CENTER FOR CONNECTED AND AUTOMATED TRANSPORTATION

Final Report #65
September 2023

AI-enabled Transportation Network Analysis, Planning and Operations

Yafeng Yin, PhD
Zhichen Liu

CIVIL & ENVIRONMENTAL ENGINEERING
UNIVERSITY OF MICHIGAN

CENTER FOR CONNECTED AND AUTOMATED TRANSPORTATION

Report No. 65

September 2023

Project Start Date: 1/1/2022

Project End Date: 8/31/2023

AI-enabled Transportation Network Analysis, Planning and Operations

Yafeng Yin

Zhichen Liu

University of Michigan

CENTER FOR CONNECTED AND AUTOMATED TRANSPORTATION

DISCLAIMER

Funding for this research was provided by the Center for Connected and Automated Transportation under Grant No. 69A3551747105 of the U.S. Department of Transportation, Office of the Assistant Secretary for Research and Technology (OST-R), University Transportation Centers Program. The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated under the sponsorship of the Department of Transportation, University Transportation Centers Program, in the interest of information exchange. The U.S. Government assumes no liability for the contents or use thereof.

Suggested APA Format Citation:

Yin, Y., & Liu, Z. (2023). AI-enabled Transportation Network Analysis, Planning and Operations. Final Report.
DOI: 10.7302/8099

Contacts

For more information:

PI Name: Yafeng Yin
University: University of Michigan
Address: 2120 GG Brown, Ann Arbor, Michigan
Phone Number: (734) 764-8249
Email Address: yafeng@umich.edu
Web Address: https://limos.engin.umich.edu/

CCAT
University of Michigan Transportation Research Institute
2901 Baxter Road
Ann Arbor, MI 48152
umtri-ccat@umich.edu
(734) 763-2498

UMTRI | The University of Akron | Illinois Center for Transportation | Purdue University | Washtenaw Community College | Central State University

CENTER FOR CONNECTED AND AUTOMATED TRANSPORTATION

Technical Report Documentation Page

1. Report No. CCAT Report No. 65	**2. Government Accession No.** Leave blank – not used	**3. Recipient's Catalog No.** Leave blank - not used
4. Title and Subtitle AI-enabled Transportation Network Analysis, Planning and Operations DOI: 10.7302/8099		**5. Report Date** August 31, 2023 **6. Performing Organization Code** Enter any/all unique numbers assigned to the performing organization, if applicable.
7. Author(s) Yafeng Yin, Ph.D.: https://orcid.org/0000-0003-3117-5463 Zhichen Liu: https://orcid.org/0000-0001-6178-9883		**8. Performing Organization Report No.** Enter any/all unique alphanumeric report numbers assigned by the performing organization, if applicable.
9. Performing Organization Name and Address Center for Connected and Automated Transportation University of Michigan Transportation Research Institute 2901 Baxter Road Ann Arbor, MI 48109 Department of Civil and Environmental Engineering University of Michigan 2350 Hayward Street Ann Arbor, MI, 48109		**10. Work Unit No.** **11. Contract or Grant No.** Contract No. 69A3551747105
12. Sponsoring Agency Name and Address U.S. Department of Transportation Office of the Assistant Secretary for Research and Technology 1200 New Jersey Avenue, SE Washington, DC 20590		**13. Type of Report and Period Covered** Final Report (January 2022 – August 2023) **14. Sponsoring Agency Code** OST-R
15. Supplementary Notes Conducted under the U.S. DOT Office of the Assistant Secretary for Research and Technology's (OST-R) University Transportation Centers (UTC) program.		
16. Abstract This study introduces a unified end-to-end framework for analyzing network traffic equilibrium. The framework learns supply and demand components directly from traffic data, using computational graphs to parameterize unknown elements. It enforces user equilibrium through variational inequalities and can incorporate various modeling approaches, including neural networks. A novel neural network architecture is proposed that guarantees equilibrium states and allows for future scenario planning. The model is trained using advanced gradient descent algorithms and leverages operator-splitting methods for solving variational inequality problems. The framework's effectiveness is confirmed through tests on three synthesized datasets.		

UMTRI · The University of Akron · Illinois Center for Transportation · Purdue University · Washtenaw Community College · Central State University

CENTER FOR CONNECTED AND AUTOMATED TRANSPORTATION

17. Key Words	18. Distribution Statement		
Network equilibrium, end-to-end learning, computational graph, and auto-differentiation	No restrictions.		
19. Security Classif. (of this report)	**20. Security Classif. (of this page)**	**21. No. of Pages**	**22. Price**
Unclassified	Unclassified	62	Leave blank – not used

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

UMTRI | The University of Akron | Illinois Center for Transportation | Purdue University | Washtenaw Community College | Central State University

Abstract

This study presents a unified end-to-end framework for network equilibrium analysis framework. The end-to-end framework directly learns model supply- and demand-side components and equilibrium states from multi-day traffic state observations. It parametrizes unknown model components with computational graphs and embeds them in a variational inequality to enforce user equilibrium conditions. Each component can be model-based, model-free (i.e., neural network), or hybrid. By minimizing the differences between the estimated and observed traffic states, the framework simultaneously estimates the unknown parameters for supply- and demand-sides.

Our study addresses key challenges in modeling and calibrating the unified end-to-end framework. We identify a novel neural network architecture that guarantees the existence of equilibrium traffic states and accommodates the potential changes in the road network topology for future what-if planning analysis. To train the model effectively, we leverage the computational power of computational graphs and design auto-differentiation-based gradient descent algorithms to handle both link- or path-based user equilibrium constraints. In forward propagation, we adopt recent developments in operator-splitting methods and differential optimization to solve a batch of VI problems. In backpropagation, iterated differentiation and implicit differentiation techniques are used to efficiently differentiate through the equilibrium states. The proposed framework and findings are validated using three synthesized datasets.

Table of Contents

List of Figures

List of Tables

1 Introduction

Transportation network equilibrium modeling paradigm plays an important role in the planning and operations of transportation networks. It has been widely used to compare different improvement designs or operation plans and aid the decision-making in selecting a better one for implementation. The paradigm was initiated by Beckmann et al. (1956) for modeling route choices in a static and deterministic network. Over the past 66 years, it has been extended to model other travel choices (e.g., destination and mode), better represent travel behaviors (e.g., bounded rationality) and capture traffic dynamics (within-day or day-to-day). Static network equilibrium models consist of two key elements: supply-side *link performance functions*, which determine the travel time based on link flow, and demand-side travel choice models, which describe the relationship between travelers' choices and travel costs. The latter typically include *demand functions*, which determine the travel demand between origin-destination (OD) pairs, and *cost functions*, which encapsulate travelers' choice preferences. The cost function can be determined once a behavior model is selected, while the link performance functions, and demand functions are usually calibrated separately using empirical data.

To improve the generalizability and accuracy of the equilibrium modeling framework, continued efforts have been devoted for over half a century to enhancing the representation of the supply- and demand-side components. For recent reviews of attempts at improving the behavioral realism of equilibrium models, refer to Xu et al. (2011) and Kitthamkesorn and Chen (2013), as an example. Despite the considerable progress in model refinement, previous frameworks have been constructed using a *"bottom-up" assembly approach*. Specifically, the process starts with adopting a particular assumption on how travelers make their travel choices (trip generation, destination, mode, route, and/or departure time) over a congestible network. By viewing the interactions among travelers as a non-cooperative non-atomic game, modelers describe the outcome of these interactions, i.e., the traffic flow distribution, as Wardrop user equilibrium (Nash equilibrium with infinitely many players) where no traveler would be better off by unilaterally changing their travel choice (Wardrop, 1952). This

equilibrium condition is then mathematically defined and subsequently formulated as an equivalent mathematical program or variational inequality (alternatively fixed point or nonlinear complementarity problem). Lastly, the formulation is solved to obtain the equilibrium flow distribution, from which various performance measures can be quantified.

The bottom-up assembly approach is justified when each model component can be properly determined and calibrated individually, which, unfortunately, is not the case. For one thing, it is difficult to obtain empirical OD demand data and even more so to properly calibrate a demand function due to the endogeneity problem (Zhang et al., 2017). It is also very challenging to properly specify and calibrate a link performance function because congestion does not persist as a steady state in practice. The appropriate specification of a link performance function depends on the underlying dynamics, which is often unobservable to modelers (Small and Chu, 2003; Cheng et al., 2022). Lastly, the travelers' utility functions are also unobservable and behavior models are selected based on modelers' beliefs or judgments. And the selected models are usually far from perfect for representing travel choice preferences (Chen et al., 2016). Recall that different supply- and demand-side components will lead to different equilibrium conditions and, consequently, different traffic flow distributions. Therefore, observed flows should inform the selection of model components in constructing the equilibrium analysis framework. However, we have never done so due to the absence of selection methodologies and the lack of empirical data.

Overall, the main limitation of the bottom-up assembly approach is that the selection of individual components is separated from the ultimate goal of a network equilibrium model: prescribing a equilibrium flow distribution that matches empirical observations as closely as possible. To overcome these limitations of the traditional "bottom-up" approach, this study aims to transform the modeling paradigm via an end-to-end framework that directly learns model components and the equilibrium state from data. As illustrated in Figure 1, the unified end-to-end framework encodes the unknown supply- and demand-side model components with parameterized computational graphs and then

embeds them in a VI that enforces user equilibrium conditions. In forward propagation (indicated by green arrows), the framework iteratively updates the traffic state via closed-form rules until reaches user equilibrium. In backpropagation (indicated by blue arrows), the loss function compares the estimated and observed traffic states and parameters θ for all components are simultaneously estimated via auto-differentiation.

The unified framework simultaneously integrates model-based and model-free modeling approaches within a single pipeline, leveraging both domain knowledge and the representational power of neural networks. In a model-free approach, we approximate the unknown component functions with neural networks and let the learning process automatically discover a good functional specification from empirical data. This end-to-end framework aligns the selection of behavior models with the ultimate goal of replicating flow distributions. Moreover, the end-to-end framework learns an equilibrium state of the network, even when real systems never truly reach equilibria and observed flows are not in equilibrium states. The learned equilibrium states then served as the benchmark to prescribe and compare different design plans. Additionally, the end-to-end framework integrates multi-source data into a single stream and addresses the inconsistencies among different data sources.

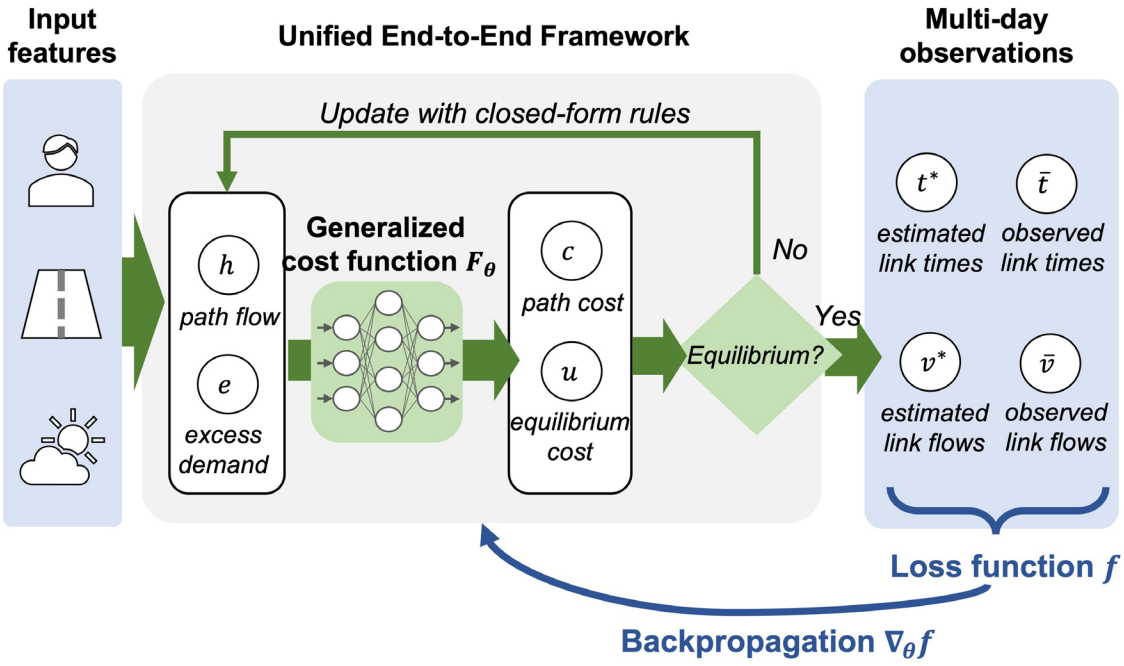


Figure 1 Illustration of the unified end-to-end framework.

To the best of our knowledge, our work is the first to integrate the learning of supply- and demand-side models in network equilibrium models into an end-to-end learning framework, with neural networks automatically discovering a good specification of route choice preferences from empirical data. This report presents our attempt to overcome the modeling and algorithmic challenges for enabling such an end-to-end learning. In a model-free approach, we identify a novel neural network architecture that guarantees the existence of an equilibrium solution and accommodates the changes in the road network topology that may arise in subsequent ”what-if” planning analysis. For training, we adopt recent developments in operator-splitting methods and differential programming to enable scalable solution algorithms for a batch of VI problems in forward propagation. In backpropagation, iterated differentiation and inexact implicit differentiation are used to the proposed framework to efficiently differentiate through the equilibrium states.

The rest of this report is organized as follows. Section 2 provides a literature review to better position this study. Section 3 formulates the elastic user equilibrium model as a VI parameterized by computational graphs. Section 4 presents the unified end-to-end framework, along with how to design neural network architecture design in a model-free approach. Section 5 details the auto-differentiation-based gradient descent method for training. Numerical experiments conducted on three synthesized datasets are reported in Section 6 to validate the proposed framework. This is a collaborative work with Fan Bai, General Motors Research and Development, and Donald K Grimm, General Motors Research and Development.

2 Background

2.1 Traffic flow prediction from observations

In the traditional ”bottom-up” network modeling paradigm, one can calibrate behavioral parameters in an equilibrium model from flow observations and then predict traffic flows

at Wardrop equilibrium. The calibration process is usually formulated as a bi-level program or a mathematical program with equilibrium constraints. For example, Yang et al. (2001) considered a logit-based stochastic user equilibrium and formulated a bi-level program to calibrate the dispersion parameter in the logit model and OD demands from link flow observations. Later studies extend such a calibration framework to accommodate more complex model structures. Wang et al. (2016) considered a dynamic dispersion parameter and performed experiments using real-world data gathered from a small network in Seattle, WA. Guarda and Qian (2022) considered a multi-criteria linear cost function. They analyzed the pseudo-convexity property of the bi-level program and developed a hypothesis test framework to examine the statistical properties of calibrated parameters. The proposed framework in this report differs from these previous studies in that it does not pre-select a behavioral model to represent the route choice preferences.

Another stream of studies uses deep neural networks, ranging from Long Short-Term Memory to Spatial-Temporal Graph Convolution Neural Network (see, e.g., Yao et al., 2019), to predict short-term traffic flows. These models can capture complex spatiotemporal correlations of traffic flows from multi-source data and show satisfactory accuracy. However, fundamentally, these models assume future flows will be generated by the same process that generated historical flows and then learn a direct mapping from input features to traffic flows. As such, the models would likely fail in an ”out-of-distribution” test where the underlying process changes. For example, in ”what-if” analysis, a planning agency may update the road network topology, thereby changing the process of generating traffic flows. More recently, some used supervised learning to learn a mapping from demands to equilibrium flows (e.g., Rahman and Hasan, 2022; Spana and Du, 2022). Such models suffer from the same limitation in ”out-of-distribution” tests. Moreover, they don’t use empirical data to learn the equilibrium state or travel choice preferences. By contrast, the proposed framework directly learns unknown model components from data and captures equilibrium conditions with a parametric VI. The learned equilibrium state will then serve as a consistent benchmark to help decision makers differentiate various plans. Although the preferences may evolve over time (but

can also be learned over time), it is reasonable to assume the same choice preferences when conducting ”what-if” planning analysis.

2.2 Computational-graph-based transportation network modeling

Computational graphs and automated differentiation provide powerful tools for numerically evaluating gradients and easily scale to very large datasets. In the field of transportation, computational graphs have been employed to model and calibrate individual components of network equilibrium models. Early explorations start with calibrating demand functions. For example, recent studies encoded trip generation, distribution, and path-based traffic loading within layered computational graphs in a static (Wu et al., 2018) and dynamic setting studies (Ma et al., 2020). This approach enables the estimation of hierarchical travel demands from various data sources.

Other advancements on the demand side integrate neural networks with discrete choice models to enhance the estimation of travel preferences. It has been demonstrated that carefully designed neural networks can provide interpretive, rather than ’black-box’, tools for choice analysis (Sifringer et al., 2020; Wang et al., 2020). One notable example is Sifringer et al. (2020), who decomposed the systematic part of the utility function into a knowledge-driven part from classical discrete choice models and a data-driven part from neural networks. By maintaining the independence of elasticities from two parts, their framework benefits from the predictive power of neural networks while keeping some key parameters interpretable. Other similar attempts include Wang et al. (2020), who encoded the irrelevant alternative constraints with alternative-specific connectivity. Their domain-knowledge-regularized neural network architecture better captures the substitution patterns of travel mode choices. Different types of neural networks, such as residual networks (Wong and Farooq, 2021), are synergized with discrete choice models to allow for similar interpretability as a Multinomial Logit model. These interpretable neural-network-based discrete choice models offer a good foundation for us to design the neural network architectures in the proposed end-to-end framework, particularly when behavior interpretability is desired.

To refine supply-side link performance functions, researchers recently developed a physics-informed neural network to approximate density and speed distribution and learned traffic states from multi-source data (Lu et al., 2023). To leverage domain knowledge and enhance estimation accuracy, they incorporate the violation of flow conservation constraints into the loss function as a regularization term. These studies, however, have overlooked the interaction between supply- and demand-side components, and consequently, fall short of generating an equilibrium state that can serve as a benchmark for ”what-if” analysis.

By contrast, the proposed framework models the interactions among travelers as a routing game and captures the equilibrium conditions with an implicit layer in end-to-end learning. The output of the implicit layer solves a fixed-point problem. It is called implicit because the output of the layer is defined implicitly–there is no analytical formula for it—and cannot be obtained via explicit computation rules, as the computational graph in standard or explicit neural networks (Travacca et al., 2020). The implicit layer was first proposed by Bai et al. (2019) and has been applied to various fields such as power flow prediction (Fioretto et al., 2020) and auction mechanism design (Feng et al., 2018). Recent studies have begun to develop computational-graph-based models that capture the interactions among travelers as a routing game and encapsulate equilibrium conditions. For instance, recent studies utilized a computational-graph-based framework to simultaneously learn supply- and demand-side components (Guarda et al., 2023). Instead of directly enforcing equilibrium conditions, they penalized the violation of these equilibrium conditions in the loss function, thereby guiding the calibration process to generate an equilibrium state.

Of the most relevant to our study are Li et al. (2020) and Heaton et al. (2021), who explored learning the equilibrium states of routing games with implicit layers. Specifically, Li et al. (2020) cast the equilibria as an implicit layer and calibrated cost parameters in an end-to-end fashion. Their study, however, still follows the traditional ”bottom-up” approach and pre-selects a behavior model before calibration. They only used computational graphs as a tool to enhance computational efficiency rather

than exploring the representation power of neural networks. Heaton et al. (2021) approximated the weather-dependent link performance functions with fully connected layers, which take the link flows and weather as input and output link travel time. They reformulated the weather-dependent equilibrium conditions as the fixed point of a decoupled projection operator and encapsulated the fixed-point problem in the implicit layer. They then trained the neural network with link flow observations. However, these methods share a common limitation: they still follow a bottom-up assembly approach and pre-select the functional form of each modeling component before encoding it as computational graphs.

Our work advances these previous studies by integrating the selection or learning of model components into an end-to-end framework, with neural networks automatically discovering a good specification of unknown model components from empirical data. It integrates model-based and model-free modeling approaches within a single pipeline. In addition, we propose a novel neural network architecture that ensures the existence of equilibria and accommodates changes in the road network topology to facilitate ”what-if” planning analysis.

2.3 Employing auto-differentiation for bi-level optimization and MPEC

Parameter calibration in network equilibrium models is typically formulated as a Mathematical Program with Equilibrium Constraints (MPEC). The objective function seeks to minimize the fitting error by adjusting parameters while adhering to Wardrop equilibrium constraints. Continuous model parameters often call for the use of gradient descent methods (Yang et al., 2001). However, these methods necessitate implicit differentiation, a typically challenging task that involves differentiating the equilibrium solution with respect to the parameters. The implicit differentiation usually requires equilibrium network sensitivity analysis, which either involves inverting a matrix, an operation that scales quadratically with the dimension of VI (Tobin and Friesz, 1988), or solving an additional linear VI (Patriksson, 2004). The former struggles with scalability issues on large road networks due to the difficulty of storing, not to mention inverting,

such a large matrix. Moreover, repeatedly solving the linear VI is also computationally expensive.

When the equilibrium constraint is equivalent to an optimization problem, the MPEC can be structured as a bi-level optimization problem. Recent advancements in computational graphs and auto-differentiation have facilitated the development of new bi-level optimization algorithms, which broadly fall into two categories. The first, known as Iterated Differentiation (ITD), approximates the implicit gradient by backpropagating along the trajectory of the lower-level optimization iterations. This method requires storing each iteration step of lower-level problems. As the computational graph expands proportionally to the number of iterations required to solve the lower-level problem, storing or unrolling a long optimization trajectory can be inefficient. The second method, known as Inexact Implicit Differentiation (IMD), sidesteps the need to store the lower-level optimization trajectory. It employs the implicit theorem to approximate the matrix inversion by iteratively solving an auxiliary fixed-point problem. Initially proposed for hyperparameter optimization and meta-learning (Maclaurin et al., 2015; Franceschi et al., 2018), both methods have been shown to converge to local optima under appropriate conditions when the lower-level is an unconstrained optimization problem (Ghadimi and Wang, 2018; Ji et al., 2021).

Several recent studies have employed auto-differentiation for solving MPECs. A recent study used mirror descent to handle path-based user equilibrium constraints (Li et al., 2022). Despite these advancements in handling equilibrium constraints, the convergence of MPEC remains a largely unexplored area. One recent study by (Li et al., 2023) demonstrated the asymptotic convergence of two modified ITD methods to a local optimum for MPEC when enforcing equilibrium conditions. However, questions remain regarding the performance of MPEC when the equilibrium constraint is replaced with a single step loading process. To the best of our knowledge, this study is the first to investigate these issues under both ITD and IMD methods.

3 Computational-graph-based VI Formulation of UE

We consider a case where partial aggregate traffic measures, such as link flow and link time, at peak periods are observable for a long period. Suppose that a planning agency is interested in developing a static network equilibrium model to analyze the network for peak periods. The general learning task is to learn the OD demands, travelers' route choice preferences, and link performance functions from multi-day observations. If prior knowledge is available, some components can be pre-calibrated, and the end-to-end framework only focuses on the remaining components.

3.1 Path-based formulation

Mathematically, consider a network $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ and $\mathcal{A}$ are the set of nodes and links. Let $\mathcal{R}$ denote the set of OD pairs. Each OD pair $r \in \mathcal{R}$ is connected by paths that form a finite and nonempty feasible path set $\mathcal{P}_r$. $\mathcal{P}$ represents the set of feasible paths for all OD pairs. Let and be the input features observed on day (sample) m. The input features include traveler characteristics like income, road network attributes like free-flow time, and contextual features like weather and gas price. Input features can vary from day to day (or sample to sample). Throughout the report, the norm denotes the L2 norm, unless otherwise indicated. Superscript m associates sample-dependent variables with the m-th sample.

We propose three continuous functions to approximate the unknown supply- or demand-side model components. The parameter of all components will be jointly learned and thus we say all components are parametrized by $\theta \in \Theta$. Each component can be model-based, model-free (e.g., neural networks), or hybrid (e.g., physics-informed neural networks). Therefore θ represents neural network parameters in a model-free approach, or parameters of a given functional form in a model-based approach.

We will elaborate on the construction of each component, starting from the supply side. The link performance function τ_θ outputs the link travel time $t^{[m]} \in \mathcal{T}$ as a function of path flow $h^{[m]} \in \mathcal{H}$ and input features, defined as:

$$\tau_\theta : \mathcal{H} \times \mathcal{X} \to \mathcal{T} \tag{1}$$

where the input features $x^{[m]} \in \mathcal{X}$ include contextual features and road network attributes, such as link capacity and free-flow time; the feasible region $\mathcal{H} \subseteq R_+^{|\mathbb{P}|}$ requires path flow to be nonnegative and is the feasible region of link time.

On the demand side, travelers are free to switch paths to improve their utilities. Findings from travel behavior research suggest that travel choice behaviors are much more complicated than just choosing the shortest path. We use the cost function π_θ to describe the perceived path cost given actual travel time. The cost function π_θ outputs the (perceived) path cost as a continuous function of link time and input features, defined as:

$$\pi_\theta : \mathcal{T} \times \mathcal{X} \to \mathcal{C} \tag{2}$$

where input features include traveler characteristics (e.g., income and travel purpose), route attributes (e.g., number of left turns), and contextual features. The feasible set $\mathcal{C} \subseteq R_+^{|\mathbb{P}|}$ requires path cost as nonnegative.

In addition to route choice, travelers have the freedom to choose travel or not and switch origin and/or destination to improve their utility. We assume the travel demand is upper bounded by a maximum possible demand $q \in R_+^{|\mathbb{R}|}$ and introduce the excess demand as $e^{[m]} = q - \Gamma^\top h^{[m]}$. Here, $\Gamma \in R^{|\mathbb{P}| \times |\mathbb{R}|}$ represents the path-OD incidence matrix and Γ_{pr} equals 1 if path p connects OD pair r and equals 0 otherwise. We use an inverse demand function λ_θ to depict the equilibrium path cost $u^{[m]} \in \mathcal{U}$ as a function of excess demand $e^{[m]} \in \mathcal{E}$ and input features, namely,

$$\lambda_\theta : \mathcal{E} \times \mathcal{X} \to \mathcal{U} \tag{3}$$

where the feasible region of excess demand is $\mathcal{E} = \{e \in R^{|\mathbb{R}|} : 0 \le e \le q\}$ and $\mathcal{U} \subseteq R_+^{|\mathbb{R}|}$ is the feasible region of equilibrium path cost.

Assuming rational travelers try to maximize their own travel utilities, the multi-class user equilibrium (UE) with elastic demand is formulated as the following parametric VI, the solution to which is the equilibrium path flow $h^{*[m]}$ and equilibrium excess demand $e^{*[m]}$ for sample m:

$$\begin{pmatrix} \pi_\theta(\tau_\theta(h^{*[m]}, x^{[m]}), x^{[m]}) \\ \lambda_\theta(e^{*[m]}, x^{[m]}) \end{pmatrix}^T \begin{pmatrix} h - h^{*[m]} \\ e - e^{*[m]} \end{pmatrix} \geq 0, \forall h \in H, e \in E \#(4)$$

To simplify notation, we introduce the response variable as $y = (h, e)$ and the generalized cost as $z = (c, u)$. By defining the generalized cost function:

$$F_\theta: \mathcal{Y} \times \mathcal{X} \to \mathcal{Z} \tag{5}$$

where $F_\theta(y, x) = [\pi_\theta^\top(\tau_\theta(h, x), x), \lambda_\theta^\top(e, x)]^\top$.

Figure 2 illustrates the computational-graph-based generalized cost function for path-based elastic UE. Supply- and demand-side components are shown in blue and green respectively. The dependence of variables on sample m is omitted to simplify the notation. Each parametrized component can be model-based, model-free, or hybrid. Then the parametric VI in Eq. (5) can be compactly reformulated as:

$$\langle F_\theta(y^{*[m]}, x^{[m]}), y - y^{*[m]} \rangle \geq 0, \quad \forall y \in Y \#(6)$$

To compactly represent the feasible region of the response variable, we introduce the augmented path-OD incidence matrix as $\Gamma = [\Gamma, I]^\top \in R^{(|\mathbb{P}|+|\mathbb{R}|)\times|\mathbb{R}|}$ and the feasible region of the response variable becomes $\mathcal{Y} = \{y \in R^{|\mathbb{P}|+|\mathbb{R}|}: y \geq 0, \Gamma^\top y = q\}$

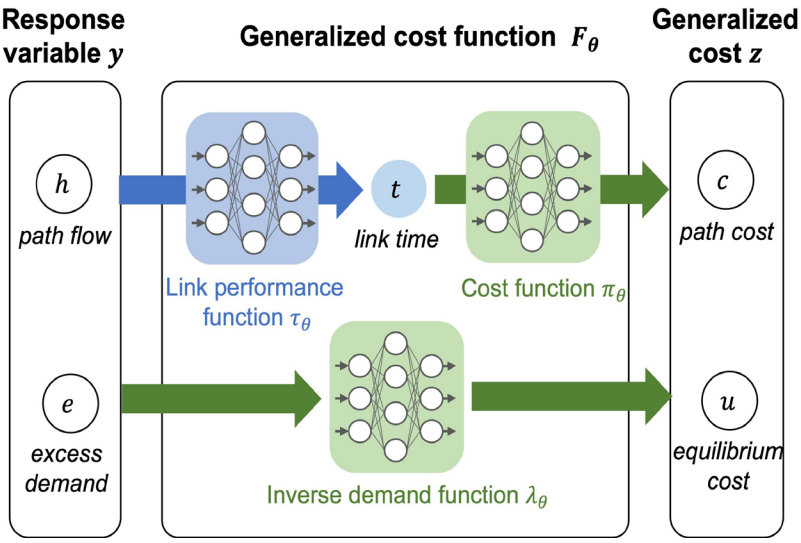


Figure 2 Illustration of the computational-graph-based generalized cost function for path-based elastic UE.

Theorem 1 (Existence of equilibrium) There exists at least one solution to the multi-*class user equilibrium problem in Eq. (6).*

Proof 1 *Response variable* $y^{[m]}$ *is a solution to* $VI(F_\theta(y,x^{[m]}),\mathcal{Y})$ *if and only if it is the fixed point of the projection operator* $P_{\mathbb{Y}}(\cdot)$ *for any* $\alpha > 0$*, defined as:*

$$y^{*[m]} \in VI(F_\theta(y,x),\mathcal{Y}) \Leftrightarrow y^{*[m]} = P_{\mathbb{Y}}(y^{*[m]} - \alpha F_\theta(y^{*[m]},x)) \#(7)$$

where the projection operator is defined as $P_{\mathbb{Y}}(y) = argmin_{y' \in \mathcal{Y}} |y' - y|$

The generalized cost function is approximated by continuous parametric functions or neural networks and thus is continuous. The fixed-point operator is a projection operator that is continuous. Because the feasible flow set is convex and compact*, as per Brouwer's fixed point theorem, there exists at least one solution to the fixed-point problem in Eq. (6).*

If we consider a special case where the OD demands are observable for each sample, the proposed framework can handle an inelastic demand setting. Let be the OD demands and link flows observed on sample m, where q. Then the multi-class UE with inelastic demand for sample m is formulated as a parameterized VI in Eq. (6), the solution to which is the equilibrium flow:

$$\left\langle \pi_{\theta}\left(\tau_{\theta}\left(h^{*[m]}, x^{[m]}\right), x^{[m]}\right), h^{[m]} - h^{*[m]} \right\rangle \geq 0, \quad \forall h^{[m]} \in \mathcal{H}^{[m]} \#(8)$$

In this case, the feasible path flow set becomes sample-dependent, i.e., $\mathcal{H}^{[m]} = \{h \in R^{|\mathbb{P}|}: h \geq 0, \Sigma^{\top} h = \overline{q}^{[m]}\}$ requires the feasible path flows to be nonnegative and satisfy flow conservation.

3.2 Link-based formulation

The parametric VI defined in Eq. (6) requires the knowledge of feasible path set. This is a common assumption for path-based UE formulation and methods for generating the feasible path set are well-developed in the literature (Frejinger et al., 2009). If the modelers believe the path cost is link-additive, the link-based elastic-UE formulation can be used instead.

We introduce OD-specific link flows for OD pair r as and the vectorized OD-specific link flows as $v = \{v_r\}_{r \in \mathcal{R}} \in \mathcal{V} \subseteq R_{+}^{|\mathbb{A}| \times |\mathbb{R}|}$. In this case, the link performance function becomes:

$$\tau_{\theta}: \mathcal{V} \times \mathcal{X} \to \mathcal{T}. \tag{9}$$

We slightly abuse the notation of path cost and define the OD-specific link cost $c_r \subseteq R_{+}^{|\mathbb{A}|}$ with its vectorized form as $c = \{c_r\}_{r \in \mathcal{R}}$. The link-based equilibrium condition for sample m is formulated as the following parametric VI:

$$\begin{pmatrix} \pi_{\theta}\left(\tau_{\theta}\left(v^{*[m]}, x^{[m]}\right), x^{[m]}\right) \\ \lambda_{\theta}\left(e^{*[m]}, x^{[m]}\right) \end{pmatrix}^{T} \begin{pmatrix} v - v^{*[m]} \\ e - e^{*[m]} \end{pmatrix} \geq 0, \forall v \in V, e \in E \#(10)$$

slightly adjust the notation for generalized cost and response variable to bring the link-based and path-based formulations under the same umbrella. For each OD pair r, we define the response variable as $y_r = (v_r, e_r)$ with its vectorized form given as $y = \{y_r\}_{r\in\mathcal{R}} \in \mathcal{Y}$. The generalized cost for OD pair r is represented as$z_r = (c_r, u_r)$, and its vectorized form is formulated as $z = \{z_r\}_{r\in\mathcal{R}} \in \mathcal{Z} \subseteq R_+^{(|\mathbb{A}|+\mathbb{1})\times|\mathbb{R}|}$.

To compactly formulate the feasible region for response variable y, we introduce the augmented link-node incidence matrix and vectorized demand constraint as follows. For the former, we add a number of $|\mathcal{R}|$ dummy links connecting the origin and destination of each OD pair, with a number of e_r travelers on each dummy link experiencing the equilibrium path cost u_r. Then we represent the augmented link-node incidence matrix including dummy link as $\Lambda \in R^{(|\mathbb{A}|+\mathbb{1})\times|\mathbb{N}|}$ where $\Lambda_{an} = 1$ if link a originates from a and $\Lambda_{an} = -1$ if link a terminates at node n. For each OD pair, we define a vectorized demand constraint, where $d_r \in R_+^{|\mathbb{A}|}$; $d_{rn} = q$ if OD pair r originates at node n and $d_{rn} = -q$ if OD pair r terminates at node n and $d_{rn} = 0$ otherwise. Then the feasible region of the response variable can be compactly formulated as $\mathcal{Y} = \{y \in R^{(|\mathbb{A}|+\mathbb{1})\times|\mathbb{R}|}: y \geq 0, \Lambda^\top y_r = d_r, \forall r \in \mathcal{R}\}$. It is straightforward to validate that both the path-based equilibrium condition in and the link-based equilibrium condition align with the same compact parametric VI in Eq. (6).

4 Framework Formulation

We consider a smooth loss function $l: \mathcal{Y} \times \mathcal{Y} \to R$ that measures the distance between the estimated equilibrium states and corresponding observations. We also consider a regularization function $r(\theta)$. The training of the end-to-end framework can be formulated as the following MPEC. Each training sample m corresponds to the pair $(x^{[m]}, \bar{y}^{[m]})$. Consider the dataset $\mathcal{S} = \{(x^{[m]}, \bar{y}^{[m]})\}_{m=1}^{|\mathcal{M}|}$, where each data point is drawn i.i.d. from an unknown probability distribution P over $\mathcal{X} \times \mathcal{Y}$.

$$\min_{\theta} E_x\left[\mathcal{L}\left(y^{\star[m]}, \bar{y}^{[m]}\right)\right]$$
$$s.t. \quad y^{\star[m]} \in VI\left(F_\theta(y, x^{[m]}), \mathcal{X}\right), \quad \forall m \in M \#(11)$$

The end-to-end framework unifies the parameters of supply- and demand-side components, either model-based or model-free, into a generalized cost function and jointly learns θ during training.

Remark 1 *If the cost function is independent of input feature x and equals the sum of link travel times and an entropy term, the learning problem will reduce to the logit dispersion parameter calibration problem investigated by Yang et al. (2001). If the equilibrium constraints are removed, the learning problem would directly learn a mapping from the context features x to link flows v. In this case, the problem reduces to neural-network-based short-term traffic flow prediction investigated in the literature (e.g., Yao et al. (2019)).*

The loss function is flexible to accommodate modelers' needs and available data sources. It can include partial aggregate traffic state observations like link flow and travel time, path choice probabilities from trajectory data, and benchmark OD demands from planning agencies. The framework integrates multi-source data into a single loss function and effectively handles inconsistencies among different data sources.

Remark 2 *In network equilibrium models, 'link flow' refers to link demand or link inflow, representing the number of travelers choosing to use a specific link. However, loop detectors record only link outflows, which can vary from link inflows when the network is congested. Therefore, when data from loop detectors is available, it's often more appropriate to use link travel time as the empirical observation, instead of link outflows.*

4.1 Neural network architecture

This section discusses the design of the neural network architecture in the proposed end-to-end learning framework. The architecture needs to accommodate the changes in the

road network topology to facilitate ”what-if” analysis. Moreover, it can be designed to ensure that the cost function possesses the desired properties to enable efficient training. We will illustrate it with the cost function with inelastic demand as an example. Hereinafter, we highlight that features/attributes are the concactation of single features/attributes for all elements within one set. For example, the link feature is. The design of the cost function requires special consideration. We distinguish ”feature” from ”attribute” to avoid ambiguity: features refer to the input data of neural networks whereas attributes refer to the learned outputs of neural networks.

4.1.1 Attribute net

We propose Attribute Net to learn the (path) attributes considered by travelers in their route choice decision process. As shown in Figure 5, attributes $s^{[m]}$ depend on path flows $h^{[m]}$ and road network features $x^{\mathcal{G}}$. Attribute Net G_{θ} learns a continuous mapping from path flows and road network features to attributes, defined as:

$$G_{\theta}: \mathcal{H}^{[m]} \times \mathcal{X}^{\mathcal{G}} \rightarrow \mathcal{S},$$

One may construct the Attribute Net with fully connected layers and learn a global mapping from link flows to link costs (e.g. Heaton et al. (2021)). In this case, the input and output dimensions of fully connected layers depend on the number of links in the road network. However, in ”what-if” analysis, a planning agency may change the road network topology by adding or removing links. The fully connected layers — by definition with fixed size input and output — are incapable of accommodating the change in the number of links.

Inspired by the ”kernel” concept in Convolution Neural Networks, we propose to learn the local attributes on the link, node, and path levels with three parallel, fully connected layers. As shown in Figure 3, the feature/attribute subscripts for enumerating the elements within a set and the superscripts for a sample m are omitted to facilitate presentation. The fully connected layers that learn link, node, and path attributes are called link, node, and path block respectively. The parameters of each block are shared among all elements of

the same level to capture repeated patterns. Each block's input and output dimensions are independent of road network topology, allowing for changeable input sizes. To facilitate the presentation, the superscripts for a sample m are omitted for the rest of this section. We use the superscript $\mathcal{A}, \mathcal{N}$, and $\mathcal{P}$ to distinguish the notations related to link, node, and path block.

The detailed constructions of link, node, and path block are similar. Hence, we take the link block as an example. As opposed to accepting multiple links as input, the link block takes the single link flow and single link features of one link $a \in A$ as input and outputs the corresponding link attributes, defined as:

$$g_{\theta}^{\mathcal{A}}: R_{+} \times \mathcal{X}_{a}^{\mathcal{A}} \rightarrow \mathcal{S}_{a}^{\mathcal{A}},$$

where $\mathcal{X}_{a}^{\mathcal{A}} \subset R^{|\mathcal{I}^{\mathcal{A}}|}$ is the feasible set of single link features; $|\mathcal{I}^{\mathcal{A}}|$is the number of features

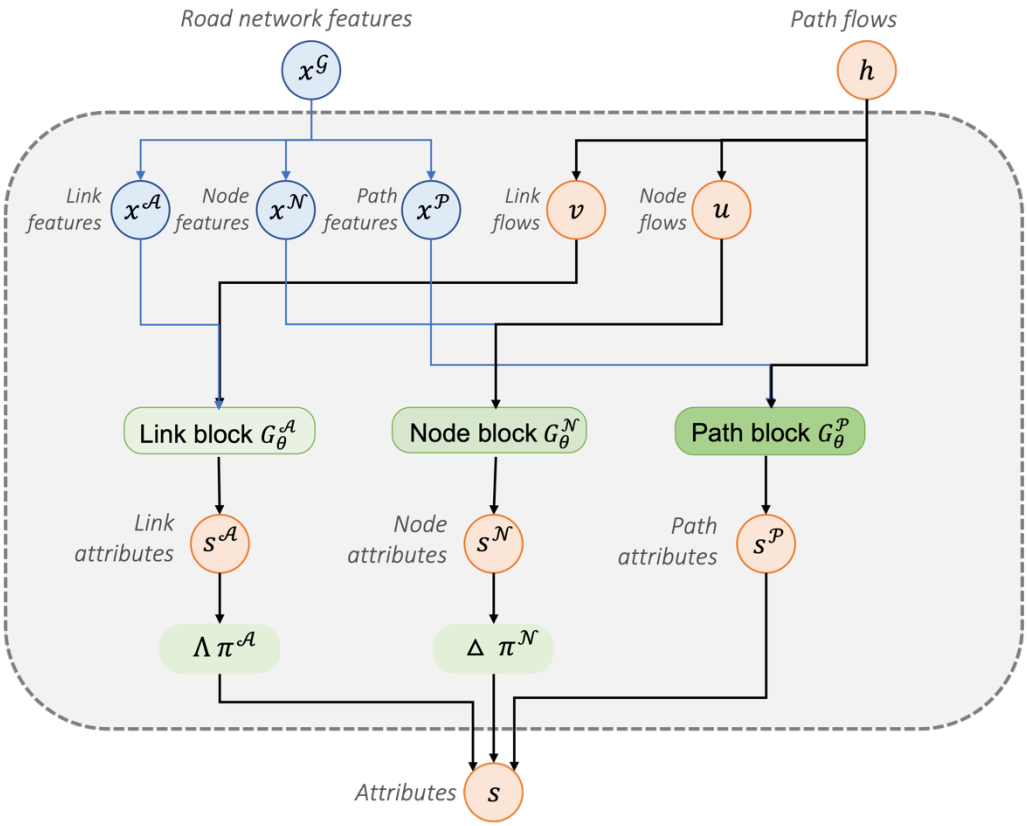


Figure 3 Illustration of Attribute Net.

associated with one link; $\mathcal{S}_a^{\mathcal{A}} \subset R^{|\mathbb{S}^{\mathbb{A}}|}$ is the feasible set of single link attributes and $|\mathcal{S}^{\mathcal{A}}|$ is the number of link attributes considered by travelers. Note the input and output dimensions of the link block are independent of link numbers. Example 1 further illustrates how the proposed neural network architecture deals with changeable size inputs. The link attributes $s^{\mathcal{A}} \in R^{|\mathbb{A}| \times |\mathbb{S}^{\mathbb{A}}|}$ are the concatenation of single link attributes, defined as:

$$s^{\mathcal{A}} = \left\{ g_\theta^{\mathcal{A}}\left(v_a, x_a^{\mathcal{A}}\right)^\top \right\}_{a \in \mathcal{A}}.$$

Similarly, let node flow be the sum of link flows from all approaches at node. To capture the interactions among link flows, node block $g_\theta^{\mathcal{N}}: R_+ \times \mathcal{X}_n^{\mathcal{N}} \to s_n^{\mathcal{N}}$ maps the single node flow $\overline{u}_n \in R_+$ and single node features of one node to its local node attributes. The node attributes are the concatenation of single node attributes, defined as:

$$s^{\mathcal{N}} = \left\{ g_\theta^{\mathcal{N}}\left(\overline{u}_n, x_n^{\mathcal{N}}\right)^\top \right\}_{n \in \mathcal{N}}.$$

And the path block $g_\theta^{\mathcal{P}}: R_+ \times \mathcal{X}_p^{\mathcal{P}} \to s_p^{\mathcal{P}}$ maps the single path flows and single path features one path to its path attributes. The path attributes are:

$$s^{\mathcal{P}} = \left\{ g_\theta^{\mathcal{P}}\left(h_p, x_p^{\mathcal{P}}\right)^\top \right\}_{p \in \mathcal{P}}.$$

Finally, the attributes π are the concatenation of link attributes, node attributes and path attributes, defined as:

$$s = \{\Lambda s^{\mathcal{A}}, \Sigma s^{\mathcal{N}}, s^{\mathcal{P}}\},$$

where is Σ the path-node incidence matrix.

Example 1 (Accommodate changeable input sizes) *Consider a road network with a single OD pair connected by two parallel paths or links (i.e., link 1 and link 2). Slash*

boxes in Figure 4 denote the change in variables when another parallel link is added to the road network. The link block takes the link flow, capacity, and free-flow time of link 1 as input and outputs the link travel time on link 1 (highlighted with red boxes). The input dimension is 3 and the output dimension are independent of the number of links in the road network. When a new link is added to the original road network, one dimension is added to path flows h (denoted as the slash box in Figure 4) whereas the input and output dimensions of the link block remain the same.

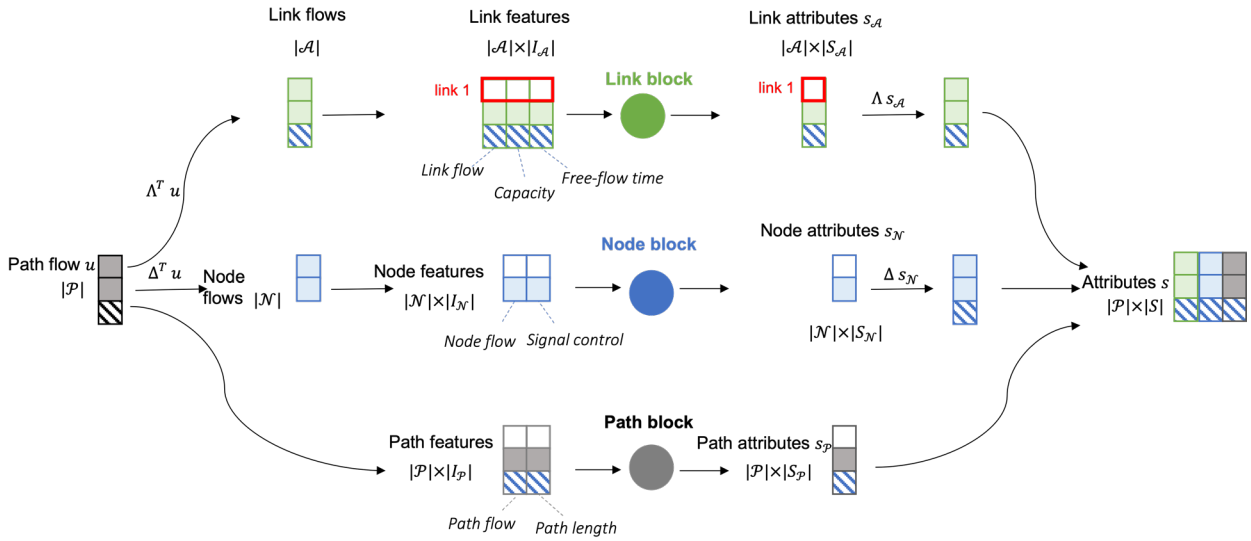


Figure 4 Illustration of link, node, and path blocks.

To facilitate training and enhance model performance, we can fully or partially replace each block with a pre-calibrated function, if available. For instance, we can replace the link block with the link performance functions calibrated by a planning agency. In addition, our future study will explore the use of convolution layers to accommodate changeable input sizes. The challenge will be to ensure the desired properties of the learned cost function.

4.1.2 Weight net

Weight Net is proposed to capture traveler heterogeneities and learn the OD-specific preferences over learned attributes (see Figure 5). We treat all travelers between the same OD pair as a single class that shares the same preferences. It is straightforward to further classify travelers between one OD pair to be multiple classes to reflect the preference heterogeneity among them. Weight Net L_θ learns a mapping from traveler characteristics to OD-specific weights $w \in W$, defined as:

$$L_\theta: \mathcal{X}^{\mathcal{R}} \rightarrow \mathcal{W},$$

OD pairs can be added or removed in "what-if" analysis thus Weight Net also needs to accommodate the change in the number of OD pairs. Weight Net learns a function that maps the single traveler characteristics of one OD pair $x_r^{\mathcal{R}}$ to its OD-specific weights w_r, defined as:

$$L_\theta\left(x_r^{\mathcal{R}}\right): \mathcal{X}_r^{\mathcal{R}} \rightarrow \mathcal{W}_r,$$

The parameters of neural network are shared among all OD-pairs to capture the repeated patterns in weights. Recent developments in interpretable neural-network-based discrete choice modeling, as discussed in Section 2, can be incorporated into the proposed framework and guide the design of neural network architectures, particularly when behavior interpretability is desired.

4.1.3 Cost Function and Regularization

Subsequently, we assume that travelers choose routes to minimize their perceived path costs, which are represented as a weighted sum of attributes:

$$\pi^{[m]} = \Sigma\, w \odot s^{[m]} 1,$$

where $\odot$ represents the Hadamard (elementwise) product and is a $\mathbb{1} \in R^{|\mathbb{S}|}$ column vector of ones to calculate the sum over the rows. Equivalently, let context features $\mathbb{1} \in R^{|\mathbb{S}|}$

include traveler characteristics $x^{\mathcal{R}}$ and road network features$x^{\mathcal{G}}$. The cost function maps path flows and context features to path costs, defined as:

$$\pi_\theta(h^{[m]},x) = \Sigma\, L_\theta(x^{\mathcal{R}}) \odot G_\theta(h^{[m]},x^{\mathcal{G}})\, 1,$$

As shown in Theorem 1, the continuity of cost function ensures the existence of equilibria. However, stronger properties of the cost function may be desired to ensure the uniqueness of equilibrium or enable an efficient solution algorithm. In this section, we seek to entail the cost function with monotonicity and Lipschitz continuity via neural network regularization techniques. Both monotonicity, which suggests the path cost is non-decreasing as more travelers use this path, and Lipschitz continuity, which suggests a finite change in path flows results in a finite change in path costs, are mild assumptions but will largely enhance computational traceability.

Theorem 2 shows sufficient conditions to entail the cost function with monotonicity and Lipschitz continuity. The proof is shown in Appendix 10. Note that only path flows are treated as variables in this case.

Theorem 2 (Monotonicity and Lipschitz continuity of cost function) The cost function πθ (h, x) defined in Eq.(9) is maximal monotone and Lipschitz continuous with respect to path flows h if weight w is positive and link block, node block and path block are column-wise monotone.

Recall that each block is composed of fully connected layers. Let $y^{(l-1)}$and $\sigma^{(l)}$ represent the input and activation function of the l-th layer respectively. The output of the l-th layer is calculated as $y^{(l)} = \sigma^{(l)}\left(W^{(l)}y^{(l-1)} + b^{(l)}\right)$where $W^{(l)}$ and $b^{(l)}$ are learnable parameters of linear layers. We constrain the sign of weights as strict positive by using SoftPlus as the last layer of Weight Net. The column-wise monotonicity and Lipschitz continuity of attribute blocks, however, are more challenging to obtain. Most activation

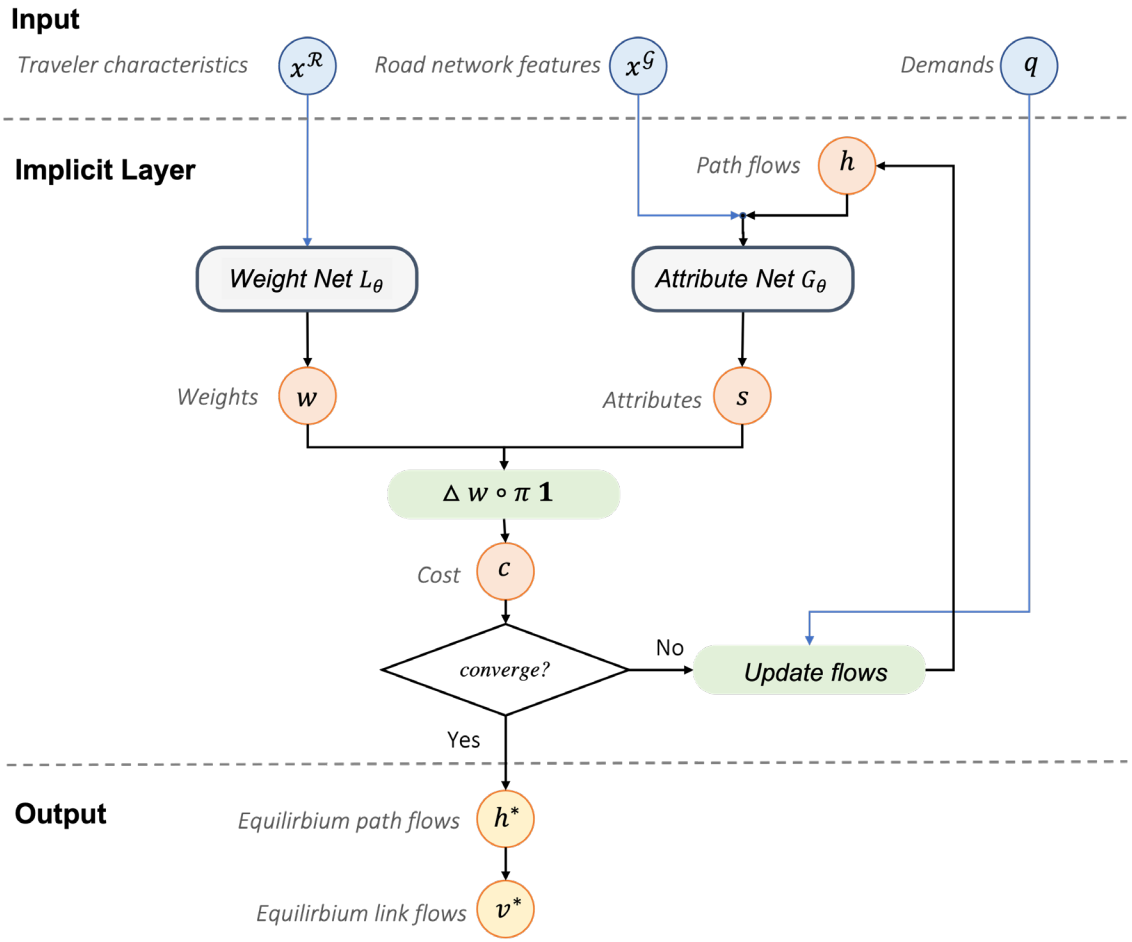


Figure 5 Illustration of the end-to-end learning framework.

layers, such as ReLU and SoftPlus, are monotone and Lipschitz (Bibi et al., 2019) and both monotonicity and Lipschitz continuity are preserved via operator composition. Therefore, we only need to regularize the linear layer to entail the block with desired properties. Without loss of generality, we design a monotonic and Lipschitz continuous architecture that explicitly constrains the weights of the linear layers. More specifically, the weight of each linear layer is constrained to be positive to maintain monotonicity. The linear layer can be parameterized as $W^{(l)} = B^\top B + \iota\, I$ with $\iota > 0$ if strict monotonicity or strong monotonicity are desired. The spectral normalization as proposed by Miyato et al. (2018) is applied to constrain the spectral norm of each $W^{(l)}$ and maintain Lipschitz continuity. This explicit method is reliable, easy to implement, and shows satisfactory performances in our numerical experiments. Other regularization methods, such as adding

heuristic penalty terms to the loss function or solving integral problems in forward propagation (Wehenkel and Louppe, 2019; Gouk et al., 2021) are open for exploration in our future study.

5 Framework Training

We need to deal with two computational challenges to implementing implicit layers in the proposed framework. First, it requires efficiently solving a batch of VI problems in the forward propagation, as previous methods for solving VI may not necessarily be suitable for batch operations. Second, because solving VIs usually entails many iterations, explicit backpropagation through each iteration can be computationally expensive. Efficient differentiation through the implicit layer, i.e., the VI, is needed.

This section presents an auto-differentiation-based gradient descent algorithm to solve the MPEC in Eq. (11). For simplicity, we explicitly formulate the dependence of the parameters while omitting input features. The optimality condition of the parametric VI in Eq. (6) can be recast as a fixed point problem as:

$$y^*(\theta) = g(\theta, y^*(\theta))$$

where $g(\theta, y)$ is the fixed point operator. We define the total loss function $f(\theta, y) = \ell(y(\theta), y) + r(\theta)$ .Using this fixed point operator, the MPEC in Eq. (11) can be reformulated as:

$$\min_{\theta} \Phi(\theta) = \frac{1}{|\mathcal{M}|} \sum_{m \in \mathcal{M}} f(\theta, y^{*[m]}(\theta))$$

$$s.t. \quad y^{*[m]}(\theta) = g(\theta, y^{*[m]}(\theta)), \quad \forall m \in \mathcal{M}$$

We consider the generalized cost function is strongly monotone and Lipschitz continuous so that the equilibrium state is unique and is a continuous function of parameter θ (Dafermos, 1988). In a model-free modeling approach, neural networks can be

regularized to ensure these desired properties. In this case, the proposed algorithm updates the parameter with its hypergradient in each training epoch, defined as:

Definition 1 (Hypergradient) *The hypergradient denotes the gradient of the loss function with respect to the parameter, defined as:*

$$\nabla \Phi(\theta) = \nabla f(\theta, y^*(\theta)) = \nabla_\theta f(\theta, y^*(\theta)) + \nabla y^*(\theta) \nabla_y f(\theta, y^*(\theta)).$$

The hypergradient requires differentiating through the equilibrium state $y*(\theta)$ to calculate the implicit gradient. To formally define the implicit gradient, we assume the following assumption holds.

Assumption 1 *The fixed-point operator $g(\theta, y)$ is continuously differentiable with respect to θ and y and matrix $\boldsymbol{I} - \nabla_y g(\theta, y^*(\theta))$is invertible.*

Supposing Assumption 1 holds, one can differentiates through the optimality condition and calculate the implicit gradient as:

Definition 2 (Implicit gradient) *Supposing Assumption 1 hold, the implicit gradient is defined as:* $\nabla y^*(\theta) = \nabla_\theta g(\theta, y^*(\theta))(\boldsymbol{I} - \nabla_y g(\theta, y^*(\theta)))^{-1}$.

Here we proceed to discuss the differentiability assumption in Assumption 1. If we assume the travelers follow the logit model when choosing their paths, the fixed-point operator is a logit loading function and is indeed differentiable. In a more general setting, the solution to VI can always be formulated as the fixed point of a gradient-projection operator, defined as:

Definition 3 (gradient-projection operator) *The gradient-projection operator is defined as follows for step size $\alpha > 0$.*

$\overline{g}(\theta, y) = P_Y(z - \alpha F(\theta, y))$.

where $\boldsymbol{P_Y}$ is the projection operator onto the feasible region.

The gradient-projection operator is non-differentiable at the boundary of the feasibility set. In this case, Assumption 1 implies that we are focusing on the differential region of the gradient-projection operator, thereby keeping it within the differential programming region for convergence analysis. This approach is also adopted by Li et al. (2022). How to tackle the non-differentiability at the boundary remains to be an open question.

We consider training the end-to-end framework with K epochs. Each epoch handles two sub-problems: forward propagation, which finds an approximate optimal response variable via N iterations, and backpropagation, which employs auto-differentiation to approximate the hypergradient and update parameters. We will then elaborate on each subproblem. Subscript script k associates a variable with the k-th epoch and superscript n and q associates a variable to the n-th forward and q-th backward iteration respectively.

5.1 Forward: N-step closed-form updates

Batched operation is essential for efficiently handling large empirical data sets when training the end-to-end framework. Specifically, forward propagation requires solving a batch of VIs in parallel, rather than solving a single constrained VI. Previous methods for solving VIs require repeatedly calling external optimization libraries to project onto the polyhedron constraint set of feasible path flows, and thus may not necessarily be suitable for batch operations (Li et al., 2020). To manage batch operations, we require a closed-form method for updating response variables so that we can encode this iterative process with computational graphs. These closed-form update rules also facilitate efficient auto-differentiation through equilibrium states during backpropagation. We will discuss two types of solution algorithms, decoupled gradient-projection and mirror descent method, to handle link-based and path-based formulation respectively. We omit the dependence on sample m in the following discussion.

5.1.1 Decoupled gradient-projection

We apply the decoupled gradient-projection method to deal with link-based equilibrium constraints. The forward propagation updates the response variable via N-step gradient-projection operations. The n-th forward iteration follows:

$$y^{n+1} = \overline{g}(\theta_k, y^n) = P_Y(y^n - \alpha F(\theta_k, y^n))$$

In a link-based formulation, the feasibility set is the Minkowski sum of the feasibility set for each OD pair, namely, $\mathcal{Y} = \sum_{r \in \mathcal{R}} \mathcal{Y}_r$ where $\mathcal{Y}_r = \{y_r : y_r \geq 0, \Lambda^\top y_r = d_r\}$. This allows us to break down the constraints by OD pairs and sequentially handle every pair on a large road network. Projecting directly onto the polyhedron constraint set, which requires repeatedly solving a batch of quadratic optimization problems. To tackle this efficiently, we leverage recent advancements in operator-splitting methods and decompose the polyhedron constraint set $\mathcal{Y}_r$ into two simpler sets: (i) one only involves inequality constraint $\mathcal{Y}_r^1 = \{y_r : y_r \geq 0\}$and (ii) another only involves equality $\mathcal{Y}_r^2 = \{y_r : \Lambda^\top y_r = d_r\}$. The projection onto two simplier set $\mathcal{Y}_r^1$ and $\mathcal{Y}_r^2$ have closed-form solutions that can be encoded within computational graphs and then efficiently implemented in a batched manner. The convergence of this decoupled gradient-projection method has been demonstrated by Heaton et al. (2021).

Starting with an initial point, the decoupled gradient projection repeats $y^{n+1} = \overline{g}(\theta_k, y^n)$ for each step until the iteration step n exceeds the maximum number of iterations N. The initial point is not necessarily feasible and will be projected onto the feasible region during the iteration. The selection of step size is vital. If the step size is too large, the iteration may diverge; if too small, the convergence can be extremely slow. The optimal step size depends on an unknown Lipschitz constant, the exact computation of which is NP-hard (Virmaux and Scaman, 2018). We thus explore two variants of decoupled gradient-projection iteration to adjust the step sizes and speed up the convergence: Anderson mixing (Walker and Ni, 2011) and weighted ergodic iteration (Davis and Yin, 2017).

Anderson mixing updates y^{n+1} as an optimal linear combination of τ previous iterations. The optimal step size solves a quadratic program:

$$\min_{\sum_{i=1}^{\tau}\alpha^i=1} \sum_{i=1}^{\tau} \left(\phi^{n-i+1}\alpha^i\right)^2,$$

where the objective function is to minimize the sum of optimality gap over τ iterations. Here ϕ^{n-i+1} represents the optimality gap defined as follows.

Definition 4 (Optimality gap) *The optimality gap measures the absolute change of the response variable between two consecutive iterations, defined as* $\phi^n = ||y^{n+1} - y^n||$.

Another variant, weighted ergodic iteration, heuristically adjusts the step size at each iteration and updates response variable as a linear combination of previous steps:

$$y^n = \frac{2}{(n+1)(n+2)} \sum_{i=0}^{n} (i+1) y^i.$$

5.1.2 Mirror descent

Mirror descent method has shown good performance in dealing with path-based equilibrium constraints. We define the path choice probability $\sigma^n = \frac{y^n}{\bar{\Gamma}^\top \bar{\Gamma} y^n}$ as auxiliary variables. For the n −th forward iteration, the path choice probability is calculated as follows:

$$\sigma^{n+1} = \frac{\sigma^n \odot \exp(-\alpha F(\theta, y^n))}{\bar{\Gamma}^\top \bar{\Gamma}(\sigma^n \odot \exp(-\alpha F(\theta, y^n)))}$$

The response variable is updated with a closed-form mirror descent operator:

$$y^{n+1} = \tilde{g}(\theta, y^n) = \bar{\Gamma}^\top q \odot \sigma^{n+1},$$

where $\alpha > 0$ is the step size. The constraint set of response variables becomes a probability simplex in path-based formulation and mirror descent has been shown efficient to deal with such a constraint set. This update rule can be viewed as a variant of the logit loading where the observed cost is scaled by the logarithm of route choice probability. The mirror descent iteration converges to the solution to the parametric VI in Eq.(2), as demonstrated in Li et al. (2022).

5.1.3 Root-finding

Solving for the auxiliary fixed point is equivalent to finding the root of $y^* - g_\theta(y^*, x) = 0$ via a root-finding method. The projection operator is non-differentiable at the boundary of a set and thus Newton's method may diverge. Therefore, we use Broyden's method, a quasi-Netown method that does not require differentiability. Broyden's method approximates Newton's direction and updates the point as $y^{n+1} = y^n - s^n$. Let the initial guess be s0 = −I and the direction is updated as:

$$s^{n+1} = s^n + \frac{\Delta y^{n+1} - s^n \Delta \phi^{n+1}}{\Delta y^{(n+1)\top} s^n \Delta \phi^{n+1}} \Delta y^{(n+1)\top} s^n,$$

where $\Delta y^{n+1} = y^{n+1} - y^n$ and $\Delta \phi^{n+1} = \phi^{n+1} - \phi^n$.

5.2 Backward: approximate hypergradient

In forward propagation, we consider a practical setting where the parametric VI is solved with N steps and terminated before reaching perfect equilibrium. Consequently, in backpropagation, we need to approximate the hypergradient at a non-optimal response variable.

Definition 5 (Approximate hypergradient) *The approximate hypergradient at* $\bar{y}$ *is defined as* $\widehat{\nabla} \Phi(\theta) = \nabla_\theta f(\theta, \bar{y}) + \widehat{\nabla} \bar{y}(\theta) \nabla_y f(\theta, \bar{y})$,

where the approximate implicit gradient $\widehat{\nabla} \bar{y}(\theta)$ is defined as follows.

Definition 6 (Approximate implicit gradient) *The approximate implicit gradient at* $\bar{y}$ *is defined as:* $\widehat{\nabla}\bar{y}(\theta) = \nabla_\theta g(\theta, \bar{y})(I - \nabla_y g(\theta, \bar{y}))^{-1}$

To avoid the computationally expensive matrix inversion in approximating the implicit gradient, we present two auto-differentiation-based methods to approximate the implicit gradient.

5.2.1 Iterated Differentiation (ITD)

ITD memorizes the trajectory of N-step forward iterations and directly backpropagates through the equilibrating trajectory. In the N-th forward iteration, the response variable y_k^N depends on θ_k and y_k^{N-1}, namely:

$$y_k^N = g(\theta_k, y_k^{N-1})$$

Here we use the fixed point operator g as the "unified" operator that includes both gradient-projection operator and mirror descent operator. By applying the chain rule, we obtain the following approximation for the implicit gradient under ITD:

$$\widehat{\nabla} y^N(\theta) = \nabla_\theta g(\theta, y^{N-1}) + \nabla_y g(\theta, y^{N-1})\widehat{\nabla} y^{N-1}(\theta)$$

By telescoping the definition of1 and using the fact that ITD approximates the implicit gradient.

5.2.2 Inexact Implicit Differentiation (IMD)

IMD approximates the Hessian-inverse-vector product by solving an auxiliary fixed-point problem. By defining the auxiliary variable as

$$v_k^* = (I - \nabla_y g(\theta_k, y_k^N))^{-1} \nabla_y f(\theta_k, y_k^N)$$

the approximate hypergradient can be formulated as:

$$\widehat{\nabla}\Phi(\theta_k) = \nabla_\theta f(\theta_k, y_k^N) + \nabla_\theta g(\theta_k, y_k^N)\, v_k^*$$

Reformulating the definition of auxiliary variable suggests that it solves an auxiliary fixed-point problem:

$$v_k^* = \nabla_y g(\theta, y_k^N)\, v_k^* + \nabla_y f(\theta, y_k^N)$$

Then IMD recursively approximates the auxiliary variable using Q-step fixed point iteration:

$$v_k^{q+1} = v_k^q + \gamma\left(\nabla_y f(\theta_k, y_k^N) - \left[I - \nabla_y g(\theta_k, y_k^N)\right] v_k^q\right),$$

where $\gamma > 0$ is the step size. The approximate hypergradient under IMD is:

$$\widehat{\nabla}\Phi(\theta_k) = \nabla_\theta f(\theta_k, y_k^N(\theta_k)) + \nabla_\theta g(\theta_k, y_k^N(\theta_k))\, v_k^Q.$$

The auxiliary fixed-point iteration converges if $I - \nabla_y g(\theta_k, y_k^N)$ is a stable matrix with a maximum eigenvalue that has a magnitude less than one. Previous studies show that these iterations typically are convergent in practice (Bai et al., 2019).

There are other methods in the literature to reduce the computational difficulty by approximating the matrix inversion. First, the Jacobian-free backpropagation replaces the matrix inverse with one identity matrix. This method can be viewed as a preconditioned gradient and only requires backpropagating through the final forward step (Fung et al., 2021). Second, an inverse matrix can be approximated with truncated Neumann series, reducing the computational cost from matrix inversion to matrix-matrix multiplications.

Remark 4 *Calculating the gradients of equilibrium flows with respect to demand or supply-side perturbations has been studied as equilibrium flow sensitivity analysis in the transportation literature. Tobin and Friesz (1988) showed that the Jacobian exists if the utilized path set remains the same with a small perturbation in parameters. Patriksson (2004) further suggested that the*

Jacobian exists if all unused paths remain unused with the perturbation. Li et al. (2020) pointed out the Jacobian exists if the cost function is strongly monotone in a neighborhood of h^*. These conditions may not hold in a general setting. However, the aforementioned numerical methods work well in our numerical experiments.

To sum up, leveraging the hypergradient approximated under ITD and IMD, the parameter for epoch k is updated with learning rate $\beta > 0$ as:

$$\theta_{n+1} = \theta_k - \beta \widehat{\nabla} \Phi(\theta_k)$$

Here we adopt a warm-start strategy by setting the initialization as the output of the preceding training epoch rather than initiating it with random values.

6 Numerical Examples

In this section, we validate the proposed end-to-end framework using three synthesized datasets from Braess, Sioux Falls, and Chicago Sketch. We use the Braess example to validate the approximation guarantee of the end-to-end framework. Through the Sioux Falls example, we examine the effect of enforcing equilibrium constraints and provide practical guidelines for training. The Chicago Sketch example demonstrates the simultaneous learning of supply- and demand-side components. We evaluate the framework performance using two key metrics: the empirical optimality gap, which measures the convergence of the parametric VI, and the Weighted Mean Absolute Percentage Error (WMAPE), which quantifies percentage differences in flow predictions. We define the empirical optimality gap as the sample-averaged inner product between the generalized cost function and the changes in the response variable across two successive.

6.1 Example 1: learn demand component on Braess

The Braess network has five links, four nodes, and a single OD pair from node 1 to node 4 with three feasible paths. With a maximum possible demand of q = 5, the ground-truth demand function for OD pair r follows:

$$u_r(e_r, x^{[m]}) = \alpha_u \cdot \bar{t}_r \cdot x^{[m]} \cdot \exp(\beta_u \cdot x_r \cdot x^{[m]} \cdot e_r))$$

where x_r represents OD-specific features; $x^{[m]}$ is a one-dimensional sample-dependent contextual feature; $\bar{t}_r$is the shortest free-flow time between OD pair r; α_u = 2 and β_u = 4 are functional parameters. We use the standard BPR function as link performance functions and assume travelers only consider travel time when selecting their paths. The dataset includes 1024 training, 258 validation, and 258 testing samples.

We will focus on learning the inverse demand function in this example and assume both link performance and cost functions are given. We consider that multi-day link flows are observable, and the loss function measures the Mean Square Error (MSE) between predicted and observed link flow distributions. The framework is trained using the Adam optimizer over $K = 500$ epochs with early stopping implemented if there is no improvement in the training MSE over twenty consecutive epochs. The forward propagation uses mirror descent with $N = 100$ iterations, while backpropagation employs the ITD method.

We evaluate both model-free and model-based end-to-end frameworks under the following scenarios, fine-tuning the learning rate and step sizes via grid search for each setting.

- *Benchmark:* We use a grid search to identify a constant demand that best matches all testing samples, which is 2.4 in this case.

- *Functional:* Assuming the functional form is known and encoded with computational graphs, the framework learns two parameters: α_u and β_u.

- *Constant:* The framework learns a context-independent fixed demand, with neural networks using only excess demand and OD-specific features as input.

- *Linear:* The neural network includes a single linear layer.

- *Nonlinear:* The neural network combines a linear part (as in the Linear scenario) and a nonlinear part, comprising three layers with eight neurons each.

- *Residual:* The neural network includes three layers with eight neurons each and employs a residual strategy between layers.

Each neural network is designed to accommodate potential changes in the number of OD pairs during ”what-if” analyses. The input dimension of these neural networks only depends on the number of input features, which in this case, is three. In the Nonlinear and Residual scenarios, neural networks are regularized to be monotone and Lipschitz continuous.

Table 1 presents the WMAPE under different learning scenarios. WMAPE is shown in percentage and parentheses display the relative reduction in WMAPE. The optimal scenario is highlighted with a star. Same for the following tables. In Benchmark scenario, the link flow WMAPE is remarkably high at 72.1%. This error drops to 4.2% when we embed the ground-truth functional form in the framework and adjust the αu and βu. The non-zero error can be attributed to the nonconvexity of the MPEC, which can trap the training process at a local minimum. The model-free Constant scenario learns context-independent demands and yields an error of 72.3%, comparable to Benchmark. The Residual scenario knows contextual information but has no information about the functional form of the inverse demand function. By exploring the representation power of neural networks, the model-free framework still yields a WMAPE of 4.7%, comparable to the Functional scenario. This result confirms that the end-to-end framework can generate reliable flow distributions without knowing each component’s functional form. The Residual scenario provides the best performance since the residual strategy helps avoid the gradient vanishing when N becomes large

Model	# Parameters	Link flow	Link time	Demand
Benchmark	/	72.1	31.2	71.9
Functional	2	4.2 (-94.1%)	1.6 (-94.9%)	3.9 (-94.6%)
Constant	109	72.3 (+ 2.8 %)	31.28 (+2.5%)	72.4 (+ 7.0%)
Linear	4	15.5 (-78.5%)	5.6 (-82.1%)	12.9 (-82.0%)
Nonlinear	117	6.3 (-91.2%)	4.3 (-86.2%)	6.2 (-91.3%)
Residual *	112	4.7 (-93.5%)	3.2 (-89.7%)	4.6 (-93.6%)

Table 1 WMAPE under different scenarios

6.2 Example 2: learn demand component on Sioux Falls

Sioux Falls network has 76 links, 28 nodes, and 528 OD pairs. We scaled the default demand in Stabler (2023) by a factor of three to serve as the maximum possible OD demand q. The rest of the ground-truth setting follows the Braess example. The dataset is divided into 1024 training, 258 validation, and 258 testing samples. We consider a link-based formulation using the decoupled gradient-projection method in forward propagation. With known link performance and cost functions, our focus is on learning the inverse demand function.

We first investigate the framework performance with different forward steps. Figure 6 shows that increasing N from 1 to 50 enables faster and better training under both IMD and ITD. A larger N requires more iterations for both forward and backward propagation and notably increases computation time under ITD. By contrast, IMD

avoids the differentiation along the equilibrating trajectory and the computation time changes relatively minimally when N varies.

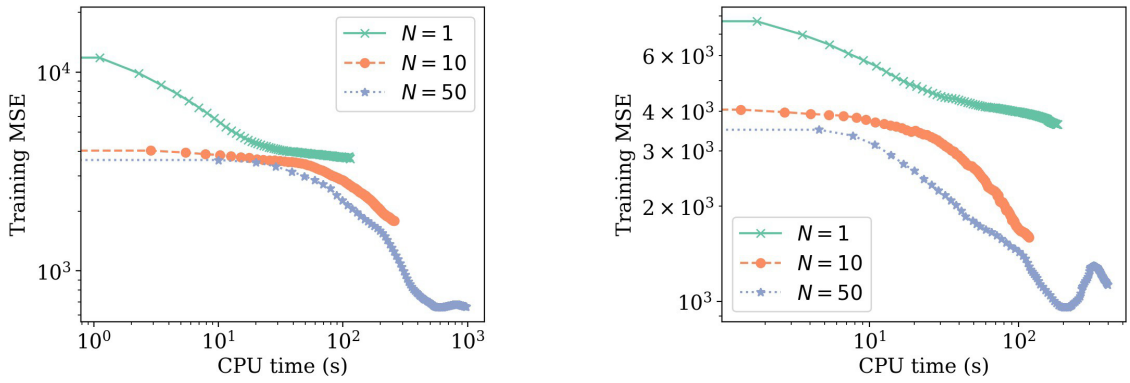
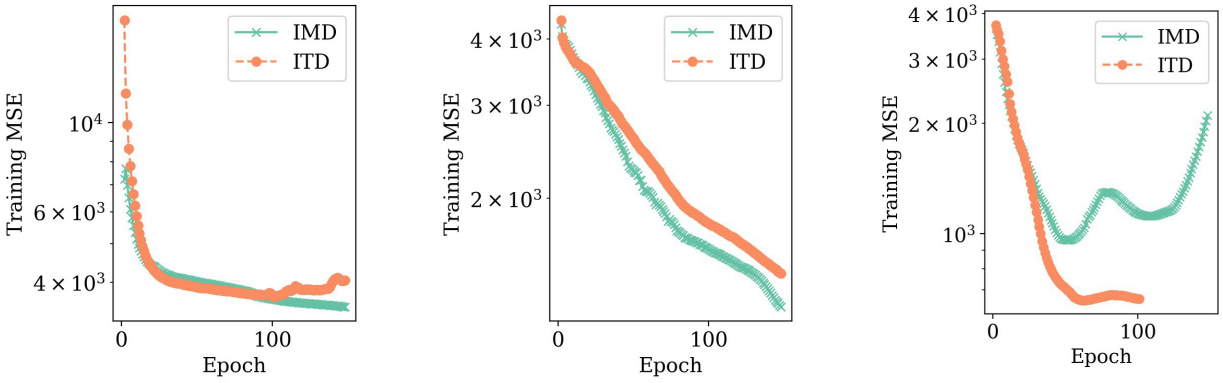


Figure 6 Figure 6: Framework performances with different forward iterations under (a) ITD and(b) IMD.

Moreover, Figure 7 shows that the training process under ITD stops prematurely with $N = 1$, resulting in a high training MSE of 4e3. This highlights an iterative equilibrium process is necessary to ensure local convergence under ITD. By contrast, IMD keeps reducing the training MSE with $N = 1$ because it uses extra information from the implicit function theorem to correct auto-differentiation. Both ITD and IMD manage to avoid getting stuck when N increases to 10. ITD outperforms IMD in finding better local optima when N increases to 50.

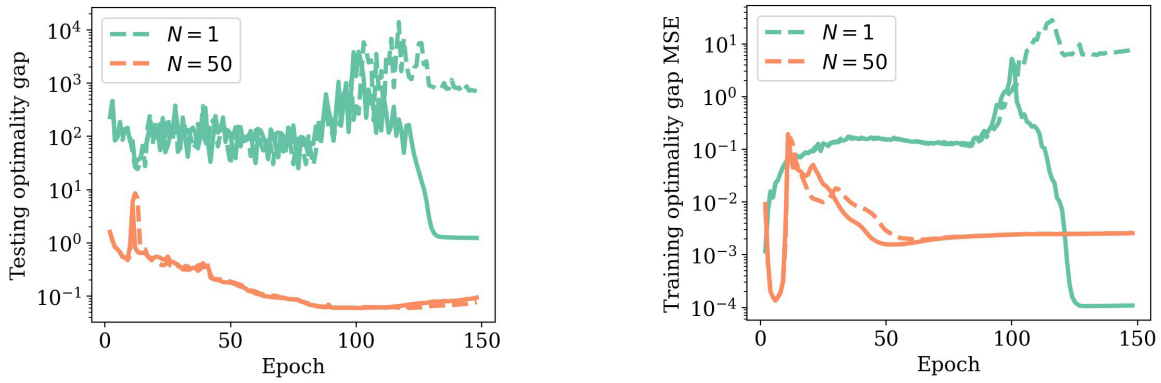
Next, we examine whether penalizing the empirical optimality gap in the loss function can replace the need for enforcing equilibrium conditions during training. As illustrated in Figure 8, Scenarios with optimality gap regularization are represented by solid lines, while those without are denoted by dotted lines. This pattern applies to Figure 9 as well. When the equilibrium constraints are poorly approximated with $N = 1$, the optimality gap regularization indeed steers the parametric VI towards a smaller empirical optimality gap. As the training proceeds, the optimality gap MSE converges towards zero (see Figure 8b). Similar findings have been found in (Guarda et al., 2023). By contrast, when the

equilibrium constraints are well-approximated with $N = 50$, the optimality gap regularization has little impact on framework performance.



(a) (b) (c)

Figure 7 Framework performances using different backward method with (a) $N = 1$, (b) $N = 10$, and (c) N = 50.



(a) (b)

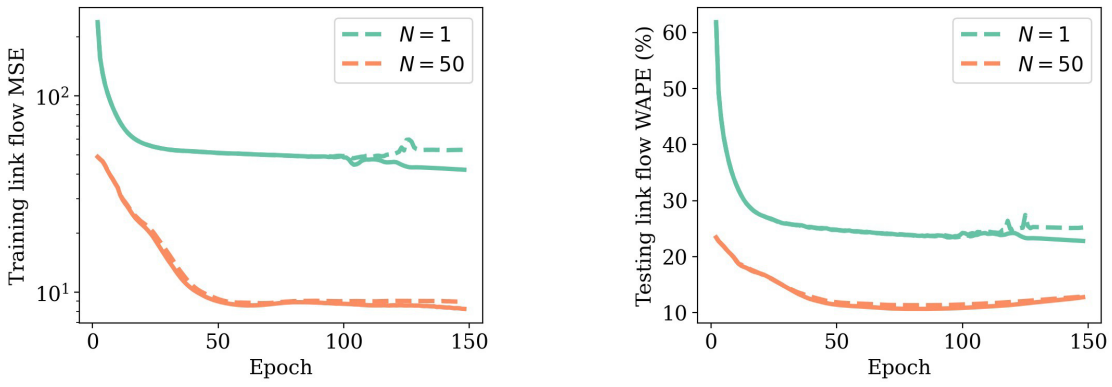
Figure 8 (a) Testing optimality gap and (b) training optimality gap MSE with respect to epochs.

Despite guiding the training process towards an equilibrium state, the optimality gap regularization fails to lead the framework to find suitable parameters. As shown in Figure 9, the training MSE with optimality regularization and $N = 1$ remains noticeably higher than that with $N = 50$. The link flow prediction error with $N = 1$ is also significantly larger. This suggests that ”softly” penalizing the optimality gap in the loss function is not a viable alternative to the ”hard” enforcement of equilibrium conditions. Therefore, it is essential to at least roughly approximate the equilibrium conditions to facilitate effective end-to-end learning.

Finally, we experiment with two enhanced training strategies:

- **Adaptive N (denoted as A):** Increases the number of forward iterations linearly during training, from 50 to 150 in our case.

- **Two-stages training (denoted as T):** Initially, the linear part of the neural network is trained while keeping the nonlinear part fixed. Once the linear part converges, both parts are trained jointly in the second stage.



(a) (b)

Figure 9 (a) Training link flow MSE and (b) testing link flow WMAPE with respect to epochs.

In the Benchmark scenario, we proportionally scaled the maximum possible OD demands q and use grid search to determine the optimal scale that best matches all observations, which is 1.2 in this case. The remaining Functional, Nonlinear, and Residual scenarios follow the Braess example with $N = 50$ and ITD as the backpropagation method. Table 2 indicates that the adaptive N strategy improves the model's performance because a rough estimation of equilibria is sufficient when the parameters are considerably off-target during the initial training epochs. The two-stage training strategy also enhanced performance because it trains a shallow linear network in the first stage. On one hand, a linear approximation of the monotone generalized cost function is relatively good. On the other hand, shallow neural networks mitigate vanishing or exploding gradients during training. Thus, incorporating both strategies, our end-to-end framework achieves the best performance of 4.3%, comparable to the Functional scenario (i.e., 1.3%).

In this example, we calculate WMAPE only for flows over 0.001 to avoid infinite WMAPE due to zero ground-truth flows in training samples. Thus, despite Sioux Falls' larger size, its WMAPE is numerically smaller than Braess. Since our main concern is the relative WMAPE reduction, rendering this should be insignificant to our conclusions.

6.3 Example 3: learn behavior component on Sioux Falls

In this case study, each OD pair r is assumed to have one continuous feature x_r^1 denoting income and one binary feature x_r^2 denoting travel purpose, which equals 1 if the destination of OD pair r is a commercial area and equals 0 otherwise. We assume the path travel time includes two parts: link travel times and node delays. The link travel time on link a follows the BPR function. The node delay on node c follows an exponential form as proposed by Jeihani et al. (2006), Moreover, pavement surface

conditions, such as roughness, are the main feature that decides user comfort (Hawas, 2004; Yin et al., 2008). We classify the links as good and bad pavement conditions and assume travelers experience a non-link-additive discomfort ep on bad-condition links. Let $0 \leq x_p \leq 1$ denote the proportion of bad-condition link length to the total path length. The discomfort follows the exponential form and increases with the bad-condition link proportion, i.e., $e_p = \exp(\alpha x_p) + \beta$. We set $\alpha = 2$ and $\beta = 1$ so that the discomfort is zero if path p only includes goodcondition links.

Scenario	# Parameters	Link flow	Link time	Demand
Benchmark	/	50.9	97.6	59.3
Functional	2	1.3 (-97.4%)	3.7 (-96.2%)	1.3 (-97.8%)
Linear	4	14.1 (-72.3%)	40.9 (-58.1%)	6.4 (-89.2%)
Nonlinear	117	10.2 (-80.0%)	12.1 (-87.6%)	6.8 (-88.5%)
Nonlinear (+ T)	117	9.0 (-82.3%)	24.4 (-75.0%)	5.6 (-90.6%)
Nonlinear (+ A)	117	7.9 (-84.5%)	22.1 (-77.4%)	4.9 (-91.7%)
Nonlinear (+ T + A) *	177	4.3 (-91.5%)	9.2 (-90.5%)	2.7 (-95.4%)
Residual	112	12.8 (-74.9%)	37.8 (-61.4%)	7.2 (-87.9%)
Residual (+ A)	112	8.3 (-83.7%)	24.4 (-75.0%)	5.2 (-91.2%)

Table 2 WMAPE under different training settings

The ”ground-truth” cost for travelers of OD pair r to use path p is a weighted sum of link travel times, node delays, and a discomfort constant:

$$c_p = \sum_{a \in p} t_a(v_a) + w_r^d \sum_{n \in p} d_n(\overline{u}_n) + w_r^e e_p,$$

This suggests that travelers with higher incomes have higher weights on both node delays and discomfort. Travelers traveling to commercial areas have higher weights on discomfort yet lower weights on node delays.

The feasible path set includes the top three paths with the shortest free-flow time. If one OD pair has fewer than three feasible paths, its path flows are padded to a dimension of three and the padded path flows are nullified with the mask trick during training. Three demand levels are considered: (i) base scenario q^0, (ii) uncongested scenario with base demand q^0 reduced by 50%, and (iii) congested scenario with base demand q^0 increased by 50%. For each scenario, we randomly sample travel demands from a uniform distribution between 0.5 q^0 and 1.5 q^0. The equilibrium flow is solved for each sampled demand given the ground-truth cost. The training and test sets include 1, 536 and 512 samples respectively. So far, all links are assumed to be observable.

The link block is replaced with pre-calibrated BPR functions. Weight Net, node block, and path block are composed of three fully connected layers with four neurons and with LeaklyReLu as the activation function. Normalization layers are added to enhance training stability. The input of the node block includes node flows and intersection parameters. The proportion of bad-condition links is the input of the path block. The input and output dimensions are as follows: Weighted ergodic iteration and IMD are used as the default forward and backward methods respectively. The model is trained with Adam optimizer with Mean Square Error as the loss function under the learning rate of 0.1. Early stop is enabled if no loss descent is observed in five consecutive epochs. To illustrate the feasibility and importance of learning route choice preferences, we benchmark our model with three well-established network equilibrium models. First, the cost function is assumed to be link travel time and travelers choose the paths with

minimum travel time, yielding conventional Deterministic User Equilibrium (denoted as DUE). The second behavior model assumes travelers' path choices follow a logit model and thus results in a Stochastic User Equilibrium (denoted as SUE). In this case, the dispersion parameter is calibrated, similar to Yang et al. (2001). The third model keeps the same path choice model but assumes the cost function is a linear combination of link travel time and the proportion of bad-condition links (denoted as SUE-2). Two linear coefficients are calibrated in this case, similar to Guarda and Qian (2022).

We compare the efficiency and robustness of different forward algorithms. The first type includes decoupled gradient-projection iteration (F) and its accelerated variant: Anderson mixing (FA) and weighted ergodic iteration (FW). The second type is Broyden's method (R). We also explore the combinations of two types (denoted as F-R, FA-R, FW-R), which use decoupled gradient-projection iterations initially and switch to the root-finding when the relative residential is sufficiently small. We consider two types of tests: in-distribution and out-of-distribution. In in-distribution tests, the model is trained on observations from the Sioux Falls network and tested on the same road network. By contrast, in out-of-distribution tests, the trained model is tested on a partially changed road network. In our experiments, four links are added to the original Sioux Falls network and 25% links are randomly selected to increase or decrease their capacities by 50%. Decreasing the capacities under congested demand generates unreasonable training sets and is excluded in later analysis.

6.3.1 Performance comparisons

Table 3 compares the MAPE of different network equilibrium models. The proposed end-to-end learning framework is denoted as "Implicit". We use DUE as the baseline and denote its MAPE as $\eta 0$. The change in MAPE of other models is denoted as $\Delta\eta = \eta - \eta_0$. Note that the behavioral assumptions of SUE are different from the ground truth. Although SUE can reduce the in-distribution MAPE by 18.2%, it shows inferior performance in out-of-distribution tests, increasing the MAPE by 9.2%. This suggests inaccurately assuming an SUE behavior model can cause bias in parameter estimation,

misleading the flow prediction in subsequent" what-if" analysis. Similar results have been shown in Torres et al. (2011) and Van Der Pol et al. (2014). In comparison, SUE-2 performs better, because it happens to capture the impact of discomfort from the bad-condition links. The performance of SUE-2 is still less satisfactory compared with the end-to-end framework because the former learns linear combinations by assumption whereas the latter can deal with nonlinear patterns. Since neural networks include more parameters than baseline models and offer greater flexibility to recover the complicated ground truth cost function, the proposed framework has the best performance in both in-distribution and out-of-distribution tests as expected, reducing the benchmark MAPE by 61.5% and 55.1% respectively.

In-distribution test

Demand	Capacity	DUE η_0	SUE $\Delta\eta$	SUE-2 $\Delta\eta$	Implicit $\Delta\eta$
Base	Default	20.6	-4.7	-11.8	-15.0
Uncongested	Default	12.5	-3.1	-0.2	-3.4
Congested	Default	13.41	-0.6	-4.4	-10.2
Mean		15.5	-2.8 (-18.2%)	-5.4 (-35.1%)	-9.5 (-61.5%)

Out-of-distribution test

Demand	Capacity	DUE η_0	SUE $\Delta\eta$	SUE-2 $\Delta\eta$	Implicit $\Delta\eta$
Base	Default	22.3	-7.3	-14.4	-16.6
	-50%	11.3	+13.4	-1.6	-7.9
	+50%	8.1	+4.8	-1.0	-1.3
Uncongested	Default	23.4	-8.5	-15.6	-14.9
	-50%	12.1	+12.6	-2.4	-4.1
	+50%	10.4	+2.5	-3.3	-1.1

Congested	Default	13.8	-3.5	-6.3	-10.1
	+50%	11.9	-3.5	-5.2	-6.4
Mean		14.2	+1.3 (+9.2%)	-6.2 (-44.0%)	-7.8 (-55.1%)

Table 3 MAPE of different network equilibrium models.

As shown in Table 4, FW and FW-R achieve the smallest MAPE of 5.7% in in-distribution tests whereas FW-R slightly outperforms FW by 1% in out-of-distribution tests. Forward algorithms involving Anderson mixing, such as FA and FA-R, can be the most unstable. By contrast, forward algorithms involving weighted ergodic iteration, such as FW and FW-R, are more stable as they consistently shrink the step size during iterations.

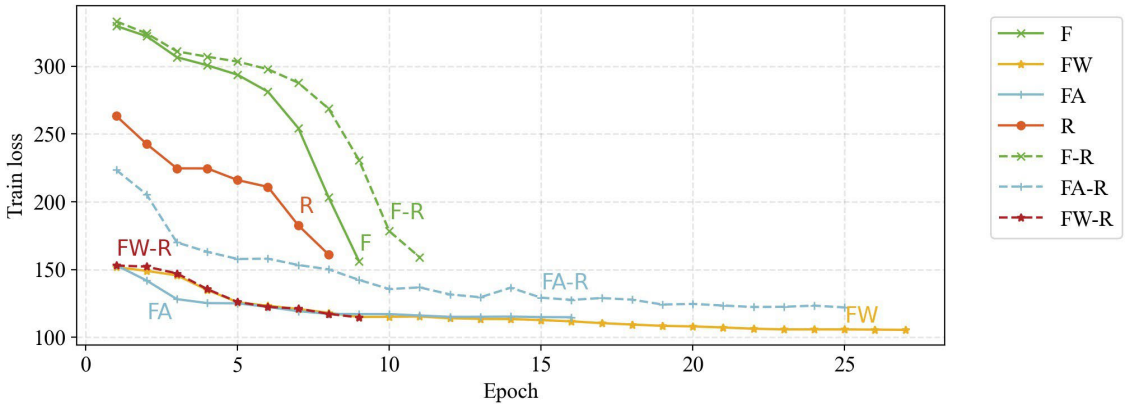


Figure 10 Training process of different forward algorithms.

Figure 11 compares the performance of three backpropagation methods: Jacobian-Free (JF) approximation, Newman Approximation (NA), and Inexact Implicit Differentiation (FA) under different demand levels. FA has the best performance among the three proposed backward methods. JF significantly hurts the learning process. Similar results have been found by Huang et al. (2021).

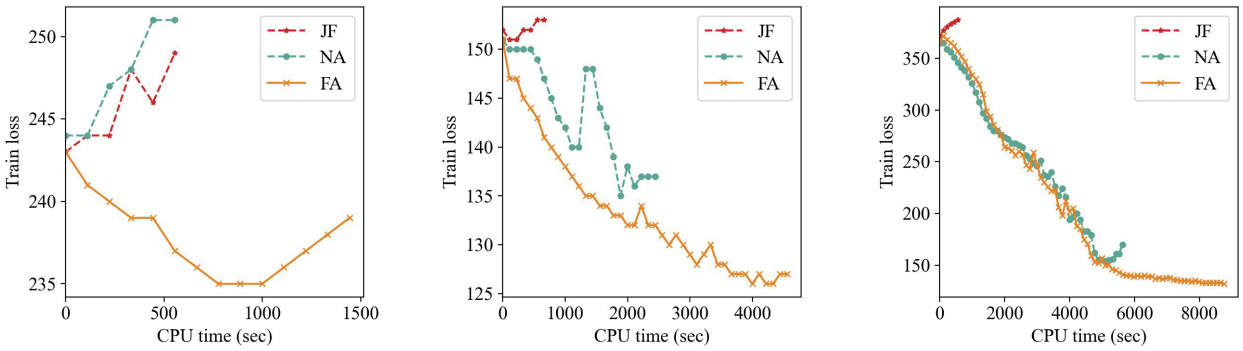
The effects of spectral normalization are shown in Figure 12. ”w” suggests ”with spectral normalization” and ”w/o” suggests ”without spectral normalization”. Although requiring additional computation, the spectral normalization constrains the Lipshitz constant of the cost function within a reasonable range and speeds up the convergence by three to four times under all demand levels.

6.3.2 Robustness analysis

In-distribution test								
Demand	Capacity	F	FA	FW	R	F-R	FA-R	FW-R
Base	Default	8.4	5.6*	5.7	8.0	8.7	6.2	6.0
Uncongested	Default	9.5	9.1	8.1	9.5	8.3	8.5	8.0*
Congested	Default	6.1	3.2	3.2	6.2	11.0	4.5	3.1*
Mean		8.0	6.0	5.7*	7.9	9.3	6.4	5.7*
Std		1.8	3.0	2.4	1.7	1.4	2.0	2.5
Out-of-distribution test								
Scenario	Capacity	F	FA	FW	R	F-R	FA-R	FW-R
Base	Default	7.5	5.7*	5.8	7.2	7.7	6.4	6.0
	-50%	4.5	3.4*	3.4*	5.0	4.8	3.6	3.4*
	+50%	9.1	6.9*	7.0	10.2	9.3	8.8	7.4
Uncongested	Default	8.3	8.5	7.6	8.1	8.0	8.2	7.5*
	-50%	9.5	8.0	7.3	9.9	7.6	14.4	6.9*
	+50%	8.4	9.4	7.9*	8.4	8.2	7.9*	7.9*
Congested	Default	5.7	3.6*	3.8	5.1	10.2	4.3	3.6*
	+50%	8.8	5.5*	5.7	6.2	11.9	6.1	5.5*
Mean		7.7	6.4	6.1	7.5	8.5	7.5	6.0*
Std		1.8	2.2	1.7	2.0	2.1	3.4	1.7

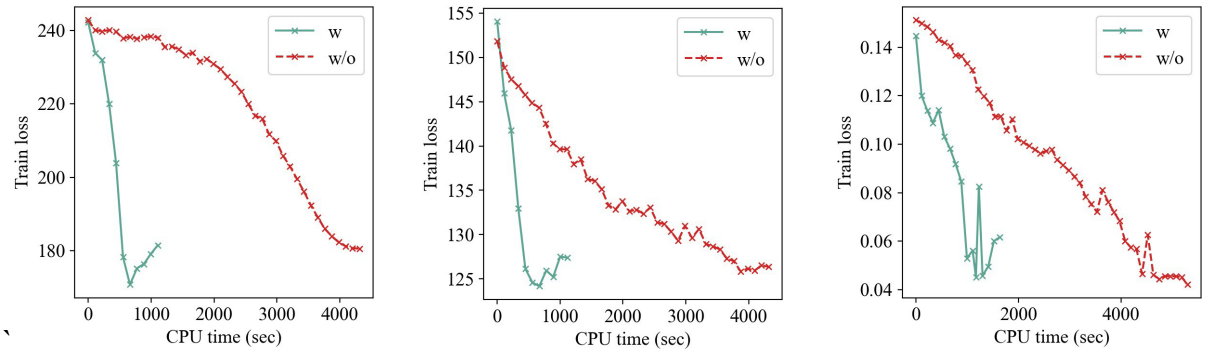
Table 4 MAPE of proposed forward algorithms.

In this section, we examine the robustness of the proposed framework by relaxing model assumptions. FW, R, and FW-R have the best performance and are thus selected. Since in-distribution and out-of-distribution performances have similar trends, all the following analyses are based on in-distribution tests.



(a) (b) (c)

Figure 11 Performances of different backpropagation methods under (a) base, (b) uncongsted, (c) congested demand.



(a) (b) (c)

Figure 12 Effects of spectral normalization under (a) base, (b) uncongested, and (c) congested demand.

All links are assumed to be observable in previous analyses. We relax this assumption by randomly observing a proportion of links. FW-R is the most stable when only a proportion of links are equipped with sensors. For example, Figure 13 shows the MAPE of FW-R slightly increases from 8.0% to 11.5% when the proportion of observable links decreases from 100% to 20% under uncongested demand. Since approximation errors can accumulate in both forward propagation, where iterations terminate with residuals, and backward propagation, where the gradients are approximated, the training of the proposed framework can stop at local optimums. Previous studies have shown the training process and final performances of models involving implicit layers can be relatively noisy and require more hyperparameter tuning (Huang et al., 2021; Li et al., 2020).

Usually, there are no direct observations of OD demands in urban road networks. OD demands need to be estimated and thus prone to estimation errors. We examine the model performances when the input OD demands are different from the ground truth. More specifically, random observation noises, which are proportional to the ground

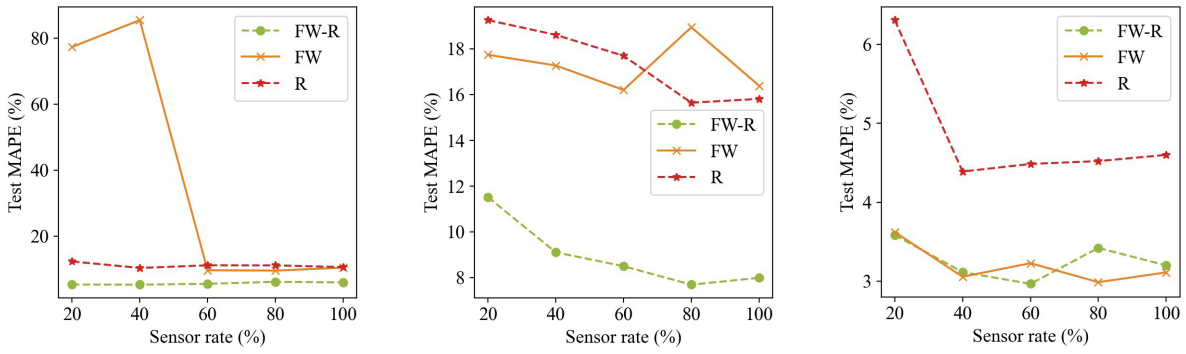


Figure 13 Model performances with different sensor coverage rates under (a) base, (b) uncongested, and (c) congested demand.

truth, are added to all demands. As shown in Figure 14, FW is the most stable in the case of demand noises. Given a noise scale of 100%, the increase in its MAPE ranges from 12.5% to 22.2% under different demand levels. Note that if we consider an elastic demand user equilibrium, the travel demand function can also be approximated with another neural network and learned with the proposed framework. The simultaneous learning of route choice preferences and demand functions will be explored in our future study.

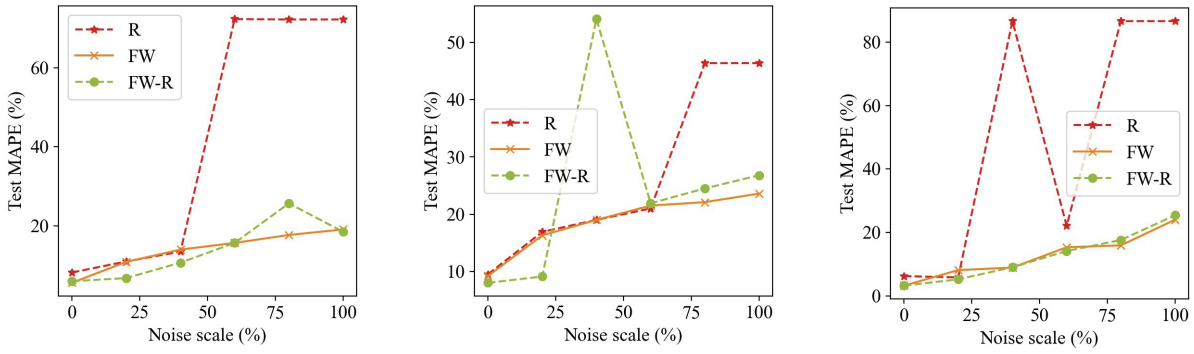


Figure 14 Model performances with demand noises under (a) base, (b) uncongested, and (c) congested demand.

The selection of feasible path sets can be tricky when no information about path choices is available. We examine the model performances when the selection of feasible paths is different from travelers' actual path choices. There are 1,587 paths in the ground-truth path set and we consider two scenarios: one with an incomplete path set of 1,058 paths and the other with a redundant path set of 2,645 paths. FW-R has the best performance when the selection of feasible paths is inaccurate. As shown in Figure 15, an incomplete path set increases the MAPE by 8.0% under base demand, compared with an increase of 2.9% induced by a redundant path set. Since an incomplete path set yields more negative effects, one can start with a large feasible set with sufficient feasible paths and gradually reduce it during training.

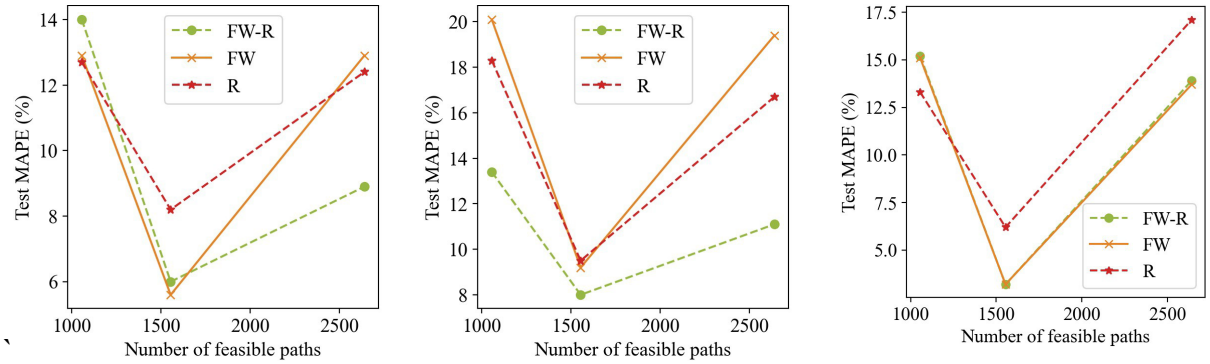


Figure 15 Effects of inaccurate feasible path sets under (a) base, (b) uncongested, and (c) congested demand.

To sum up, the proposed framework is robust to incomplete observations and input noises. More specifically, the combined method (i.e., FW-R) is more robust when only a proportion of links are equipped with sensors or no information about path choice is available. The fixed-point iteration method (i.e., FW) is preferred when the input OD demands are poorly estimated.

6.4 Example 4: learn demand and supply component on Chicago Sketch

We consider a path-based formulation on the Chicago Sketch with 2950 links, 933 nodes, and 2493 OD pairs. Each OD pair has three feasible paths, and the feasible path set is assumed as prior information. We scale the default demand in Stabler (2023) by a factor of five and use it as the maximum possible OD demand. The following inverse demand function is used and the ground-truth BPR function is assumed with a context-dependent capacity for each link $a \in A$:

$$cap_a(x) = cap_a^0 \cdot (\alpha_c \cdot e^x + \beta_c)$$

where cap_a^0 is the default capacity; $\alpha_c = 1.5$ and $\beta_c = 1.4$. The dataset contains 258 training, 64 validation, and 64 testing samples. We assume the cost function is known

and focus on learning the inverse demand function and link performance function. Mirror descent with a forward step of $N = 10$ and ITD are used in training.

The end-to-end framework is set to learn the inverse demand function, the link performance function, or both, using either a model-based or a model-free approach. In the model-free setting, the inverse demand function is approximated using the residual neural networks specified in the Sioux Falls example. We employ a physics-informed neural network to learn the link-performance function. We retain the functional form of the BPR function and approximate the context-dependent capacities using neural networks with three layers and eight neurons each. Additionally, both link time and flows are assumed observable, enabling modelers to include either or both of these observations in the loss function We consider two benchmarks with fixed capacities in the standard BPR function. Benchmark-1 scales the default demand with a factor of 3.56 and achieves the best match to observed flows (29.5%) with a high time error of 160.5%. Benchmark-2 scales the default demand with a factor of 1.4 and achieves the best match to observed time (5.1%) with a high flow error of 68.9%.

Table 5 shows the performance of the end-to-end framework with different learnable components and loss functions. Scenarios yielding the lowest errors are marked: a single star denotes the best model-free scenario, while double stars indicate the best model-based one. The joint calibration of supply and demand-side components proves important. Both Functional and Residual scenarios, when adjusting both sides, yield the lowest time and demand errors. The Functional scenario has the lowest flow error of 18.1% and time error of 2.6%, while the Residual scenario generates comparable results of 23.3% and 8.2%. Incorporating flow observations into the loss function in general outperforms the use of link time. Nevertheless, using link time observations can help avoid overfitting when the link performance function can be adjusted. Overall, the complexity of training escalates with the size of the road network. The Chicago sketch example has higher errors than Sioux Falls and Braess, regardless of the approach used.

Component	Approach	Loss function	Link flow	Link time	Demand
Benchmark-1	/	/	29.5	160.5	21.5
Benchmark-2	/	/	68.9	5.1	68.3
λ_θ	Functional	Flow	22.1	27.1	14.2
		Time	28.3	9.2	21.9
		Flow + time	22.1	27.1	14.1
	Residual	Flow	23.5	64.8	15.3
		Time	23.5	54.2	16.2
		Flow + time	23.3 *	65.8	15.1
τ_θ	Functional	Flow	32.9	1751.8	19.6
		Time	39.3	8.6	19.6
		Flow + time	32.8	1734.3	19.6
	Residual	Flow	34.1	15.8	19.6
		Time	36.5	16.0	19.6
		Flow+ time	32.8	15.9	19.6
λ_θ and τ_θ	Functional	Flow	18.1 **	2.6 **	7.7 **
		Time	22.3	5.1	9.5
		Flow + time	18.3	2.6	7.9
	Residual	Flow	26.9	8.2 *	13.6 *
		Time	37.6	10.4	19.0
		Flow + time	25.0	193.9	13.9

Table 5 WMAPE under different training settings.

7 Findings and Conclusions

This study aims to transform the modeling paradigm via an end-to-end framework that directly learns model components and the equilibrium state from data. This report outlines our solutions to the modeling and algorithmic challenges for implementing such an end-to-end learning. The unified end-to-end framework encodes the unknown supply- and demand-side model components with parameterized computational graphs and then embeds them in a VI that enforces user equilibrium conditions. In forward propagation, the framework iteratively updates the traffic state via closed-form rules until reaches user equilibrium. In backpropagation, the loss function compares the estimated and observed traffic states and then simultaneously estimate parameters for all components via auto-differentiation.

One major advantage of the proposed end-to-end framework is that it integrates model-based and model-free modeling approaches within a single pipeline, leveraging both domain knowledge and the representational power of neural networks the proposed framework leverages. The proposed framework automatically discovers a good functional specification from empirical data during training and aligns the selection of behavior models with the ultimate goal of replicating flow distributions.

More importantly, the end-to-end framework learns the equilibrium state of the network. Because real systems never settle into equilibrium, observed flows are indeed not equilibrium flows. The training process essentially yields an equilibrium state that matches all the observations as closely as possible. The learned equilibrium state will then serve as a consistent benchmark or reference point against which improvement plans can be designed and compared. The resulting equilibria are ”perturbed” from the learned equilibrium state and will help decision-makers differentiate various plans. In this sense, the proposed framework melds the data-decision pipeline by integrating learning and decision/optimization into a single end-to-end system. Additionally, the end-to-end framework integrates multi-source data into a single stream and addresses the inconsistencies among different data sources.

Our study solves the key challenges in modeling and calibrating the unified ”end-to-end” framework. To facilitate model-free approach, we design a novel neural network architecture that can adjust to the changes in the road network topology for future ”what-if” planning analysis. We also regularized the neural network to guarantees the existence of equilibrium traffic states. To efficiently train the proposed framework, we introduced an auto-differentiation-based gradient descent algorithm and leverage the computational power of computational graphs. In forward propagation, we adopt recent developments in operator-splitting methods and differential optimization to solve a batch of VI problems. We employ first-order methods like decoupled gradient projection and mirror descent, specifically tailored for link- and path-based equilibrium constraints, as well as second-order root-finding methods. In backpropagation, iterated differentiation and inexact implicit differentiation are used to efficiently differentiate through the equilibrium states.

To validate the robustness and efficacy of the proposed framework, we conduct a series of numerical experiments on synthesized data from various networks, including Braess, Sioux Falls, and Chicago Sketch. Our framework achieves a satisfactory accuracy rate in predicting link flows when subjected to changes in road network topology. Additionally, the model demonstrated robust performance in the face of incomplete data and various input noises.

The outputs, outcomes, and impacts of this study are summarized as follows:

Research Outputs

- *Publication:* "End-to-end learning of user equilibrium with implicit neural networks." Transportation Research Part C: Emerging Technologies 150 (2023): 104085.

- *Poster:* End-to-end learning of user equilibrium with neural networks. Transportation Research Board 102ed Annual Meeting. Washington, D.C. 2023.

- *Presentation:* End-to-end Learning of Transportation Network Equilibrium. INFORMS Annual Meeting, Indianapolis, IN. 2022.

- *Presentation:* A Unified Framework for End-to-End Transportation Network Equilibrium Modeling. International Symposium on Transportation Data & Modelling. Ispra, Italy. 2023

Research Outcomes

- *Policy Planning Support:* The framework serves as a decision support tool for policymakers exploring various improvement schemes like capacity expansion and congestion pricing.

- *Network Modeling as a Service:* The framework can be deployed as a cloud-based service, allowing cities and municipalities to access advanced modeling capabilities without the need for specialized hardware or expertise.

Research Impacts

- *Traffic Congestion:* The framework can help reduce traffic congestion by optimizing transportation network planning for future scenarios.

- *Resource Allocation:* Policymakers can make more informed decisions about where to allocate resources, potentially saving public money by avoiding unnecessary infrastructure development.

- *Operational Efficiency:* Automated tools that implement the framework could lead to more efficient operations within transportation departments, reducing both capital and operational costs.

8 Recommendations

This study can be extended in multiple directions. We plan to leverage the established end-to-end learning framework to prescribe improvement schemes, such as capacity expansion and congestion pricing. We consider that policymakers attempt to perturb the equilibrium flow distribution by changing certain continuous decision variables that would affect travelers' route choices. These decision variables can be encoded as additional learnable parameters in Attribute Net. By maximizing the expected social welfare, the proposed end-to-end framework can be trained to update the decision variables and output optimal decisions. The optimization problem becomes more challenging when dealing with discrete decision variables, such as the incorporation of new roads or lanes. This will be a focus of our future studies.

Additionally, the proposed framework has been tested on a synthesized dataset. We plan to validate the proposed framework with real-world datasets in the next step. Vehicle connectivity and automation will make trajectory data more readily available. Leveraging this dataset, the proposed modeling paradigm, if successful, can potentially help metropolitan planning organizations and traffic authorities in the US better plan and manage their traffic networks to reduce traffic congestion and vehicle emissions, without requiring new investment in expanding the existing infrastructure. With more and more connected vehicles, we believe that the solution would transform the existing paradigm of transportation systems planning and management and has a great potential for widespread market adoption. The proposed work will use real-world datasets to validate the proposed framework. If successful, further development beyond this project will be needed to develop a deployable platform or system that can automate the proposed processes to provide diagnosis and treatments for various urban traffic networks.

9 References

Bai, S., Kolter, J.Z., Koltun, V., 2019. Deep equilibrium models. Advances in Neural Information Processing Systems 32.

Beckmann, M., McGuire, C.B., Winsten, C.B., 1956. Studies in the Economics of Transportation. Technical Report.

Bibi, A., Ghanem, B., Koltun, V., Ranftl, R., 2019. Deep layers as stochastic solvers.

Chen, C., Ma, J., Susilo, Y., Liu, Y., Wang, M., 2016. The promises of big data and small data for travel behavior (aka human mobility) analysis. Transportation research part C: emerging technologies 68, 285–299.

Cheng, Q., Liu, Z., Guo, J., Wu, X., Pendyala, R., Belezamo, B., Zhou, X.S., 2022. Estimating key traffic state parameters through parsimonious spatial queue models. Transportation Research Part C: Emerging Technologies 137, 103596.

Dafermos, S., 1988. Sensitivity analysis in variational inequalities. Mathematics of Operations Research 13, 421–434.

Davis, D., Yin, W., 2017. A three-operator splitting scheme and its optimization applications. Set-valued and variational analysis 25, 829–858.

Emberton, J., 2008. An elucidation of vector calculus through differential forms.

Feng, Z., Narasimhan, H., Parkes, D.C., 2018. Deep learning for revenue-optimal auctions with budgets, in: Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems, pp. 354–362.

Fioretto, F., Mak, T.W., Van Hentenryck, P., 2020. Predicting ac optimal power flows: Combining deep learning and lagrangian dual methods, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 630–637.

Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R., Pontil, M., 2018. Bilevel programming for hyperparameter optimization and meta-learning, in: International conference on machine learning, PMLR. pp. 1568–1577.

Frejinger, E., Bierlaire, M., Ben-Akiva, M., 2009. Sampling of alternatives for route choice modeling. Transportation Research Part B: Methodological 43, 984–994.

Fung, S.W., Heaton, H., Li, Q., McKenzie, D., Osher, S.J., Yin, W., 2021. Fixed point networks: Implicit depth models with jacobian-free backprop.

Ghadimi, S., Wang, M., 2018. Approximation methods for bilevel programming. arXiv preprint arXiv:1802.02246.

Gouk, H., Frank, E., Pfahringer, B., Cree, M.J., 2021. Regularisation of neural networks by enforcing lipschitz continuity. Machine Learning 110, 393–416.

Guarda, P., Battifarano, M., Qian, S., 2023. Estimating network flow and travel behavior using day-to-day system-level data: A computational graph approach. Available at SSRN 4490930.

Guarda, P., Qian, S., 2022. Statistical inference of travelers' route choice preferences with system-level data. arXiv preprint arXiv:2204.10964.

Hawas, Y.E., 2004. Development and calibration of route choice utility models: factorial experimental design approach. Journal of transportation engineering 130, 159–170.

Heaton, H., McKenzie, D., Li, Q., Fung, S.W., Osher, S., Yin, W., 2021. Learn to predict equilibria via fixed point networks. arXiv preprint arXiv:2106.00906.

Huang, Z., Bai, S., Kolter, J.Z., 2021. Implicit layers for implicit representations. Advances in Neural Information Processing Systems 34, 9639–9650.

Jeihani, M., Lawe, S., Connolly, J., 2006. Improving traffic assignment model using intersection delay function. Technical Report.

Ji, K., Yang, J., Liang, Y., 2021. Bilevel optimization: Convergence analysis and enhanced design, in: International conference on machine learning, PMLR. pp. 4882–4892.

Kitthamkesorn, S., Chen, A., 2013. A path-size weibit stochastic user equilibrium model.

Procedia-Social and Behavioral Sciences 80, 608–632.

Li, J., Yu, J., Liu, B., Nie, Y., Wang, Z., 2023. Achieving hierarchy-free approximation for bilevel programs with equilibrium constraints, in: International Conference on Machine Learning, PMLR. pp. 20312–20335.

Li, J., Yu, J., Nie, Y., Wang, Z., 2020. End-to-end learning and intervention in games.

Advances in Neural Information Processing Systems 33, 16653–16665.

Li, J., Yu, J., Wang, Q., Liu, B., Wang, Z., Nie, Y.M., 2022. Differentiable bilevel programming for stackelberg congestion games. arXiv preprint arXiv:2209.07618.

Liu, Z., Yin, Y., Bai, F., Grimm, D.K., 2023. End-to-end learning of user equilibrium with implicit neural networks. Transportation Research Part C: Emerging Technologies 150, 104085.

Lu, J., Li, C., Wu, X.B., Zhou, X.S., 2023. Physics-informed neural networks for integrated traffic state and queue profile estimation: A differentiable programming approach on layered computational graphs. Transportation Research Part C: Emerging Technologies 153, 104224.

Ma, W., Pi, X., Qian, S., 2020. Estimating multi-class dynamic origin-destination demand through a forward-backward algorithm on computational graphs. Transportation Research Part C: Emerging Technologies 119, 102747.

Maclaurin, D., Duvenaud, D., Adams, R., 2015. Gradient-based hyperparameter optimization through reversible learning, in: International conference on machine learning, PMLR. pp. 2113–2122.

Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y., 2018. Spectral normalization for generative adversarial networks. arXiv preprint arXiv:1802.05957.

Patriksson, M., 2004. Sensitivity analysis of traffic equilibria. Transportation Science 38, 258–281.

Rahman, R., Hasan, S., 2022. Data-driven traffic assignment: A novel approach for learning traffic flow patterns using a graph convolutional neural network. arXiv preprint arXiv:2202.10508.

Ryu, E., Yin, W., 2021. Large-scale convex optimization via monotone operators (2020). URL https://large-scale-book. mathopt. com/LSCOMO. pdf.(visited on 03/2021) .

Sifringer, B., Lurkin, V., Alahi, A., 2020. Enhancing discrete choice models with representation learning. Transportation Research Part B: Methodological 140, 236–261.

Small, K.A., Chu, X., 2003. Hypercongestion. Journal of Transport Economics and Policy (JTEP) 37, 319–352.

Spana, S., Du, L., 2022. Optimal information perturbation for traffic congestion mitigation: Gaussian process regression and optimization. Transportation Research Part C: Emerging Technologies 138, 103647.

Stabler, B., 2023. Transportationnetworks. https://github.com/bstabler/TransportationNetworks. Tobin, R.L., Friesz, T.L., 1988. Sensitivity analysis for equilibrium network flow. Transportation Science 22, 242–250.

Torres, C., Hanley, N., Riera, A., 2011. How wrong can you be? implications of incorrect utility function specification for welfare measurement in choice experiments. Journal of Environmental Economics and Management 62, 111–121.

Travacca, B., El Ghaoui, L., Moura, S., 2020. Implicit optimization: Models and methods, in: 2020 59th IEEE Conference on Decision and Control (CDC), IEEE. pp. 408–415.

Van Der Pol, M., Currie, G., Kromm, S., Ryan, M., 2014. Specification of the utility function in discrete choice experiments. Value in Health 17, 297–301.

Virmaux, A., Scaman, K., 2018. Lipschitz regularity of deep neural networks: analysis and efficient estimation. Advances in Neural Information Processing Systems 31.

Walker, H.F., Ni, P., 2011. Anderson acceleration for fixed-point iterations. SIAM Journal on Numerical Analysis 49, 1715–1735.

Wang, S., Mo, B., Zhao, J., 2020. Deep neural networks for choice analysis: Architecture design with alternative-specific utility functions. Transportation Research Part C: Emerging Technologies 112, 234–251.

Wang, Y., Ma, X., Liu, Y., Gong, K., Henricakson, K.C., Xu, M., Wang, Y., 2016. A two-stage algorithm for origin-destination matrices estimation considering dynamic dispersion parameter for route choice. PloS one 11, e0146850.

Wardrop, J.G., 1952. Road paper. some theoretical aspects of road traffic research. Proceedings of the institution of civil engineers 1, 325–362.

Wehenkel, A., Louppe, G., 2019. Unconstrained monotonic neural networks. Advances in neural information processing systems 32.

Wong, M., Farooq, B., 2021. Reslogit: A residual neural network logit model for data-driven choice modelling. Transportation Research Part C: Emerging Technologies 126, 103050.

Wu, X., Guo, J., Xian, K., Zhou, X., 2018. Hierarchical travel demand estimation using multiple data sources: A forward and backward propagation algorithmic framework on a layered computational graph. Transportation Research Part C: Emerging Technologies 96, 321–346.

Xu, H., Lou, Y., Yin, Y., Zhou, J., 2011. A prospect-based user equilibrium model with endogenous reference points and its application in congestion pricing. Transportation Research Part B: Methodological 45, 311–328.

Yang, H., Meng, Q., Bell, M.G., 2001. Simultaneous estimation of the origin-destination matrices and travel-cost coefficient for congested networks in a stochastic user equilibrium. Transportation science 35, 107–123.

Yao, H., Tang, X., Wei, H., Zheng, G., Li, Z., 2019. Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction, in: Proceedings of the AAAI conference on artificial intelligence, pp. 5668–5675.

Yin, Y., Lawphongpanich, S., Lou, Y., 2008. Estimating investment requirement for maintaining and improving highway systems. Transportation Research Part C: Emerging Technologies 16, 199–211.

Zhang, C., Osorio, C., Flo¨ ttero¨ d, G., 2017. Efficient calibration techniques for large-scale traffic simulators. Transportation Research Part B: Methodological 97, 214–239.

10 Appendix: Proof of Theorem 2

To facilitate understanding, this section omits the superscript for a sample m and the dependence upon both context features x and neural network parameters θ. The Jacobian matrix of a vector-to-vector function $F(x): R^n \rightarrow R^m$ is denoted as $J_F(x) \in R^{m \times n}$.

We first give the formal definition of monotonicity and Lipschitz continuity of a vector-to-vector function $F(x)$ and equivalent conditions when the function is a self-mapping and differentiable everywhere on its domain. The equivalent conditions are more tractable and used in proving the monotonicity and Lipschitz continuity of the cost function.

Definition 7 (Monotonicity) *A function $F(x): R^n \rightarrow R^m$ is monotone if $\langle F(x) - F(y), x - y \rangle \geq 0$, $\forall x, y \in R^n$. A differentiable function $F(x): R^n \rightarrow R^n$ is monotone if and only if its Jacobian matrix $J_F(x) \in R^{n \times n}$ is positive-semidefinite.*

Definition 8 (Lipschitz Continuity) *A function is L-Lipschitz continuous if there exists, such that $\|F(x) - F(y)\| \leq L \|x - y\|$, $\forall x, y \in R^n$. A differentiable function $F(x): R^n \rightarrow R^n$ is L-Lipschitz continuous if and only if its Jacobian matrix $J_F(x) \in R^{n \times n}$ has finite spectral norms.*

To begin with, consider a special one-column scenario where Attribute Net has only one link block and the output of the link block has one column, i.e., $g^{\mathcal{A}}(v_a): R_+ \rightarrow R$. By assumption, is monotone and Lipschitz continuous with respect to v_a, i.e., $0 \leq \frac{dg^{\mathcal{A}}}{dv_a} \leq L$.

Let $G^{\mathcal{A}}(v): R^{|\mathbb{A}|} \rightarrow R^{|\mathbb{A}|}$ denote the mapping from link flows to link attributes, defined as $G^{\mathcal{A}}(v) = \{g^{\mathcal{A}}(v_a)\}_{a \in \mathcal{A}}$. Its Jacobian matrix,

$$J_{G^{\mathcal{A}}}(v) = Diag\left(\left\{\frac{dg^{\mathcal{A}}}{dv_a}\right\}_{a \in \mathcal{A}}\right),$$

is a diagonal matrix with nonnegative and finite elements. It is straightforward to show that $J_{G^{\mathcal{A}}}(v)$ is positive-semidefinite with finite spectral norm $|J_{G^{\mathcal{A}}}(v)| \leq \max_{a \in A}\left\{\frac{dg^{\mathcal{A}}}{dv_a}\right\} = L$

Recall that the attributes are the product of path-link incidence matrix Λ and link attributes. The Attribute Net is now defined as a self-mapping with respect to path flows, i.e., $G(h): R^{|\mathbb{P}|} \to R^{|\mathbb{P}|}$. It follows that the Jacobian matrix of $G(h) = \Lambda\, G^{\mathcal{A}}(\Lambda^{\top} h)$ is symmetric and positive-semidefinite. Path-link incidence matrix Λ is a 0-1 matrix with a bounded spectral norm. As per Cauchy–Schwarz inequality, the spectral norm $|J_G(h)| \leq L|\Lambda|^2$.

The cost function $\pi(h): R^{|\mathbb{P}|} \to R^{|\mathbb{P}|}$ is formulated as $\pi(h) = \Sigma\, w \odot G(h)$. Suppose the weights are positive the Jacobian matrix of the cost function, $J_\pi(h) = Diag(\Sigma\, w)\, J_G(h)$, is the product of two symmetric positive-semidefinite matrices and thus symmetric positive-semidefinite with spectral norm bounded by $|(\Sigma\, w)|\, |J_G(h)|$. It is equivalent to saying the cost function is monotone and Lipschitz continuous with respect to the path flows. This proof can be adapted to node block and path block by replacing the path-link incidence matrix Λ with the path-node incidence matrix Γ or an identity matrix.

Now we consider a general case. Let w_i denote the i-th column of weights and G_i denote the i-th column of attributes. The cost function is:

$$\pi(h) = \Sigma\; w \odot G(h)\, 1 = \sum_{i=1}^{|\mathcal{S}|} \Sigma\; w_i \odot G_i(h).$$

Suppose each block is column-wise monotone and Lipschitz continuity, is monotone and Lipschitz continuous following previous proof for one-column scenarios. Monotonicity and Lipschitz continuity are preserved under summation, it follows that the cost function is monotone and Lipschitz continuous concerning the path flows.

Additionally, it is straightforward to show that the Jacobian matrix of the cost function is the sum of symmetric matrix and thus is symmetric. Suppose $J_\pi(h)$ is real

everywhere, there exists a scalar function such that $\pi(h)$ is the gradient of a continuous function (Emberton, 2008). Under mild assumptions that the function is closed and proper, the monotonicity of $\pi(h)$ is equivalent to maximal monotonicity (Ryu and Yin, 2021). This completes the proof.