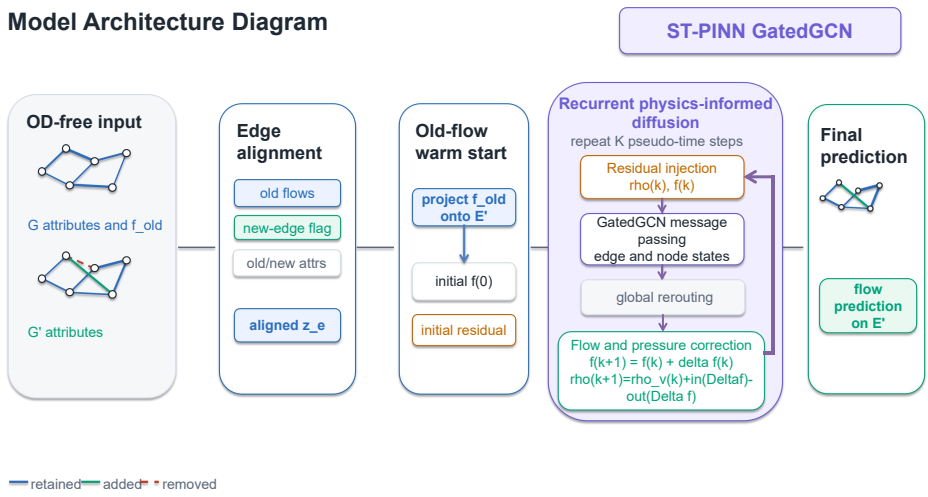

Figure 4.2: Implemented ST-PINN GatedGCN architecture. Edge alignment maps information from G to the new edge set E'. Retained edges can inherit old-flow information, while newly added edges are marked because they have no old flow. The recurrent GatedGCN diffusion then updates the flow state through residual injection, message passing, and flow correction.

The model uses $d_h = 128$ hidden channels and four pseudo-time diffusion steps. Each diffusion step uses GatedGCN message passing [17]. Residual connections [32] are enabled, and dropout with probability 0.1 is applied during training. The global message channel is enabled and applied at every diffusion step, with four attention heads following the multi-head attention mechanism. The feed-forward block uses SwiGLU [33], with $d_{\text{ff}} = \text{round}(8d_h/3) = 341$. RMSNorm is used with pre-normalization [34]. The model uses an old-flow warm start, recurrent LWR-style pressure updates, a shared diffusion cell across pseudo-time steps, and physical

residual injection into both edge and node updates, as specified and detailed in Chapter 3.

The optimizer is AdamW [35], with base learning rate $\eta = 10^{-3}$ and weight decay 10^{-5}. Training runs for 200 epochs with batch size 32. A warmup-stable-decay schedule [36] is used: the learning rate warms up for 10 epochs, remains stable for 140 epochs, and then follows a cosine decay for 50 epochs down to a minimum learning rate of 10^{-6}. Gradients are clipped by global norm with threshold 1.0. The checkpoint selected for final evaluation is the epoch with the lowest validation normalized RMSE.

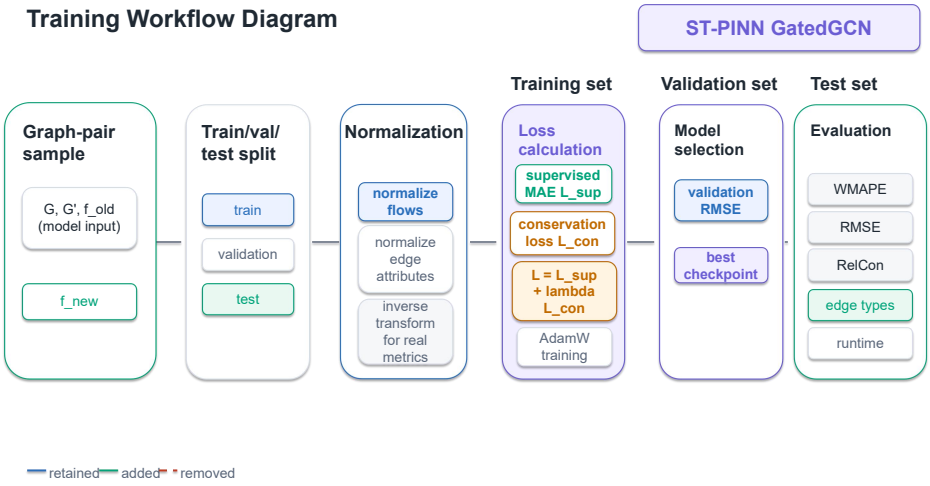

Figure 4.3: Training and evaluation workflow. Graph-pair samples are split into training, validation, and test sets. Flow values and edge attributes are normalized before training. The model is optimized with supervised edge-flow loss and conservation loss. The final checkpoint is selected on the validation set and then evaluated on the test set using WMAPE, RMSE, RelCon, edge-type errors, and runtime.

The main hyperparameters of our model are summarized in Table 4.1.

4.2 Baselines

This section describes the comparison models used to evaluate ST-PINN GatedGCN. It includes one nonlearning model and three learing-based models. The three learning-based baselines use the same aligned edge input, so that their differences come from how they process graph structure. All learned baselines predict normalized link flows on E' and are trained with the same supervised and conservation loss structure. Still, they do not use the OD matrix during inference.

4.2.1 Old-Flow Baseline

The Old-Flow baseline is a non-parametric baseline. It does not train a model. In this setting, if a directed edge in the reconfigured graph already existed in the

Table 4.1: Training hyperparameters of ST-PINN GatedGCN.

Category	Hyperparameter/Structure	Value
Architecture	Hidden dimension	$d_h = 128$
Architecture	Diffusion steps	4
Architecture	Local message passing	GatedGCN
Architecture	Global message channel	Enabled, every step
Architecture	Attention heads	4
Architecture	Residual connections	Enabled
Architecture	Dropout	0.1
Architecture	Normalization	RMSNorm, pre-norm
Architecture	FFN	SwiGLU, $d_{\text{ff}} = 341$
Architecture	Old-flow warm start	Enabled
Architecture	Pressure update	LWR
Architecture	Shared diffusion cell	Enabled
Architecture	Physical residual injection	Edges and nodes
Optimization	Loss function	L1
Optimization	Optimizer	AdamW
Optimization	Base learning rate	$\eta = 10^{-3}$
Optimization	Weight decay	10^{-5}
Optimization	Batch size	32
Optimization	Maximum epochs	200
Optimization	Gradient clipping	Global norm 1.0
Optimization	Scheduler	Warmup-stable-decay
Optimization	Warmup / stable / decay epochs	10 / 140 / 50
Optimization	Decay type	Cosine
Optimization	Minimum learning rate	10^{-6}
Model selection	Selection metric	Best validation normalized RMSE

pre-edit graph, its old flow then is copied as the prediction of this edge in the new graph.

For a retained edge $e \in E \cap E'$, the predicted flow is set equal to its pre-edit flow. For a newly added edge $e \in E' \setminus E$, no old flow is available. The prediction is then filled with the mean old flow over the retained edges in the same graph sample:

$$\hat{f}_e^{\text{new}} = \begin{cases} f_e^{\text{old}}, & e \in E \cap E', \\ \dfrac{1}{|E \cap E'|} \displaystyle\sum_{a \in E \cap E'} f_a^{\text{old}}, & e \in E' \setminus E. \end{cases} \tag{4.4}$$

Here, f_e^{old} is the old flow on edge e, and $\hat{f}_e^{\text{new}}$ is the predicted flow on the reconfigured graph. The mean is computed separately for each network-pair sample.

This baseline is not intended to be a strong predictor. It provides a lower reference for the task. It answers a simple question: how much of the post-edit flow can be explained by just copying the old traffic flow alone?

4.2.2 MLP Baseline

The MLP baseline is an edge-wise feed-forward model. It uses only the aligned local feature $\mathbf{z}_e$ and does not perform graph message passing. Each edge in G' is processed independently by the same multilayer perceptron [37]:

$$\hat{f}_e^{\text{new}} = \phi_{\text{MLP}}(\mathbf{z}_e). \tag{4.5}$$

This baseline tests how much of the post-edit flow can be inferred from local edge attributes and pre-edit flow alone. Since it does not exchange information across nodes or edges, it cannot explicitly propagate non-local effects caused by topology changes.

4.2.3 Single-Topology GatedGCN Baseline

The Single-Topology GatedGCN baseline performs message passing only on the reconfigured topology G'. It uses $\mathbf{z}_e$ as the edge feature for each edge in E', while node features are obtained from a simple node input encoding. GatedGCN layers then update node states and edge states jointly [17].

After message passing, the model predicts the post-edit link flow directly from the final edge representation:

$$\hat{f}_e^{\text{new}} = \psi_{\text{edge}}\left(\mathbf{h}_e^{\text{edge}}\right). \tag{4.6}$$

This baseline is closer to edge-level prediction than the Node-Centric GNN, because edge states are explicitly updated during message passing. However, it propagates only on G'. It does not include the recurrent physical correction and diffusion design used in ST-PINN GatedGCN.

4.2.4 Node-Centric GNN Baseline

The Node-Centric GNN baseline uses node-level message passing. It first encodes the original network G. During this stage, node states are initialized uniformly, and edge weights are computed from old edge attributes and old flow. The resulting node representation summarizes the pre-edit traffic state around each node.

The model then reasons on the reconfigured network G'. It fuses the old-network node state with a new node input, computes edge weights from $\mathbf{z}_e$, and applies graph convolutional message passing on G' [15]. The final prediction for an edge $e = (u, v)$ is decoded from the two endpoint node states and the aligned edge feature:

$$\hat{f}_e^{\text{new}} = \psi_{\text{node}}\left(\mathbf{h}_u^{\text{new}}, \mathbf{h}_v^{\text{new}}, \mathbf{z}_e\right), \quad e = (u, v). \tag{4.7}$$

This baseline can use topological adjacency, but its link-flow prediction is still mediated through node representations. It therefore tests whether node-centric message passing is sufficient for link-flow redistribution after reconfiguration.

4.2.5 SUE Solver

The SUE solver is not a learning model. It is the traffic-assignment reference used to generate ground-truth equilibrium flows and to support the runtime comparison. For each network state, the solver takes the OD matrix, link capacities, and free-flow travel times as inputs, and computes equilibrium link flows under stochastic route choice [5, 7, 6]. The edge travel time follows the BPR form [6]:

$$t_e(f_e) = t_e^0 \left[1 + \alpha \left(\frac{f_e}{\kappa_e} \right)^{\beta} \right], \tag{4.8}$$

where t_e^0 is the free-flow travel time and c_e is capacity. The implementation uses Markov-Logit network loading to compute stochastic loaded flows under current travel times. For a given graph $G = (V, E, A)$, OD demand matrix Q, and travel-time vector t, stochastic network loading is formulated through a Markov-logit route-choice model, following Markovian stochastic assignment and recursive logit route-choice models [38]. An outer method-of-successive-averages style update then mixes the current flow with the newly loaded flow until convergence.

The SUE solver is physically grounded because it solves each reconfigured network G' using the OD matrix and congestion-dependent travel times. At the same time, it must be run separately for each scenario, which makes it computationally heavier than a trained forward model.

Together, these comparisons form a progression. The Old-Flow baseline tests whether old flows alone explain the target. The MLP tests local edge-wise supervised learning without message passing. The Node-Centric GNN tests node-level graph propagation. The Single-Topology GatedGCN tests edge-aware message passing on G'. The SUE solver provides the traffic-assignment reference.

4.3 Ablation Study Design

The full ST-PINN GatedGCN combines old-flow information, old-new edge alignment, physical residual injection, a global message channel, shared recurrent diffusion cells, and recurrent pressure updates. The ablation study changes one component at a time while keeping the main training protocol and dataset split fixed. The goal is to identify which parts support edge-wise prediction, topology adaptation, and physical consistency. All ablation runs keep the same optimizer, batch size, number of epochs, hidden dimension, diffusion steps, local GatedGCN backbone [17], and dataset split as the full model.

4.3.1 Removal of Old-Flow Information

This ablation removes the direct old-flow signal from the aligned edge input. It sets the edge-alignment mode to `wo_old_flow`. For each edge $e \in E'$, the aligned feature keeps the matched old edge attributes and the new edge attributes, but replaces f^{old}

with zero:

$$\mathbf{z}_e^{\text{wo flow}} = \left[\mathbf{a}_{m(e)}^{\text{old}}, 0, \mathbf{a}_e^{\text{new}}, \mathbb{I}(e \notin E)\right]. \tag{4.9}$$

In addition, the initial flow mode is changed to zero initialization, and the initial pressure mode is also set to zero. Old edge attributes and new edge attributes remain available. Physical residual injection, the recurrent pressure update, the global message channel, and the shared diffusion cell are kept. This ablation tests whether old equilibrium flows act as a state anchor for predicting flows after network reconfiguration.

4.3.2 New-Attribute-Only Input

This ablation sets the edge-alignment mode to `new_attr_only`. It removes the aligned old edge attributes, old flow, and new-edge indicator from the edge input. The edge descriptor keeps only the new-network edge attributes:

$$\mathbf{z}_e^{\text{new attr}} = [\mathbf{0}, \mathbf{a}_e^{\text{new}}, 0]. \tag{4.10}$$

This is not the same as removing all old-state information. The old-flow warm start remains enabled, and the initial pressure is still computed from the initial projected flow. Physical residual injection, recurrent pressure update, global message channel, and shared diffusion cell are also kept. This ablation tests whether direct old-new edge alignment is needed beyond the warm-start state.

4.3.3 Removal of Physical Residual Injection

This ablation disables direct injection of the pressure residual that is sent to the latent edge and node states. The conservation loss is not removed. The model is still trained with the same λ_{con} schedule, and the terminal balance residual $\rho(\hat{f}^{\text{new}})$ is still penalized as defined above. Old-flow warm start, recurrent pressure update, global message channel, and shared diffusion cell are also kept. This ablation separates the effect of adding a physics-informed loss [18] from the effect of feeding the residual state into the model during diffusion.

4.3.4 Removal of Global Message Channel

This ablation disables the global message channel by setting `enable_global_attn = False`. All local GatedGCN diffusion steps remain active. Old-flow warm start, physical residual injection, recurrent pressure update, and shared diffusion cell are kept.

The purpose is to test whether graph-level communication helps when route changes are not limited to local neighborhoods. The full model uses a multi-head attention style global channel [29]; this ablation removes that channel and leaves only local propagation on G'.

4.3.5 Unshared Diffusion Cells

The full model reuses the same diffusion cell across pseudo-time steps. This gives the rollout a recurrent structure: each step applies the same learned update rule to the current flow and pressure state. The unshared-cell ablation sets `share_diffusion_cell = False`, so each diffusion step has separate parameters.

All other mechanisms are kept, including old-flow warm start, old-new edge alignment, physical residual injection, the global message channel, and recurrent pressure update. This ablation tests whether the shared recurrent update is a useful inductive bias, or whether separate step-specific parameters are preferable.

4.3.6 Removal of Recurrent Pressure Update

This ablation sets `pressure_update_mode = fixed_initial`. The model still computes the initial pressure from the old-flow warm start, but it does not update this pressure state after each diffusion step. Edge alignment, physical residual injection, global message channel, and shared diffusion cell are kept.

The pressure update is motivated by conservation-style traffic-flow dynamics, related to the LWR view of flow evolution [39]. This ablation tests whether repeated pressure updates are needed to track redistribution.

These ablations are designed as controlled changes around the same ST-PINN GatedGCN training protocol. They therefore support component-level interpretation in the Results chapter. The ablation studies are summarized in Table 4.2.

Table 4.2: Ablation variants used to isolate the main components of ST-PINN GatedGCN.

Variant	Modified component	Changed setting	Purpose
w/o old flow	Old-flow state	Remove f^{old}, zero initial flow and pressure	Test old-flow anchoring
new attributes only	Edge alignment	Keep only $\mathbf{a}^{\text{new}}$ in direct edge input	Test old-new alignment
w/o physical residual injection	Residual injection	Disable residual input to edges and nodes	Test residual feedback
w/o global message channel	Global communication	Disable global message channel	Test long-range exchange
unshared diffusion cells	Diffusion rollout	Use separate cells at each step	Test recurrent sharing
w/o recurrent pressure update	Pressure evolution	Keep fixed initial pressure	Test recurrent pressure tracking

5
Results

5.1 Experimental details

5.1.1 Datasets and Network Reconfiguration Scenarios

The experiments use two directed road networks: Sioux Falls and EMA. Each sample is formulated as a paired graph problem. The original graph is denoted by $G = (V, E, A)$, and the reconfigured graph is denoted by $G' = (V, E', A')$. The node set is kept fixed within each pair, while the edge set and edge attributes may change. In the datasets, Sioux Falls contains 24 nodes, 76 original directed edges. EMA contains 74 nodes, 258 original directed edges.

For each sample, the original graph stores the pre-edit topology, normalized edge attributes, and the observed old equilibrium flow f^{old} on each edge in E. The edge attributes are capacity, speed, and length. The reconfigured graph stores the modified topology and the updated edge attributes on E'. Reconfiguration includes added directed edges, edge removals, and attribute changes. Existing selected edges may have capacity and speed rescaled, while newly added edges are assigned capacity, speed, and length values.

The reconfiguration process is illustrated in Figures 5.1 and 5.2. These examples show one test sample from each network. The left picture shows the original network G, and the right one shows the corresponding reconfigured network G'. Dashed red links mark removed links, while the colour scale represents edge capacity. The figures also show that the number of edges in G' can vary by sample, because edges may be added and removed during scenario generation.

The prediction target is the post-edit equilibrium flow f^{new} on every edge of G'. These labels are produced by the data-generation pipeline using a traffic assignment solver for stochastic user equilibrium. The OD matrix is used only during this label-generation step. It is not stored as an input feature and is not used by any model at inference time.

The old edge flows are used as the observable pre-edit traffic state. They provide information about previous loading, imbalance, and likely redistribution pressure after the network change. In particular, a node-level net-demand pressure is derived from f^{old} on the original graph and is used for the conservation-based evaluation and physics-informed components.

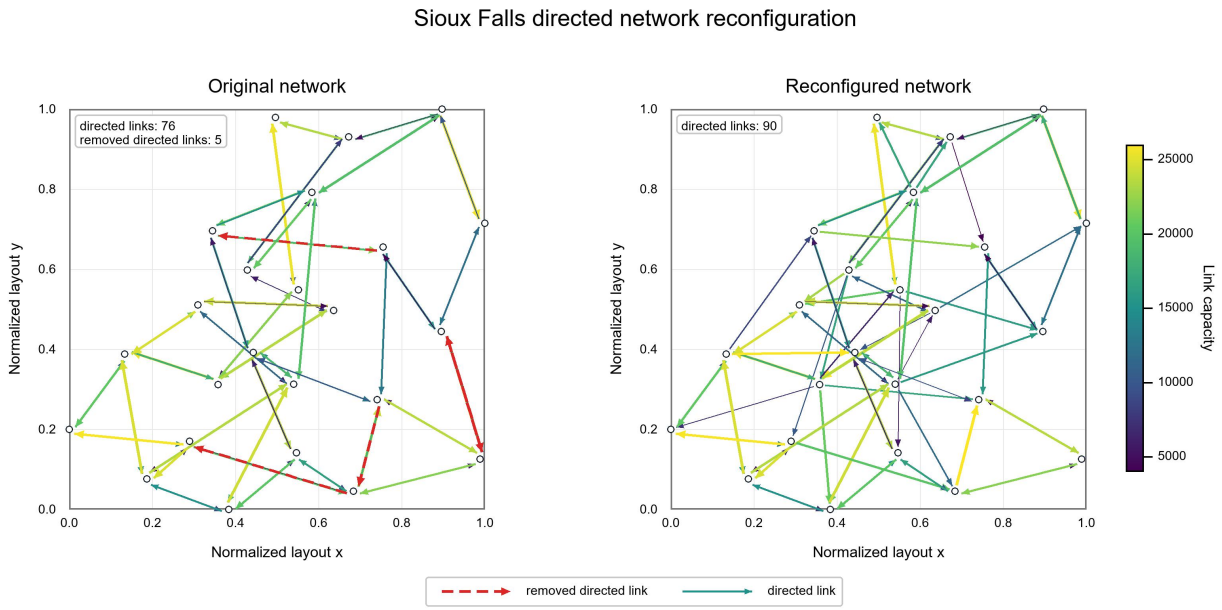

Figure 5.1: Example reconfiguration sample for Sioux Falls.

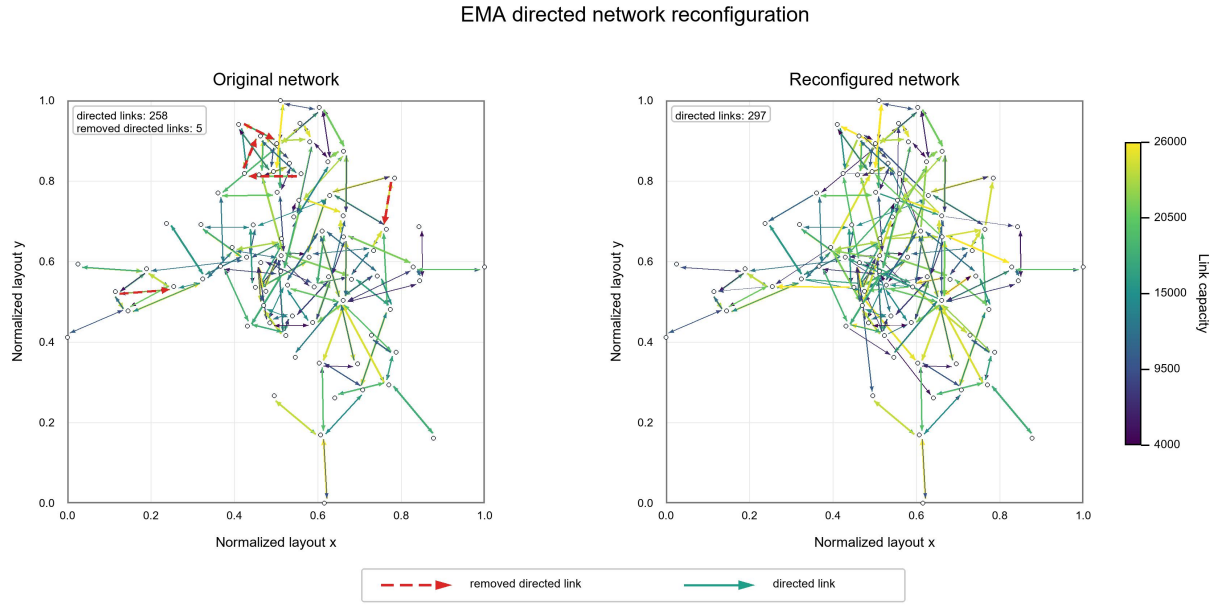

Figure 5.2: Example reconfiguration sample for the EMA network.

For both Sioux Falls and EMA, the processed datasets contain 10000 graph-pair samples. The split is 6000 training samples, 2000 validation samples, and 2000 test samples, corresponding to a 60%/20%/20% split. Feature and flow scalers are fitted on the training split and then applied to the validation and test splits.

5.1.2 Evaluation Metrics

All models are evaluated on edge-level flow prediction. Let $S \subseteq E'$ denote the set of evaluated edges, $\tilde{f}_e$ the normalized target flow, $\hat{\tilde{f}}_e$ the normalized prediction, and f_e, $\hat{f}_e$ the corresponding real-scale values after inverse normalization. The normalized RMSE is

$$\mathrm{RMSE}_{\mathrm{norm}}(S) = \sqrt{\frac{1}{|S|} \sum_{e \in S} \left(\hat{\tilde{f}}_e - \tilde{f}_e \right)^2}. \tag{5.1}$$

This metric is used for model selection because training is performed in normalized flow space.

The real-scale RMSE is computed after converting predictions back to physical flow units:

$$\mathrm{RMSE}_{\mathrm{real}}(S) = \sqrt{\frac{1}{|S|} \sum_{e \in S} \left(\hat{f}_e - f_e \right)^2}. \tag{5.2}$$

This reports the absolute error in the original flow scale.

The main scale-aware relative error metric is Weighted Mean Absolute Percentage Error (WMAPE):

$$\mathrm{WMAPE}(S) = \frac{\sum_{e \in S} |\hat{f}_e - f_e|}{\max\left(\sum_{e \in S} |f_e|, \epsilon\right)}. \tag{5.3}$$

WMAPE is preferred over plain MAPE because some true edge flows can be small. A per-edge percentage error can then become unstable, while WMAPE uses the aggregate true flow as the denominator. WMAPE is reported for all new-graph edges, retained edges, and newly added edges. The retained-edge set is $E_{\mathrm{ret}} = E' \cap E$, and the added-edge set is $E_{\mathrm{add}} = E' \setminus E$. The added-edge metric is reported separately because these edges have no direct historical flow observation and are central to the reconfiguration task.

A conservation metric evaluates whether predicted flows satisfy nodal flow conservation on the reconfigured graph. For node v, define

$$d_v = \sum_{e \in \mathrm{In}_G(v)} f_e^{old} - \sum_{e \in \mathrm{Out}_G(v)} f_e^{old}. \tag{5.4}$$

The residual on G' is

$$\rho_v = \sum_{e \in \mathrm{In}_{G'}(v)} \hat{f}_e - \sum_{e \in \mathrm{Out}_{G'}(v)} \hat{f}_e - d_v. \tag{5.5}$$

The Relative Conservation Violation (RelCon) for one graph is

$$\mathrm{RelCon} = \frac{\sum_{v \in V} |\rho_v|}{\max\left(\sum_{v \in V} |d_v|, \epsilon\right)}. \tag{5.6}$$

The reported value is averaged over test graphs, with the 95th percentile also recorded.

Inference efficiency is measured as forward-pass inference time per graph:

$$T_{\text{graph}} = \frac{T_{\text{forward}}}{N_{\text{graphs}}}. \tag{5.7}$$

It is reported in milliseconds per graph on the test split.

5.2 Overall Prediction Performance

5.2.1 Performance on Sioux Falls

Table 5.1 reports the Sioux Falls results. On Sioux Falls, our model ST-PINN GatedGCN gives the lowest error on all accuracy metrics. It has the lowest normalized RMSE, real RMSE, overall WMAPE, old-edge WMAPE, and new-edge WMAPE. It also has the lowest RelCon value. Among the baselines, the node-centric GNN is the strongest. Compared with this baseline, our model reduces normalized RMSE and real RMSE by 33.1%, overall WMAPE by 29.9%, and RelCon by 64.9%. The improvement is larger on retained edges than on newly added edges. Old-edge WMAPE falls from 17.60% to 12.07%, a relative reduction of 31.4%. New-edge WMAPE falls from 23.73% to 18.40%, a relative reduction of 22.5%. This pattern is expected. Retained edges have aligned old-flow observations, while newly added edges have no direct historical flow. Their target flows must be inferred from the reconfigured topology, updated attributes, and the redistribution pressure.

Table 5.1: Overall prediction performance on Sioux Falls.

Model	Norm. RMSE	Real RMSE	WMAPE	Old WMAPE	New WMAPE	RelCon	Time (ms)
Old-flow	0.5718	1938.67	54.96%	41.25%	144.01%	4.506	0.0090
MLP	0.419	1419.33	37.12%	29.70%	85.36%	2.146	0.033
Single-topology GatedGCN	0.253	858.30	21.57%	20.45%	28.80%	1.865	0.075
Node-centric GNN	0.220	744.74	18.42%	17.60%	23.73%	1.661	0.094
ST-PINN GatedGCN	**0.147**	**498.18**	**12.92%**	**12.07%**	**18.40%**	**0.583**	0.302

New-edge WMAPE is higher than old-edge WMAPE for every model. This shows that the added-link part of the task is harder than updating flows on existing links. The result also explains why local edge features alone are not enough. The MLP has the largest new-edge error, while the graph models reduce it by exchanging information through G'. Our model performs best because it combines topology-aware propagation with the old-flow state and a physical consistency term.

ST-PINN GatedGCN has a RelCon of 0.583, while the strongest baseline has 1.661. This indicates that our model produces flows that are closer to nodal balance with respect to the net demand inferred from f^{old}. The price is runtime. ST-PINN GatedGCN takes 0.302 ms per graph, compared with 0.094 ms for the node-centric GNN and 0.033 ms for the MLP. The added cost comes from topology-aware redistribution and physics-informed updates, but the absolute inference time remains below one millisecond per graph, which is a major speed-up compared to the SUE solver.

The convergence curves shown in Figure 5.3 and Figure 5.4 indicate stable learning with late improvement. ST-PINN GatedGCN reaches its best validation RMSE at

the final epoch. The node-centric GNN also keeps improving in late training, while the MLP changes little after early epochs. Figure 5.3 shows that the graph-based models reduce training loss more strongly than the MLP. ST-PINN GatedGCN continues to decrease late in training, which is consistent with its best validation point at the end of the run. Figure 5.4 shows a clear separation between the models. The MLP stays at the highest validation RMSE, the single-topology GatedGCN improves over it, the node-centric GNN is the strongest baseline, and ST-PINN GatedGCN reaches the lowest validation RMSE,and there is no clear sign of overfitting. The validation RMSE of ST-PINN GatedGCN keeps decreasing until the end of the training epoch.

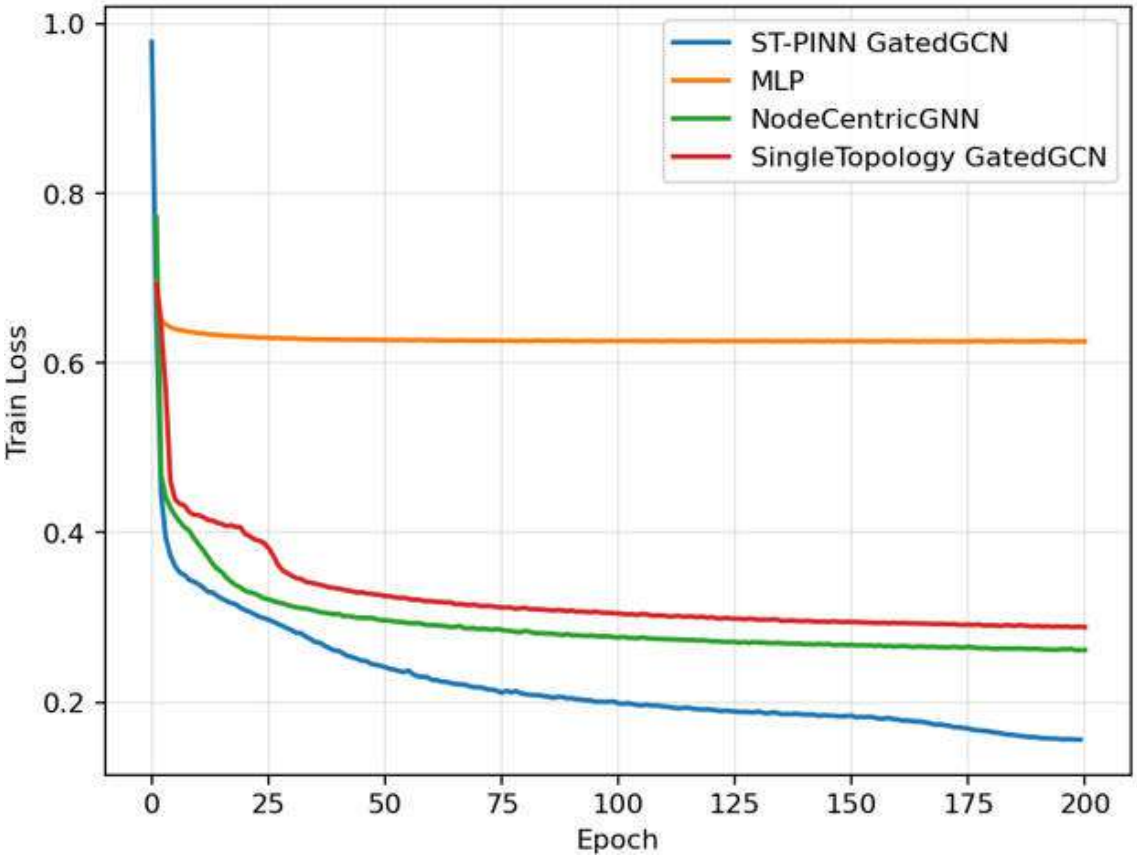

Figure 5.3: Overall training loss on Sioux Falls.

5.2.2 Performance on EMA Network

Table 5.2 reports the EMA results. The results on EMA network follow the same ranking as Sioux Falls. ST-PINN GatedGCN has the lowest evaluation matrics. The advantage of ST-PINN GatedGCN over the strongest baseline is larger on EMA than on Sioux Falls when measured by WMAPE. Compared with the node-centric GNN, ST-PINN GatedGCN reduces normalized RMSE and real RMSE by 36.2%, overall WMAPE by 34.0%, old-edge WMAPE by 34.3%, and new-edge WMAPE by 31.9%. The RelCon reduction is 31.2%, which is smaller than on Sioux Falls but still the

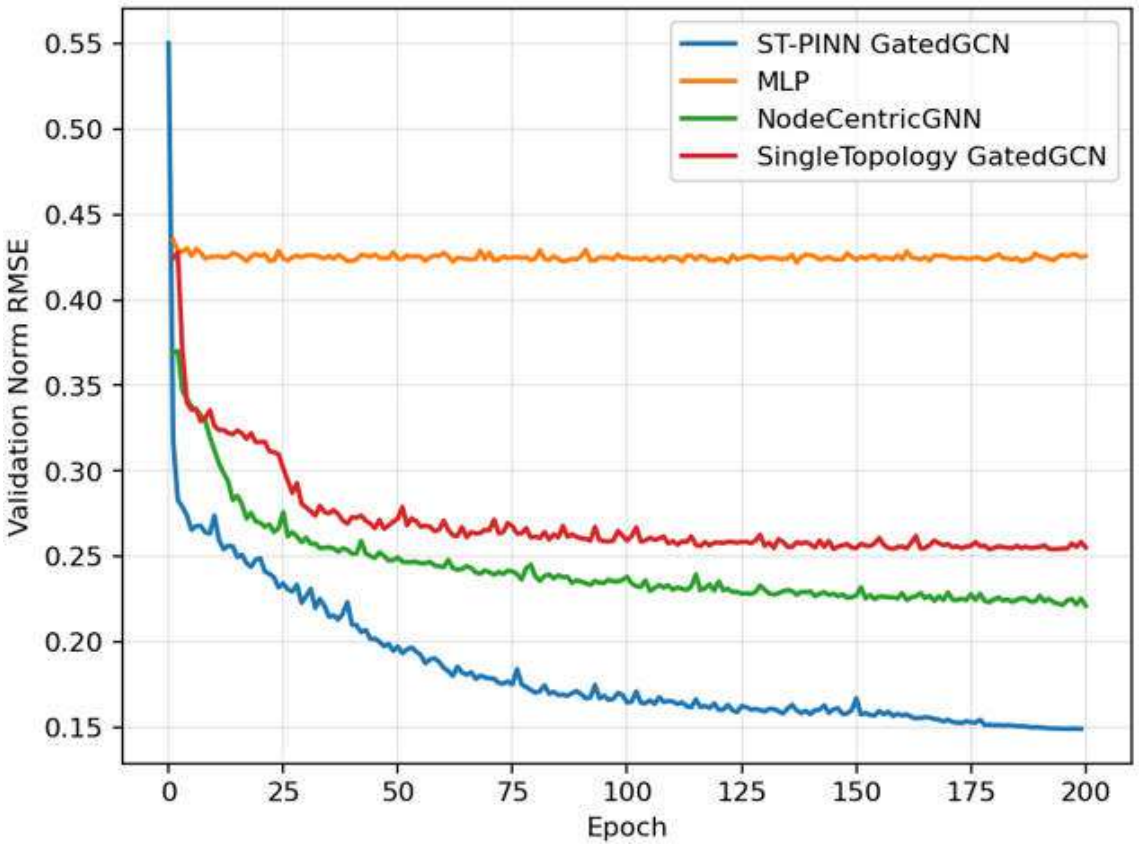

Figure 5.4: Overall validation RMSE on Sioux Falls.

best among the compared models. The real RMSE values are larger on EMA than on Sioux Falls, but the WMAPE values are lower. This is not contradictory. EMA has larger absolute flow values, so the same relative error can correspond to a larger error in physical flow units. For this reason, real RMSE is useful within one network, while WMAPE is more informative when comparing the two networks.

Table 5.2: Overall prediction performance on the EMA network.

Model	Norm. RMSE	Real RMSE	WMAPE	Old WMAPE	New WMAPE	RelCon	Time (ms)
Old-flow	0.6930	5596.51	27.20%	20.28%	75.03%	4.132	0.0214
MLP	0.449	3623.18	17.85%	16.57%	26.68%	4.360	0.035
Single-topology GatedGCN	0.313	2524.05	12.33%	11.96%	14.93%	3.385	0.078
Node-centric GNN	0.296	2393.75	11.87%	11.57%	13.93%	3.187	0.095
ST-PINN GatedGCN	**0.189**	**1527.27**	**7.84%**	**7.60%**	**9.49%**	**2.192**	0.312

New-edge prediction remains harder than old-edge prediction. For ST-PINN GatedGCN, old-edge WMAPE is 7.60%, while new-edge WMAPE is 9.49%. The gap is smaller than on Sioux Falls, but it has the same direction. This supports the same interpretation: added links require the model to infer new route usage without a direct old-flow value on the link.

The RelCon values are higher on EMA than on Sioux Falls for all models. This indicates that nodal consistency is harder to maintain on the larger network. ST-PINN GatedGCN still gives the lowest RelCon, which suggests that the physical

constraint remains useful when the number of nodes and edges increases. Runtime increases a little for our model, from 0.302 ms per graph on Sioux Falls to 0.312 ms per graph on EMA, indicating advantageous scaling with graph size.

The learning curves in Figure 5.5 and Figure 5.6 show stable training. ST-PINN GatedGCN reaches its best validation RMSE at epoch 185 and then remains close to that value. The MLP changes little after early training. The two graph baselines improve for longer, but their validation RMSE remains above our model. Figure 5.5 shows a larger initial loss range than on Sioux Falls. The graph-based models continue to reduce training loss after the MLP has mostly flattened. Figure 5.6 shows

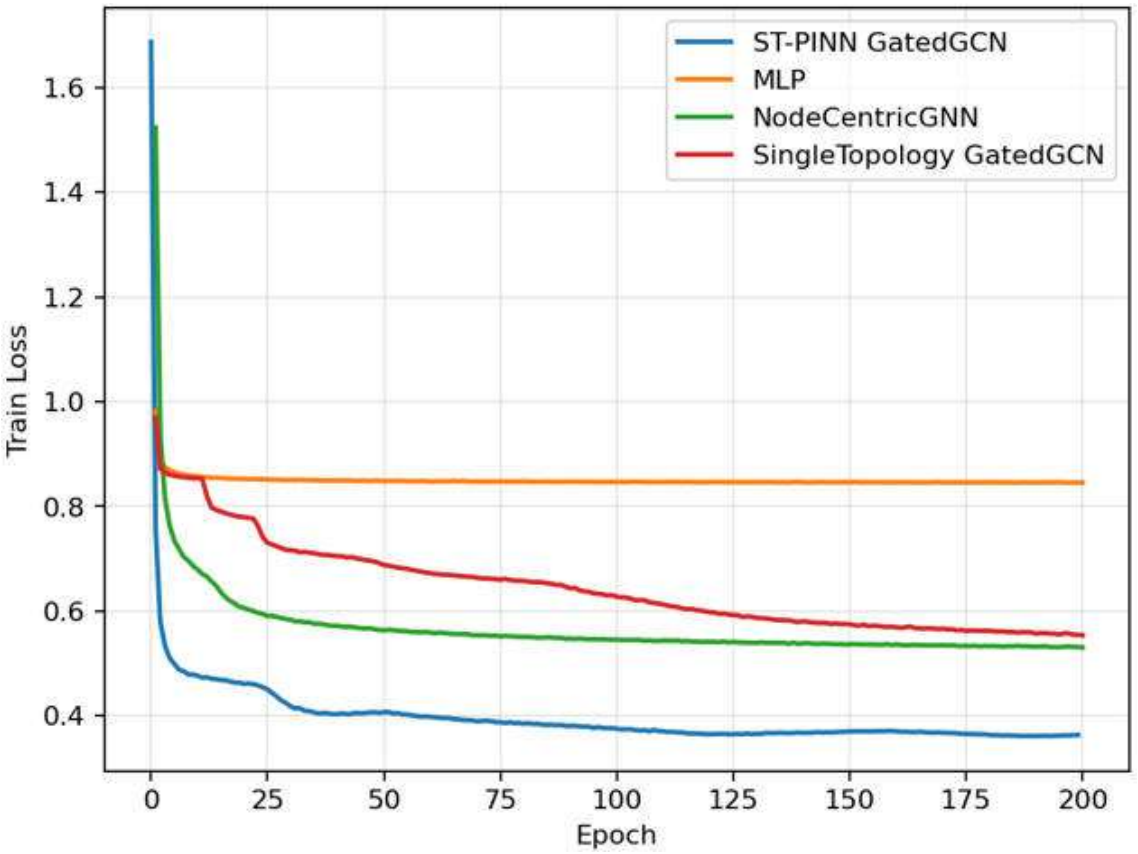

Figure 5.5: Overall training loss on the EMA network.

the same ordering as the final test metrics. ST-PINN GatedGCN reaches the lowest validation RMSE, while the node-centric GNN is the strongest baseline.

5.2.3 Performance by Edge Type

The edge-type analysis separates retained edges from newly added edges. This is needed because the all-edge average can hide different error mechanisms. Retained edges have a direct counterpart in G, while newly added edges only appear in G'. The latter case is a stricter test of whether the model can infer flow redistribution after reconfiguration.

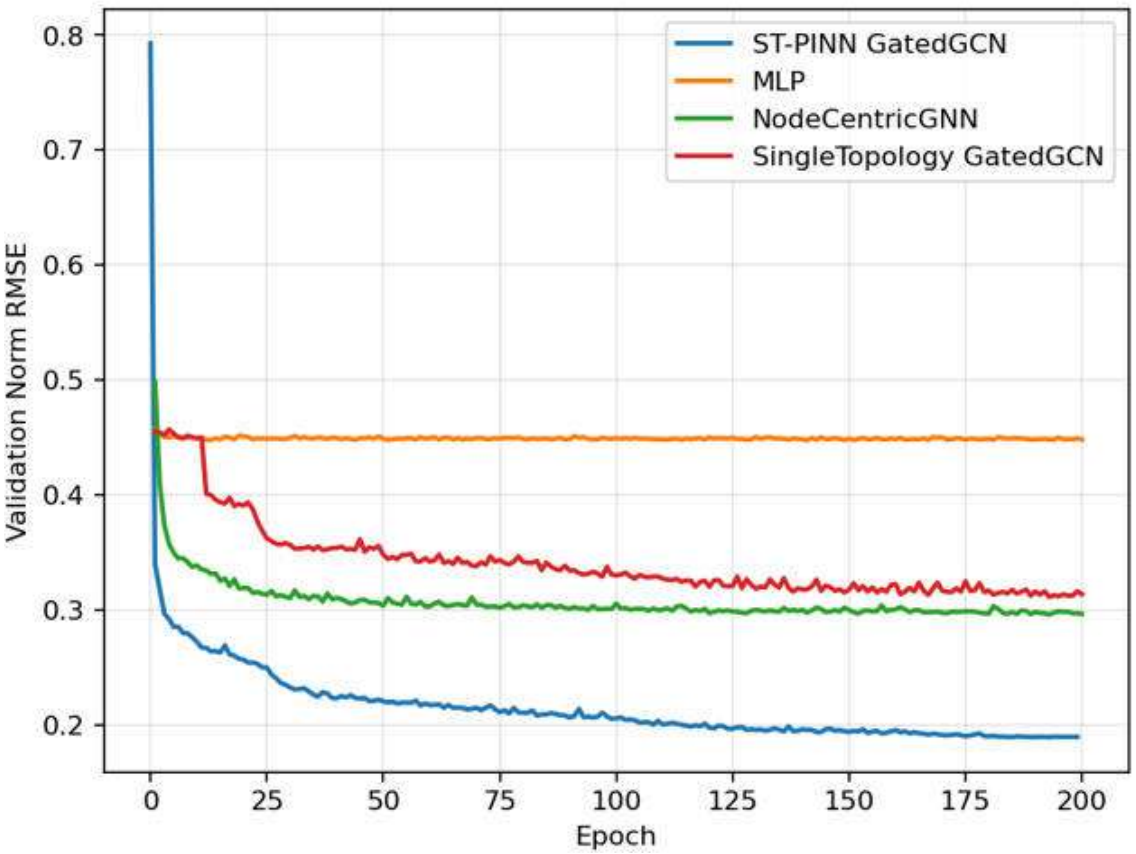

Figure 5.6: Overall validation RMSE on the EMA network.

5.2.3.1 Retained-Edge Flow Prediction

Retained edges are easier to predict because f^{old} provides a direct reference. This reference is not sufficient by itself, since the equilibrium on a retained edge may still change after links are added, removed, or modified elsewhere in the network. The results show that topology-aware models benefit from this old-flow information while still using G' to account for redistribution.

Table 5.3 reports improvement measures derived from the strongest model to our model. ST-PINN GatedGCN gives the strongest retained-edge performance on both networks. The consistent ranking suggests that using the old flow together with edge-level propagation is useful even when the edge itself is not changed.

Table 5.3: Relative retained-edge improvement of ST-PINN GatedGCN over the strongest baseline.

Network	Strongest baseline	RMSE reduction	WMAPE reduction	WMAPE gap
Sioux Falls	Node-centric GNN	34.8%	31.4%	5.53 pp
EMA network	Node-centric GNN	37.2%	34.3%	3.97 pp

Figure 5.7 highlights that the retained-edge gain is similar across the two networks. The improvement is slightly larger on the EMA network in relative term, although its absolute flow scale is larger.

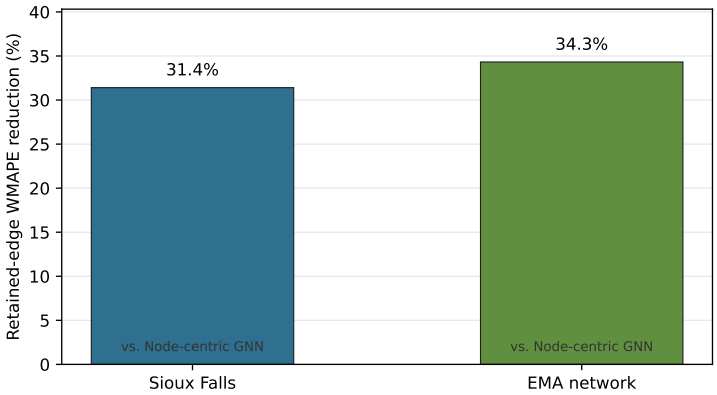

Figure 5.7: Relative retained-edge WMAPE reduction of ST-PINN GatedGCN over the strongest baseline.

5.2.3.2 Newly Added Edge Flow Prediction

Newly added edges are harder to predict because they have no direct old-flow observation. Their flows depend on whether they reduce route costs enough to attract traffic from other paths. Local attributes such as capacity and speed are therefore not enough. The model must reason over the reconfigured topology and the old traffic state.

The new-to-retained WMAPE ratio is used to summarize the additional difficulty of newly added edges:

$$R_{\text{new/ret}} = \frac{\text{WMAPE}_{S^{\text{new}}}}{\text{WMAPE}_{S^{\text{ret}}}}. \tag{5.8}$$

Table 5.4 shows that ST-PINN GatedGCN reduces new-edge error on both networks. The reduction is larger on the EMA network. The ratio remains above one in both cases, so newly added edge prediction is still harder than retained-edge prediction. This indicates that our model reduces, but does not remove, the main difficulty of reconfiguration.

Table 5.4: New-edge penalty and relative improvement of ST-PINN GatedGCN.

Network	Strongest baseline	New-edge WMAPE reduction	New-edge gap reduction	$R_{\text{new/ret}}$
Sioux Falls	Node-centric GNN	22.5%	5.33 pp	1.52
EMA network	Node-centric GNN	31.9%	4.44 pp	1.25

Figure 5.8 shows that the penalty is higher on Sioux Falls. This suggests that some added links in the smaller network cause sharper route-choice changes. The EMA network has a lower ratio, but newly added edges remain the more difficult subset.

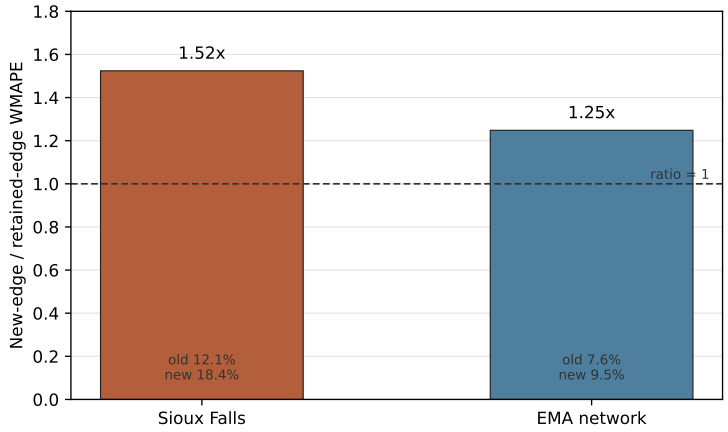

Figure 5.8: New-edge prediction penalty of ST-PINN GatedGCN, measured as the ratio between new-edge WMAPE and retained-edge WMAPE.

Overall, retained edges are easier because old-flow information is available. Newly added edges are harder because their flows must be inferred from network-level redistribution on G'. ST-PINN GatedGCN gives the most standout edge-type performance in the available results, but new-edge prediction remains the main bottleneck.

5.3 Physical Consistency of Predicted Flows

The previous section evaluated the models mainly as a predictor of edge flows. This section asks a complementary question: Does the predicted post-edit flow field satisfy the physical law on the reconfigured network. Here, the physical law means the predicted stationary flow should be close to node-level balance on the edited graph. This is weaker than verifying a full Wardrop equilibrium, but it is still essential for interpreting the output as a feasible post-edit traffic state. A model can obtain a low RMSE or WMAPE while still producing local excesses or deficits of flow at nodes.

The notation follows the above sections. Let the reconfigured graph be $G' = (V, E', A')$, and let $\hat{f}^{\text{new}}$ denote the predicted real-scale post-edit flow vector on E'. The inherited node-level net-demand proxy is computed from the observed pre-edit flow f^{old} on the original graph G:

$$d_v = \sum_{e \in \text{In}_G(v)} f_e^{\text{old}} - \sum_{e \in \text{Out}_G(v)} f_e^{\text{old}}. \tag{5.9}$$

Given a predicted post-edit flow vector $\hat{f}^{\text{new}}$, the node-level conservation residual on G' is

$$\rho_v(\hat{f}^{\text{new}}) = \sum_{e \in \text{In}_{G'}(v)} \hat{f}_e^{\text{new}} - \sum_{e \in \text{Out}_{G'}(v)} \hat{f}_e^{\text{new}} - d_v. \tag{5.10}$$

The reported relative conservation violation is

$$\text{RelCon} = \frac{\sum_{v \in V} |\rho_v(\hat{f}^{\text{new}})|}{\max\left(\sum_{v \in V} |d_v|, \epsilon\right)}. \tag{5.11}$$

A smaller RelCon therefore means that the predicted flows are closer to the inflow–outflow balance. In the following subsections, RelCon is used together with RMSE and WMAPE to separate edge-wise predictive accuracy from physical consistency.

5.3.1 Flow-Conservation Violation

Table 5.5 shows that ST-PINN GatedGCN has both the lowest prediction error and the lowest conservation violation on the two networks. On Sioux Falls, the proposed model reduces RMSE from 744.74 for the strongest baseline to 498.18, and RelCon from 1.661 to 0.583. This indicates that the model is not only fitting edge flows more accurately. It also produces a flow field that is more consistent with the node-level balance implied by f^{old}.

Table 5.5: Prediction accuracy and flow-conservation violation on the test set. RelCon measures the relative violation of the node-level flow-conservation constraint.

Network	Model	R^2	RMSE	WMAPE	Old WMAPE	New WMAPE	RelCon	RelCon p95	Time
Sioux Falls	Ours	**0.973**	**498.18**	**12.92%**	**12.07%**	**18.40%**	**0.583**	**0.975**	0.302
Sioux Falls	Node-centric GNN	0.939	744.74	18.42%	17.60%	23.73%	1.661	2.825	0.094
Sioux Falls	Single-topology GatedGCN	0.918	858.30	21.57%	20.45%	28.80%	1.865	3.063	0.075
Sioux Falls	MLP	0.777	1419.33	37.12%	29.70%	85.36%	2.146	3.505	**0.033**
EMA	Ours	**0.969**	**1527.27**	**7.84%**	**7.60%**	**9.49%**	**2.192**	**2.883**	0.312
EMA	Node-centric GNN	0.923	2393.75	11.87%	11.57%	13.93%	3.187	4.149	0.095
EMA	Single-topology GatedGCN	0.915	2524.05	12.33%	11.96%	14.93%	3.385	4.451	0.078
EMA	MLP	0.824	3623.18	17.85%	16.57%	26.68%	4.360	5.632	**0.035**

The baseline results show why the conservation metric is needed. The node-centric GNN has the strongest baseline accuracy, but its RelCon remains much higher than that of ours. This suggests that it captures a useful edge-flow pattern, but it does not enforce the same level of feasibility at the node level.

On the EMA network, all models have larger absolute RMSE than on Sioux Falls. This is expected because the flow scale is larger. RelCon is more useful for comparing physical consistency across networks, because it normalizes by the inferred net-demand magnitude. ST-PINN GatedGCN still has the lowest RelCon on EMA, but the value is higher than on Sioux Falls. The gap suggests that satisfying node balance is harder on the larger network, where more paths and OD interactions can contribute to each local residual.

5.3.2 Accuracy-Conservation Trade-Off

Table 5.6 reports the accuracy-conservation trade-off under different values of the conservation loss weight λ_{con}. The purpose of this comparison is not only to find the lowest WMAPE. It tests whether conservation regularization can reduce node-level imbalance while keeping edge-wise prediction accuracy close to its best value. The conservation loss weight is varied as

$$\lambda_{\text{con}} \in \{0, 0.01, 0.05, 0.1, 0.2\}. \tag{5.12}$$

On Sioux Falls, $\lambda_{\text{con}} = 0.2$ gives the lowest RMSE, the lowest WMAPE, and the lowest RelCon. This shows that a stronger conservation term does not damage

edge-wise accuracy on this network. It improves both prediction error and node-level balance. One probable reason is that Sioux Falls is small, so flow redistribution caused by topology changes is more concentrated. The conservation term can help the model avoid locally inconsistent flow assignments. The result for $\lambda_{\text{con}} = 0.01$ is worse than the case without conservation regularization. This indicates that a very small conservation weight does not necessarily improve the solution. It may be too weak to control imbalance while still changing the training dynamics.

On EMA, the lowest RMSE and WMAPE occur at $\lambda_{\text{con}} = 0.01$, whereas the lowest RelCon occurs at $\lambda_{\text{con}} = 0.1$. This gives a clearer accuracy-conservation trade-off. As λ_{con} increases from 0 to 0.1, RelCon decreases, but RMSE and WMAPE increase. When $\lambda_{\text{con}} = 0.2$, edge-wise accuracy decreases further, and RelCon no longer improves. This suggests that the larger EMA network is more sensitive to the strength of the conservation term. With more nodes, edges, and route alternatives, a strong conservation penalty may push the model toward node-balanced flows that are not always the most accurate edge-level predictions.

Table 5.6: Accuracy-conservation trade-off under different conservation weights.

Network	λ_{con}	RMSE	WMAPE	old WMAPE	new WMAPE	RelCon	RelCon p95
Sioux Falls	0	515.44	13.32%	12.46%	18.88%	0.690	1.109
Sioux Falls	0.01	563.12	14.53%	13.60%	20.59%	0.739	1.215
Sioux Falls	0.05	499.00	12.89%	12.06%	18.27%	0.616	1.010
Sioux Falls	0.1	503.16	13.04%	12.20%	18.47%	0.610	1.010
Sioux Falls	0.2	488.40	12.70%	11.89%	17.97%	0.547	0.941
EMA	0	1538.85	7.83%	7.56%	9.66%	2.434	3.168
EMA	0.01	1513.90	7.71%	7.45%	9.51%	2.352	3.073
EMA	0.05	1590.45	8.13%	7.85%	10.04%	2.216	2.909
EMA	0.1	1598.97	8.23%	7.96%	10.12%	2.160	2.829
EMA	0.2	1666.59	8.50%	8.18%	10.68%	2.319	3.036

Figure 5.9 shows the same pattern in the WMAPE–RelCon plane. The horizontal axis is RelCon, and the vertical axis is WMAPE, so the preferred region is the lower-left corner. For Sioux Falls, the points move roughly toward this region as λ_{con} increases. This means that conservation regularization helps both accuracy and feasibility in this case. For EMA, the points show a clearer compromise. The setting $\lambda_{\text{con}} = 0.01$ gives the best WMAPE, while $\lambda_{\text{con}} = 0.1$ gives the best RelCon. This indicates that a single conservation weight is not necessarily optimal across different network scales.

Overall, the conservation loss does not simply reduce numerical error. It changes which flow distributions the model prefers. On Sioux Falls, stronger conservation regularization improves both edge-wise accuracy and physical consistency. On EMA, a moderate value is needed to reduce imbalance without giving up too much edge-wise accuracy. The choice of λ_{con} should therefore be interpreted as a choice of physical constraint strength. If the main goal is edge-flow prediction, the setting

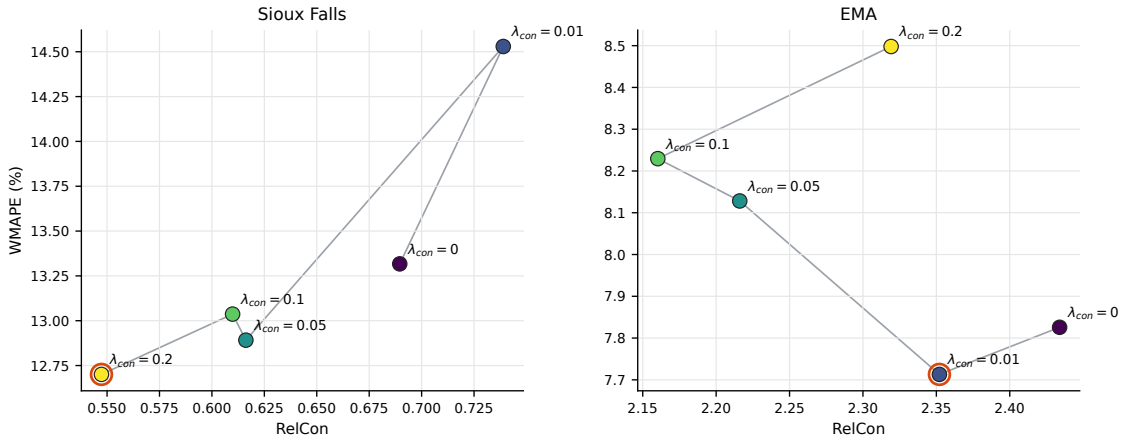

Figure 5.9: Accuracy-conservation trade-off under different values of λ_{con}.

with lower WMAPE is preferable. If the main goal is a more physically feasible flow field, the setting with lower RelCon is preferable.

5.3.3 Qualitative Examples of Flow Redistribution

Figure 5.10 gives an edge-level view of the redistribution task. The observed redistribution picture shows how the equilibrium flow changes after the network is reconfigured. The predicted redistribution picture shows the corresponding output of ST-PINN GatedGCN. The error one shows where the prediction differs most from the observed change.

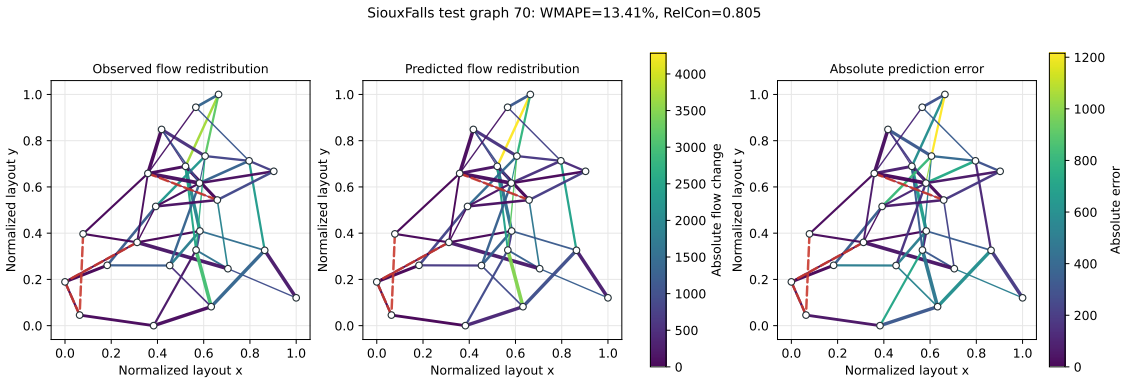

Figure 5.10: Qualitative example of flow redistribution on a Sioux Falls test graph. The first two pictures compare observed and predicted flow redistribution after reconfiguration, while the third one shows the absolute prediction error. Dashed red links indicate removed links.

The predicted redistribution follows the main spatial pattern of the observed redistribution. Roads with larger flow changes appear in similar parts of the network, rather than being placed uniformly across all links. This indicates that the model has learned more than a global scaling of old flows. It uses the reconfigured topology

and edge attributes to decide where flow should move after the change.

The removed links create local disruptions, but their effect is not limited to their endpoints. Traffic can shift to nearby connectors and to alternative corridors farther away. The qualitative example shows that the predicted pattern captures this non-local effect in several parts of the graph. This is important for reconfiguration modelling, because the main difficulty is often route substitution rather than the flow on an unchanged link.

Overall, ST-PINN GatedGCN gives lower prediction error and lower conservation violation than the baselines on both networks. The RelCon reduction indicates that the predicted flows are more consistent with node-level balance. The trade-off sweep further clarify how the conservation weight controls the balance between RMSE, WMAPE, and RelCon. The qualitative example shows that the model can recover the main flow redistribution pattern after a topology change.

5.4 Ablation Studies

The ablation study evaluates which parts of ST-PINN GatedGCN are responsible for prediction accuracy and physical consistency. Each ablation keeps the same training setup as the our full model and removes only one mechanism. The comparison is made on both Sioux Falls and EMA.

5.4.1 Ablation Setup

The full model is compared with six variants. The w/o old flow variant removes the old-flow warm start and sets the initial pressure to zero. The new attributes only variant removes old-network attributes from the alignment features, while keeping the old-flow warm start. The w/o residual injection variant removes the pressure residual input from edges and nodes. The w/o global message variant removes the global message channel. The unshared diffusion cells variant replaces the shared recurrent cell with separate cells for each diffusion step. The w/o recurrent pressure update variant keeps the initial pressure but does not update the pressure residual after each diffusion step.

5.4.2 Overall Ablation Results

Table 5.7 reports the test results. The comparison uses RMSE and WMAPE for edge-wise prediction accuracy, and RelCon for node-level physical consistency.
The removal of old flow gives the largest degradation on Sioux Falls. WMAPE increases from 12.92% to 18.92%, and RelCon increases from 0.583 to 1.344. Both old-edge WMAPE and new-edge WMAPE increase, from 12.07% to 18.31% and from 18.40% to 22.91%, respectively. This shows that the old flow is not just an additional local feature. It acts as a prior state for the pre-edit traffic condition. Without it, the model has to infer the flow level from topology and attributes alone. On EMA, the same ablation is also worse, but the increase is smaller. WMAPE

Table 5.7: Ablation results of ST-PINN GatedGCN on Sioux Falls and EMA.

Network	Variant	RMSE	WMAPE	old WMAPE	new WMAPE	RelCon	Params
Sioux Falls	Full model	498.18	12.92%	12.07%	18.40%	0.583	332459
Sioux Falls	w/o old flow	744.72	18.92%	18.31%	22.91%	1.344	332459
Sioux Falls	new attributes only	496.50	12.84%	12.14%	17.42%	0.540	332459
Sioux Falls	w/o residual injection	573.29	14.71%	13.77%	20.87%	0.947	332075
Sioux Falls	w/o global message	634.79	16.43%	15.56%	22.10%	0.728	134401
Sioux Falls	unshared diffusion cells	539.26	13.89%	12.94%	20.10%	0.642	1326380
Sioux Falls	w/o recurrent pressure update	498.98	12.84%	12.04%	18.04%	0.798	332459
EMA	Full model	1527.27	7.84%	7.60%	9.49%	2.192	332459
EMA	w/o old flow	1837.68	9.72%	9.72%	9.70%	2.274	332459
EMA	new attributes only	1796.03	9.20%	8.95%	10.90%	2.203	332459
EMA	w/o residual injection	2190.26	10.83%	10.54%	12.83%	2.854	332075
EMA	w/o global message	1739.79	8.85%	8.54%	10.99%	2.229	134401
EMA	unshared diffusion cells	1596.36	8.14%	7.81%	10.43%	2.209	1326380
EMA	w/o recurrent pressure update	2076.22	10.32%	10.00%	12.52%	2.747	332459

rises from 7.84% to 9.72%, and RelCon rises from 2.192 to 2.274. This suggests that old flow remains useful on the larger network, but the error is less obvious because there are more alternative paths.

The new attributes only result needs a more careful interpretation. On Sioux Falls, WMAPE is 12.84%, slightly below the full model value of 12.92%, and RelCon is also slightly lower. This does not mean that old-network information is useless. In this ablation, the old-flow warm start is still kept. The result instead suggests that, on the smaller network, the old-flow state carries more useful information than the old-attribute part of the alignment feature. EMA behaves differently. WMAPE increases from 7.84% to 9.20%, and new WMAPE increases from 9.49% to 10.90%. This indicates that the larger network depends more on old-new attribute alignment. When only new attributes are used, the model loses part of the before-after contrast that it can use to interpret capacity and topology changes.

Removing residual injection is one of the clearest tests of the physical component. On Sioux Falls, RelCon increases from 0.583 to 0.947, while WMAPE increases from 12.92% to 14.71%. On EMA, RelCon increases from 2.192 to 2.854, and WMAPE increases from 7.84% to 10.83%. The residual signal therefore affects both conservation and edge-flow accuracy. It tells the model whether the current flow estimate violates local inflow–outflow balance. Without this signal, the model can still regress edge flows, but it more easily produces node-level imbalance. The increase in new WMAPE is also clear, from 18.40% to 20.87% on Sioux Falls and from 9.49% to 12.83% on EMA. Added edges have no direct historical flow, so they rely more on this physical residual information.

Removing the global message channel also hurts prediction accuracy. On Sioux Falls, WMAPE increases from 12.92% to 16.43%. This is a large change for a single removal. The global channel helps spread non-local redistribution effects, which are important when a network edit affects a large share of available routes. On EMA, WMAPE increases to 8.85%, and new WMAPE increases from 9.49% to 10.99%. The effect is smaller than on Sioux Falls, but it is still visible, especially for added

edges. This suggests that EMA still benefits from non-local information.

The unshared diffusion cells variant shows that more parameters do not lead to better results here. The parameter count increases from 332459 to 1326380, but WMAPE rises from 12.92% to 13.89% on Sioux Falls and from 7.84% to 8.14% on EMA. RelCon also does not improve. This supports the use of a shared diffusion cell. The shared cell makes each diffusion step apply the same update rule, which is closer to an iterative flow-adjustment process in the reality. Unshared cells increase the ability to capture the features, but they weaken the recurrent consistency. The full model therefore benefits from a more suitable update structure rather than from a larger parameter setup.

The w/o recurrent pressure update variant separates edge-wise accuracy from physical consistency on Sioux Falls. Its WMAPE is 12.84%, slightly lower than the full model value, but RelCon increases from 0.583 to 0.798. Thus, the model can predict similar edge flows while producing less balanced node-level flow fields. On EMA, the same ablation is much worse on both dimensions. WMAPE increases from 7.84% to 10.32%, and RelCon increases from 2.192 to 2.747. This indicates that recomputing the residual after each diffusion step is more important on the larger network. EMA has longer-range redistribution patterns, and a fixed initial pressure does not describe how imbalance changes as flow is updated.

5.4.3 Impact on Prediction Accuracy and Physical Consistency

To compare the impact of each ablation experiment with our full model, the changes in WMAPE and RelCon are defined as

$$\Delta\text{WMAPE} = \text{WMAPE}_{\text{abl}} - \text{WMAPE}_{\text{full}}, \tag{5.13}$$

and

$$\Delta\text{RelCon} = \text{RelCon}_{\text{abl}} - \text{RelCon}_{\text{full}}. \tag{5.14}$$

For WMAPE, the reported changes are expressed in percentage points. Positive values mean that the ablation is worse than the full model. Negative values mean that the metric is lower than the full model.

Figure 5.11 summarizes the same results as relative changes from the full model. The WMAPE bars show that old-flow removal is the dominant accuracy loss on Sioux Falls, with ΔWMAPE = 6.01 percentage points. On EMA, the largest WMAPE increase comes from removing residual injection, with ΔWMAPE = 2.99 percentage points, followed by removing the recurrent pressure update, with ΔWMAPE = 2.48 percentage points. These bars show that the two networks fail in different ways when mechanisms are removed.
The RelCon bars emphasize the physical part of the model. On Sioux Falls, removing old flow gives the largest RelCon increase, with ΔRelCon = 0.762. Removing residual injection and recurrent pressure update also increases RelCon by 0.364 and

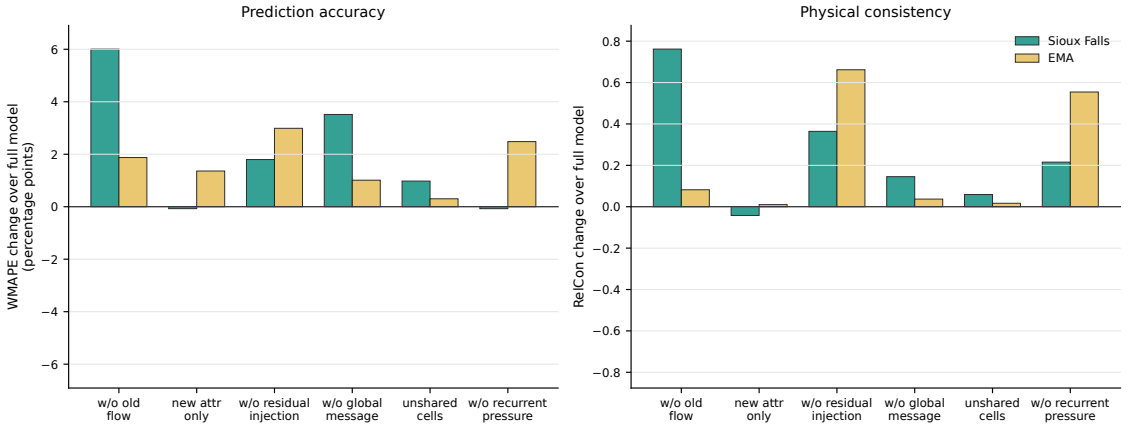

Figure 5.11: Ablation impact on WMAPE and RelCon relative to the full ST-PINN GatedGCN model. Positive values indicate worse performance than the full model.

0.215, respectively. On EMA, the strongest RelCon increases come from removing residual injection and removing recurrent pressure update, with $\Delta\text{RelCon} = 0.662$ and 0.554. These values show that residual-based updates are more closely tied to physical consistency on the larger network.

The negative bars are also informative. On Sioux Falls, the new attributes only variant has $\Delta\text{WMAPE} = -0.07$ percentage points and $\Delta\text{RelCon} = -0.043$. A possible explanation is that, on the smaller Sioux Falls network, the old-flow warm start already provides the dominant pre-edit state information. Adding old edge attributes may introduce redundant or noisy information, especially when the target is defined on the edited network. Removing those attributes can make the model rely more directly on the post-edit attributes and the old-flow state, which is enough for this smaller graph. The gain is small, so it should be interpreted as a result for this situation rather than as a general phenomenon that a model with only new attributes performs better.

Across networks, Sioux Falls is more sensitive to old-flow removal and global message removal. EMA is more sensitive to residual injection and recurrent pressure update. This suggests that the main difficulty on the smaller network is preserving continuity between the pre- and post-reconfiguration states. On the larger network, the main difficulty is maintaining and propagating the physical residual as flow moves across more route alternatives. The ablation ranking is therefore not the same on the two networks.

Overall, the ablation results show that the full model benefits from three kinds of information: the old-flow state, old-new alignment, and recurrent residual correction. The old-flow warm start is crucial for preserving the pre-edit traffic state, especially on Sioux Falls. Residual injection and recurrent pressure updates are most closely tied to physical consistency, especially on EMA. Shared diffusion cells improve robustness without increasing the parameter count. These results suggest

that ST-PINN GatedGCN should be understood as a structured predictor whose components work together, rather than as a collection of independent additions.

5.5 Failure Cases

The main failure case is prediction on newly added edges. These edges have no direct old-flow history. Their flows depend on how travelers redistribute on the new topology. This redistribution is influenced by the latent OD demand, which is not available to the model. Therefore, the higher new-edge errors are not only an optimization issue. They reflect missing information in the OD-free setting.

Network reconfiguration can also create non-local effects. A local edge addition or deletion may change route choices far away from the edited link. This makes the problem harder than copying old flows or applying a local correction. The model must infer how changes in G' affect flow patterns across the network. This is especially difficult when several route alternatives become available after the edit.

The RelCon results also need careful interpretation. A low WMAPE does not guarantee exact physical feasibility. RelCon measures node-level flow-conservation violation, but it does not test all conditions of a full stochastic user equilibrium. It predicts a flow field that is regularized by conservation.

The EMA results show that physical consistency can be harder to satisfy on a larger network. EMA has more links and more possible redistribution patterns than Sioux Falls. This can increase the difficulty of satisfying node-level balance while also minimizing edge-flow error. The λ_{con} sweep further shows that accuracy and conservation can conflict. The conservation weight therefore needs to be tuned rather than fixed only from intuition.

5.6 Summary

The main result of this chapter is that our model ST-PINN GatedGCN predicts reconfigured network flows with good accuracy. On Sioux Falls, the model obtains an RMSE of 498.18, WMAPE of 12.92%, and RelCon of 0.583. On EMA, it obtains an RMSE of 1527.27, WMAPE of 7.84%, and RelCon of 2.192. The higher RMSE on EMA does not mean that the relative prediction quality is worse. EMA has a larger absolute flow scale, so RMSE is not directly comparable across the two networks. WMAPE gives a more scale-aware comparison.

The edge-type results show that retained edges remain easier than newly added edges. This is expected because retained edges have old-flow values that act as a strong state anchor. New edges have no direct old-flow inputs. Their flows must be inferred from route redistribution after the topology change. This is reflected by the new-edge penalty, which is 1.524 on Sioux Falls and 1.248 on EMA. The larger penalty on Sioux Falls suggests that each topology change is more disruptive,

because fewer alternative routes are available in the smaller network.

The conservation-weight sweep shows that λ_{con} changes the balance between edge-wise accuracy and node-level feasibility. On Sioux Falls, $\lambda_{\text{con}} = 0.2$ gives both the best WMAPE and the lowest RelCon, with WMAPE of 12.70% and RelCon of 0.547. On EMA, the best WMAPE is obtained at $\lambda_{\text{con}} = 0.01$, while the lowest RelCon is obtained at $\lambda_{\text{con}} = 0.1$. The larger network therefore shows a clearer accuracy-conservation trade-off. The conservation term does not simply improve all metrics.

The ablation results support the same interpretation. Removing old flow is most harmful on Sioux Falls: WMAPE increases from 12.92% to 18.92%, and RelCon increases from 0.583 to 1.344. Removing residual injection hurts both networks, raising WMAPE from 12.92% to 14.71% on Sioux Falls and from 7.84% to 10.83% on EMA. Removing the recurrent pressure update is especially harmful on EMA, where WMAPE increases from 7.84% to 10.32% and RelCon increases from 2.192 to 2.747. The unshared diffusion-cell variant increases the parameter count from 332459 to 1326380, but it does not improve the results. This shows that the gain is not only due to model size. The useful components are those that preserve the old flow state, align old and new network information, and iteratively correct physical residuals.

The computational comparison is summarized using the mean time per graph. The speed-up is computed as the ratio between the mean SUE solve time and the mean ST-PINN GatedGCN inference time,

$$\text{Speed-up} = \frac{T_{\text{SUE}}}{T_{\text{ST-PINN}}}, \tag{5.15}$$

where T_{SUE} is the mean SUE solve time and $T_{\text{ST-PINN}}$ is the mean ST-PINN GatedGCN inference time.

Table 5.8: Summary of prediction accuracy, physical consistency, and computational efficiency.

Network	RMSE	WMAPE	RelCon	$T_{\text{ST-PINN}}$	T_{SUE}	Speed-up
Sioux Falls	498.18	12.92%	0.583	0.302 ms	59.42 ms	197×
EMA	1527.27	7.84%	2.192	0.312 ms	591.15 ms	1896×

The SUE runtime benchmark uses 2000 test graphs for each network, with 2000 successful solves in both cases. As shown in Table 5.8, we can see that on Sioux Falls, the mean SUE time is 59.42 ms per graph. On EMA, the mean SUE time is 591.15 ms per graph. In contrast, ST-PINN GatedGCN takes 0.302 ms per graph on Sioux Falls and 0.312 ms per graph on EMA. The resulting speed-up is about 197 times on Sioux Falls and about 1896 times on EMA. The speed-up is larger on EMA because SUE time grows strongly with network size, while the learned forward pass remains almost unchanged in this experiment.

The offline training cost of ST-PINN GatedGCN is reported in Table 5.9. The model was trained for 200 epochs on each network. Training took 26.6 minutes on Sioux Falls and 33.3 minutes on EMA, corresponding to 7.98 seconds and 9.99 seconds per epoch, respectively.

Table 5.9: Training time of ST-PINN GatedGCN.

Network	Epochs	Training time	Time per epoch
Sioux Falls	200	26.6 min	7.98 s
EMA	200	33.3 min	9.99 s

Together, the runtime and training-time results indicate that ST-PINN GatedGCN has a fast offline training cost and a very small online inference cost. This makes it useful for quickly screening many candidate network reconfiguration scenarios.

Overall, the results support ST-PINN GatedGCN as a fast and physically informed surrogate for OD-free flow prediction under network reconfiguration. The main remaining limitations are added-edge prediction and the choice of λ_{con}, since new edges require true redistribution reasoning and conservation tuning affects the balance between WMAPE and RelCon.

6
Discussion

6.1 Main Findings

6.1.1 RQ1: Methodological Gap

The main methodological gap is the full setting studied in this thesis. Classical traffic assignment can compute equilibrium flows on a given network. It usually requires an OD demand matrix and must be solved again for each reconfigured network. OD estimation addresses the missing-demand problem, but it always requires a fixed networks. Many graph surrogates for traffic assignment still use OD demand as an input. Physics-informed graph learning provides useful modelling ideas, but it does not directly define an OD-free reconfiguration predictor. This gap motivates the formulation used in this thesis. The aim of this thesis is to estimate how equilibrium flows redistribute from G to G' when f^{old} is observed but Q is hidden.

6.1.2 RQ2: OD-Free Flow-to-Flow Formulation

The OD-free reconfiguration problem can be formulated as a flow-to-flow learning problem. The observable input is

$$X = (G, G', f^{\text{old}}),$$

and the target is

$$Y = f^{\text{new}}.$$

The learning problem is therefore to learn a statistical mapping from the observed old flow and the graph change to the post-edit flow on E'.

This formulation is important because the OD-free problem is not a deterministic inverse problem in general. Different OD matrices may produce similar f^{old} on G, but different f^{new} on G'. The model should therefore be interpreted as a predictor under partial information.

6.1.3 RQ3: Predictive Performance

Our model ST-PINN GatedGCN predicts post-edit flows more accurately than the learning-based baselines. It achieves the best WMAPE and RMSE on both Sioux Falls and EMA. The WMAPE is 12.92% on Sioux Falls and 7.84% on EMA.

This result matters because the model is not only performing independent edge regression. It uses edge alignment to transfer old-flow information from E to E'. It then uses GatedGCN pseudo-time diffusion to propagate information over the reconfigured graph. The physical residual terms and conservation loss guide the prediction toward node-level balance.

The results support the use of ST-PINN GatedGCN as a fast surrogate. The propose of designing this model is not to replaces the SUE. The model does not solve a full traffic assignment problem during inference. It approximates the SUE labels learned from the training distribution.

6.1.4 RQ4: Error and Physical Consistency

Prediction error and physical consistency vary with edge type, network scale, and model components. Retained edges are easier to predict because they have old-flow history. Newly added edges are harder because they have no direct value in f^{old}. Their flows depend more strongly on latent route choice and the hidden OD demand Q.

The two benchmark networks also behave differently. EMA has lower relative flow error, but its RelCon is higher than Sioux Falls. This suggests that larger networks can still be harder in terms of node-level physical consistency. The λ_{con} sweep also shows that accuracy and conservation are linked. In some cases they support each other. In other cases a stronger conservation penalty can increase edge-flow error.

These results mean that ST-PINN GatedGCN should be judged by both prediction accuracy and physical consistency. WMAPE and RMSE describe edge-flow error. RelCon describes node-level balance. The limitation is that RelCon does not verify full Wardrop or stochastic user equilibrium conditions.

6.2 Implications

The first implication concerns computational use. A SUE solver must be run again for each reconfigured graph. A trained graph surrogate only requires a forward pass. In the experiments, ST-PINN GatedGCN was about 197 times faster than SUE on Sioux Falls and about 1896 times faster on EMA. This makes the model better suited to rapid what-if screening than to replace high-precision assignment.

The second implication concerns physical consistency. The physics-informed terms do not make the model a full traffic assignment solver. They provide a regularizing signal that constraint the flow fields to physical space. This matters because low WMAPE or RMSE alone does not ensure that predicted flows form a feasible traffic state. Prediction accuracy and RelCon should therefore be interpreted together.

The third implication follows from the well-posedness analysis. The OD-free task is generally not a deterministic inverse recovery problem because Q is hidden. The

same observable input may be derived from more than one latent demand pattern. Statistical prediction remains well defined. The model should therefore be understood as learning $\mathbb{E}[f^{\text{new}} \mid G, G', f^{\text{old}}]$, or the corresponding conditional location functional under the training loss, rather than recovering the true OD demand.

6.3 Limitations

The main limitation is also part of the problem setting. OD demand is not observed. This creates irreducible uncertainty. Different latent matrices Q can produce similar old flows on G, but different post-reconfiguration flows on G'. No model can remove this uncertainty without extra information.

A second limitation is the topology setting within each benchmark. For a given network, the pre-edit graph G share the same base topology across samples. Edge attributes, flows, and reconfigurations vary, but the original network itself is fixed. This focuses the experiments on network reconfiguration. It also limits conclusions about topology-independent generalization.

A third limitation is that physical consistency is only partially measured. RelCon evaluates node-level flow-conservation violation. It does not cover all conditions of a full equilibrium assignment. The model does not explicitly solve route choice equilibrium at inference time.

6.4 Future Work

Future work should place more emphasis on uncertainty estimation. The OD-free setting contains conditional uncertainty because Q is hidden. It would therefore be useful to predict intervals or distributions for f^{new}, rather than only point estimates. Another option is to allow the model access to partial OD information, which might be available in realistic settings

A second direction is broader topology generalization. Future datasets should include more base networks and more diverse reconfiguration policies. This would allow a clearer test of whether the learned mapping transfers across network layouts, rather than only across scenarios generated from the same base graph.

A third direction is real-world validation. The method should be tested on observed traffic counts before and after real interventions. This would require careful treatment of temporal variation, demand changes, and measurement noise.

A fourth direction is stronger sensitivity testing. The model should be evaluated under larger topology edits, different capacity changes, different speed changes, and shifted demand distributions. This would clarify when the surrogate can be useful and when a full assignment solve is needed.

A final direction is to study downstream tasks. One example is the identification of Braessian edges. These are edges whose removal can improve equilibrium network performance, in the sense associated with Braess' paradox [40].

Bibliography

[1] E. I. Vlahogianni, M. G. Karlaftis, and J. C. Golias, "Short-term traffic forecasting: Where we are and where we're going," *Transportation Research Part C: Emerging Technologies*, vol. 43, pp. 3–19, 2014.

[2] K. Nellore and G. P. Hancke, "A survey on urban traffic management system using wireless sensor networks," *Sensors*, vol. 16, no. 2, p. 157, 2016.

[3] Y. Zheng, Y. Liu, J. Yuan, and X. Xie, "Urban computing with taxicabs," in *international conference on Ubiquitous computing*, 2011, pp. 89–98.

[4] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *in International Conference on Learning Representations*, 2018.

[5] J. G. Wardrop, "Road paper. some theoretical aspects of road traffic research." *Proceedings of the institution of civil engineers*, vol. 1, no. 3, pp. 325–362, 1952.

[6] Y. Sheffi, *Urban transportation networks*. Prentice-Hall, Englewood Cliffs, NJ, 1985, vol. 6.

[7] C. F. Daganzo and Y. Sheffi, "On stochastic models of traffic assignment," *Transportation science*, vol. 11, no. 3, pp. 253–274, 1977.

[8] M. G. Bell, "The estimation of an origin-destination matrix from traffic counts," *Transportation Science*, vol. 17, no. 2, pp. 198–217, 1983.

[9] M. L. Hazelton, "Inference for origin–destination matrices: estimation, prediction and reconstruction," *Transportation Research Part B: Methodological*, vol. 35, no. 7, pp. 667–676, 2001.

[10] Y. Cao, H. van Lint, P. Krishnakumari, and M. Bliemer, "Data driven origin–destination matrix estimation on large networks—a joint origin–destination-path-choice formulation," *Transportation Research Part C: Emerging Technologies*, vol. 168, p. 104850, 2024.

[11] T. Liu and H. Meidani, "End-to-end heterogeneous graph neural networks for traffic assignment," *Transportation Research Part C: Emerging Technologies*, vol. 165, p. 104695, 2024.

[12] M. Beckmann, C. B. McGuire, and C. B. Winsten, "Studies in the economics of transportation," Tech. Rep., 1956.

[13] M. Patriksson, "On the convergence of descent methods for monotone variational inequalities," *Operations Research Letters*, vol. 16, no. 5, pp. 265–269, 1994.

[14] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

[15] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International conference on machine learning*. PMLR, 2017, pp. 1263–1272.

[16] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia, "Graph networks as learnable physics engines for inference and control," in *International conference on machine learning*. PMLR, 2018, pp. 4470–4479.

[17] X. Bresson and T. Laurent, "Residual gated graph convnets," *arXiv preprint arXiv:1711.07553*, 2017.

[18] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational physics*, vol. 378, pp. 686–707, 2019.

[19] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," *Nature Reviews Physics*, vol. 3, no. 6, pp. 422–440, 2021.

[20] V. Sharma and O. Fink, "A physics-informed graph neural network conserving linear and angular momentum for dynamical systems," *Nature Communications*, 2026.

[21] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," *in International Joint Conference on Artificial Intelligence*, 2018.

[22] J. Xue, R. Tan, J. Ma, and S. V. Ukkusuri, "Data science in transportation networks with graph neural networks: a review and outlook," *Data Science for Transportation*, vol. 7, no. 2, p. 10, 2025.

[23] T. Liu and H. Meidani, "Multi-class traffic assignment using multi-view heterogeneous graph attention networks," *Expert Systems with Applications*, vol. 286, p. 128072, 2025.

[24] E. J. Pebesma and G. B. Heuvelink, "Latin hypercube sampling of gaussian random fields," *Technometrics*, vol. 41, no. 4, pp. 303–312, 1999.

[25] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM journal on computing*, vol. 1, no. 2, pp. 146–160, 1972.

[26] R. Z. Farahani, E. Miandoabchi, W. Y. Szeto, and H. Rashidi, "A review of urban transportation network design problems," *European journal of operational research*, vol. 229, no. 2, pp. 281–302, 2013.

[27] O. B. Lassen, S. Agriesti, M. Eldafrawi, D. Gammelli, G. Cantelmo, G. Gentile, and F. C. Pereira, "Learning traffic flows: Graph neural networks for metamodelling traffic assignment," in *International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*. IEEE, 2025, pp. 1–8.

[28] X. Bresson and T. Laurent, "An experimental study of neural networks for variable graphs," International Conference on Learning Representations (ICLR) Workshop Track, 2018.

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, 2017.

[30] H. W. Engl, M. Hanke, and A. Neubauer, *Regularization of inverse problems*. Springer Science & Business Media, 1996, vol. 375.

[31] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[33] N. Shazeer, "Glu variants improve transformer," *arXiv preprint arXiv:2002.05202*, 2020.

[34] B. Zhang and R. Sennrich, "Root mean square layer normalization," *in Advances in neural information processing systems*, 2019.

[35] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *in International Conference on Learning Representations*, 2019.

[36] K. Wen, Z. Li, J. Wang, D. Hall, P. Liang, and T. Ma, "Understanding warmup-stable-decay learning rates: A river valley loss landscape view," in *International Conference on Learning Representations*, 2025.

[37] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[38] T. Akamatsu, "Cyclic flows, markov process and stochastic traffic assignment," *Transportation Research Part B: Methodological*, vol. 30, no. 5, pp. 369–386, 1996.

[39] M. J. Lighthill and G. B. Whitham, "On kinematic waves ii. a theory of traffic flow on long crowded roads," *Proceedings of the royal society of london. series a. mathematical and physical sciences*, vol. 229, no. 1178, pp. 317–345, 1955.

[40] D. Braess, A. Nagurney, and T. Wakolbinger, "On a paradox of traffic planning," *Transportation science*, vol. 39, no. 4, pp. 446–450, 2005.

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY