Transportation Research Part C 150 (2023) 104085

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

End-to-end learning of user equilibrium with implicit neural networks

Zhichen Liu [a], Yafeng Yin [a,*], Fan Bai [b], Donald K. Grimm [b]

[a] Department of Civil and Environmental Engineering, University of Michigan, Ann Arbor, MI, 48109, United States
[b] General Motors Research and Development, Warren, MI, 48092, United States

ARTICLE INFO

Keywords:
Network equilibrium
Neural-network-based variational inequality
End-to-end learning
Implicit layer

ABSTRACT

This paper intends to transform the transportation network equilibrium modeling paradigm via an "end-to-end" framework that directly learns travel choice preferences and the equilibrium state from multi-day link flow observations. The centerpiece of the proposed framework is to use deep neural networks to represent travelers' route choice preferences and then encapsulate the neural networks in a variational inequality that prescribes the user equilibrium flow distribution. The proposed neural network architecture ensures the existence of equilibrium and accommodates future changes in road network topology. The variational inequality is then embedded as an implicit layer in a learning framework, which takes the context features (e.g., road network and traveler characteristics) as input and outputs the user equilibrium flow distribution. By comparing computed equilibrium flows with observed flows, the neural networks can be trained. The proposed end-to-end framework is demonstrated and validated using synthesized data for the Sioux Falls network.

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).

This modeling paradigm has been established via a *"bottom-up"* 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 Wardropian 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.

A constant battle modelers have been fighting over the past half a century when developing a network equilibrium model is to strike a balance between behavioral realism and mathematical tractability. Informed by findings from travel behavior

* Corresponding author.
E-mail address: yafeng@umich.edu (Y. Yin).

https://doi.org/10.1016/j.trc.2023.104085
Received 23 August 2022; Received in revised form 25 February 2023; Accepted 27 February 2023
Available online 20 March 2023

research, modelers understand that travelers are boundedly rational (Simon, 1955), and their travel choice behaviors are much more complicated than what has been assumed in existing equilibrium models. However, when incorporating a better behavioral consideration as per a more advanced theory, e.g., the prospect theory (Tversky and Kahneman, 1992), the resulting model quickly becomes computationally intractable for large-scale networks (Xu et al., 2011). Therefore, despite numerous efforts for enhancing behavioral realism in network models, those with simplified behavioral rules, e.g., travelers choosing the fastest path, still dominate in the planning practice.

In this "bottom-up" approach, the selection of the behavior model is divorced from the end goal of the model building, i.e., prescribing an equilibrium flow distribution that matches observations as closely as possible. To construct a network equilibrium model, we would *pre-select* a behavior model, e.g., adopting multinomial logit for modeling route choice, calibrate the associated parameters with stated or revealed preference route-choice data and then proceed with the model building. This process is justified when the behavior model is perfect (or near so) for modeling route choice, which is certainly far from being true (Chen et al., 2016). The selection of the multinomial logit model reflects our belief or judgment rather than being the outcome of a calibration process against empirical flow data. Recall that a different behavior model yields different equilibrium conditions that would produce a different traffic flow distribution. Therefore, observed flows should play a role in selecting the behavior model. However, we have never done so, due to the lack of a selection methodology and empirical data. This model pre-selection bias is actually beyond the behavioral side, as the equilibrium model involves the supply-side components that also need to be learned.

In contrast to the traditional "bottom-up" approach for building a network equilibrium model, this study aims to transform the modeling paradigm via an *end-to-end* framework that directly learns travel choice preferences and the equilibrium state from data. More specifically, we consider a routing game where the context information such as route features and traveler characteristics is known and origin–destination (OD) demand observations and partial link flow data are available for an extended period of time. The centerpiece of the proposed end-to-end framework is to use deep neural networks to represent travelers' route choice preferences. The neural networks will then be embedded in a variational inequality (VI) that prescribes the user equilibrium flow distribution (see Fig. 1). We will treat this VI as an implicit layer in a learning framework, which takes the context features and demands as input and outputs the user equilibrium flow distribution. By comparing computed equilibrium flows with observed flows, the neural networks can be trained.

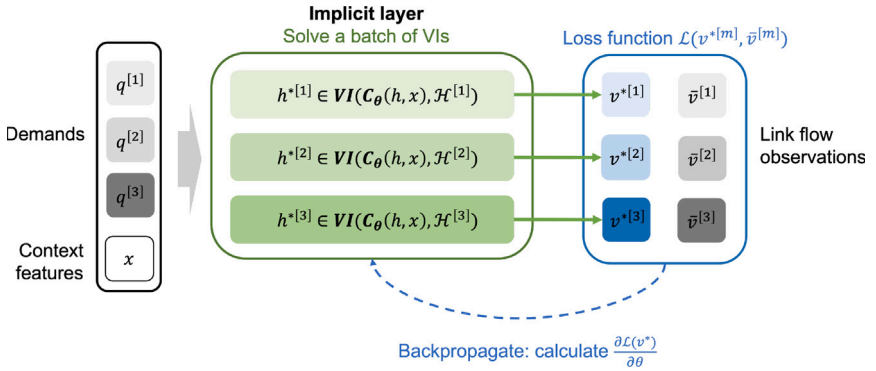

Fig. 1. An illustrative example of the proposed end-to-end learning framework, where demands and link flow observations are available over $\mathcal{M} = \{1, 2, 3\}$ days; h denotes path flows, and travelers' route choice preferences are represented by the cost function $C_\theta(h, x)$ parameterized by neural networks θ. The implicit layer takes the demands $(q^{[1]}, q^{[2]}, q^{[3]})$ and context x as input and outputs the equilibrium link flows $(v^{\star[1]}, v^{\star[2]}, v^{\star[3]})$, where superscript $[m]$ associates a variable with observation $m \in \mathcal{M}$.

One major advantage of the proposed end-to-end framework is that it does not pre-select a particular behavioral rule or theory to model travelers' choices. Instead, it represents travelers' route choice preferences with neural networks and directly learns them from observations. Deep neural networks can achieve a much richer representation of travelers' choice preferences with comparable interpretability as the classical discrete choice models (Sifringer et al., 2020). 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 (measured by a distance/loss function). 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.

To the best of our knowledge, our work is the first to integrate the learning of travel choice preferences into an end-to-end learning framework, with neural networks automatically discovering a good specification of route choice preferences from empirical data. This paper presents our first attempt to overcome the modeling and algorithmic challenges for enabling such an end-to-end learning. 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 to enable scalable solution algorithms for a batch of VI problems in forward propagation.

In backpropagation, implicit differentiation techniques enable the proposed framework to efficiently differentiate through the implicit layer.

The rest of this paper is organized as follows. Section 2 presents a summary of the relevant literature to better position this paper. Section 3 models the user equilibrium as a VI parameterized by neural networks and demonstrates the neural network architecture. Section 4 presents the end-to-end framework and elaborates on the forward and backward propagations. Numerical experiments are conducted on the Sioux Falls network to demonstrate the proposed framework in Section 5. Finally, Section 6 concludes the paper.

2. Relevant literature

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 Wardropian 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 paper 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 do not use empirical data to learn the equilibrium state or travel choice preferences. By contrast, the proposed framework directly learns travel choice preferences from data and captures equilibrium conditions with a neural-network-based 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. Neural-network-based discrete choice model

Recent studies have explored integrating *neural networks* with discrete choice models for more accurate prediction and more tractable parameter estimation. They show that neural networks, when carefully designed, are not "black-box" and can be interpreted for use in choice analysis. 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.

2.3. Learning-based transportation network modeling

Computational graphs along with automated differentiation provide powerful tools to calculate gradients in *end-to-end learning*, where the entire framework connecting the input features to the desired output is learned from data. In the field of network modeling, Wu et al. (2018) made an interesting attempt to encode trip generation, trip distribution, and path-based traffic loading with layered computational graphs and estimated hierarchical travel demands from multi-source data. Later studies extend the static framework to estimate multi-class dynamic OD demands (Ma et al., 2020). Apart from pre-selecting a behavior model before calibration, these studies ignore the flow-dependent congestion effect and only consider a one-step network loading encoded with standard or explicit neural networks.

By contrast, the proposed framework models the interactions among travelers as a routing game and captures the equilibrium conditions with a *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).

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. Our work advances these previous studies by integrating the selection or learning of behavior rules into an end-to-end framework, with neural networks automatically discovering a good specification of travel choice preferences from empirical data. 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.

3. Neural-network-based VI formulation of UE

3.1. Model formulation

We consider a case where all OD demands and partial or all link flows at peak periods are observable for a long period of time. This case is particularly relevant for a tolled freeway network for which the OD demand data are available from the electronic tolling collection system and link flow data are available from detectors, or a subway network where the OD demands are from smart card data and link flows are available from automatic passenger counting system. Suppose that a planning agency is interested in developing a static network equilibrium model to analyze the network for peak periods. The learning task is to approximate the cost function with neural networks and learn travelers' route choice preferences from multi-day observations.

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. OD demands and a subset of link flows are observed over a number of days (sample set) $\mathcal{M}$. Let $\overline{q}^{[m]} \in \overline{\mathcal{Q}}$ and $\overline{v}^{[m]} \in \overline{\mathcal{V}}$ be the OD demands and link flows observed on day (sample) $m \in \mathcal{M}$, where $\overline{\mathcal{Q}} = \{\overline{q} \in \mathbb{R}^{|\mathcal{R}|} : \overline{q} \geq 0\}$ and $\overline{\mathcal{V}} = \{\overline{v} \in \mathbb{R}^{|\mathcal{A}|} : \overline{v} \geq 0\}$ respectively.

Hereinafter, superscript $[m]$ associates a variable with a sample m. All vectors are assumed to be column vectors unless otherwise specified. 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. Appendix A summarizes the notations used throughout this paper.

We propose *Attribute Net* to learn the (path) *attributes* considered by travelers in their route choice decision process. As shown in Fig. 2, attributes $\pi^{[m]}$ depend on path flows $h^{[m]}$ and road network features $x^{\mathcal{G}}$. Attribute Net G_θ learns a continuous mapping from path flows $h^{[m]} \in \mathcal{H}^{[m]}$ and road network features $x^{\mathcal{G}} \in \mathcal{X}^{\mathcal{G}}$ to attributes $\pi^{[m]} \in \Pi$, defined as:

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

where θ is the learnable parameters; the feasible path flow set $\mathcal{H}^{[m]} = \{h \in \mathbb{R}^{|\mathcal{P}|} : h \geq 0, \Sigma^\top h = \overline{q}^{[m]}\}$ requires the feasible path flows $h^{[m]}$ to be nonnegative and satisfy flow conservation; $\Sigma \in \mathbb{R}^{|\mathcal{P}|\times|\mathcal{R}|}$ is the path-OD incidence matrix, in which Σ_{pr} equals one if path $p \in \mathcal{P}_r$ and zero otherwise; $\mathcal{X}^{\mathcal{G}}$ is the feasible set of road network features; $\Pi \subset \mathbb{R}^{|\mathcal{P}|\times|S|}$ is the feasible attribute set and $|S|$ is the number of attributes considered by travelers.

Weight Net is proposed to capture traveler heterogeneities and learn the OD-specific preferences over learned attributes (see Fig. 2). 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 F_θ learns a mapping from traveler characteristics $x^{\mathcal{R}} \in \mathcal{X}^{\mathcal{R}}$ to OD-specific weights $w \in \mathcal{W}$, defined as:

$$F_\theta : \mathcal{X}^{\mathcal{R}} \to \mathcal{W},$$

where $\mathcal{X}^{\mathcal{R}} \subset \mathbb{R}^{|\mathcal{R}|\times|\mathcal{I}^{\mathcal{R}}|}$ is the feasible set of traveler characteristics; $|\mathcal{I}^{\mathcal{R}}|$ is the number of characteristics attached to each OD; $\mathcal{W} \subset \mathbb{R}^{|\mathcal{R}|\times|S|}$ is the feasible weight set. The parameters of Weight Net and Attribute Net are learned simultaneously and thus are both represented as θ.

Subsequently, we assume that travelers choose routes to minimize their perceived path costs $c^{[m]} \in C \subset \mathbb{R}^{|\mathcal{P}|}$, which are represented as a weighted sum of attributes:

$$c^{[m]} = \Sigma\, w \odot \pi^{[m]}\, \mathbb{1}, \tag{1}$$

where $\odot$ represents the Hadamard (element-wise) product and $\mathbb{1} \in \mathbb{R}^{|S|}$ is a column vector of ones to calculate the sum over the rows. Equivalently, let context features $x \in \mathcal{X}$ include traveler characteristics $x^{\mathcal{R}}$ and road network features $x^{\mathcal{G}}$. The cost function $C_\theta : \mathcal{H}^{[m]} \times \mathcal{X} \to C$ maps path flows and context features to path costs, defined as:

$$C_\theta(h^{[m]}, x) = \Sigma\, F_\theta(x^{\mathcal{R}}) \odot G_\theta(h^{[m]}, x^{\mathcal{G}})\, \mathbb{1}, \tag{2}$$

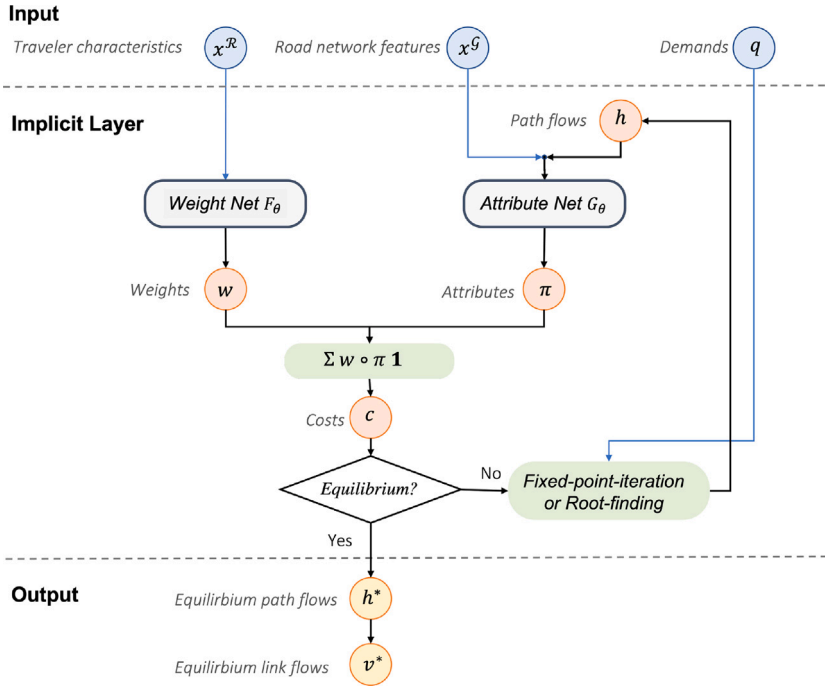

Fig. 2. Illustration of the end-to-end learning framework. The superscripts for a sample m are omitted.

is approximated by neural networks.

The *multi-class User Equilibrium (UE) with inelastic demand* for sample m is formulated as a parameterized VI in Eq. (3), the solution to which is the equilibrium flow $h^{\star[m]}$, i.e., $h^{\star[m]} \in \mathbb{VI}(C_\theta(h, x), \mathcal{H}^{[m]})$.

$$\left\langle C_\theta(h^{\star[m]}, x), h^{[m]} - h^{\star[m]} \right\rangle \geq 0, \quad \forall h^{[m]} \in \mathcal{H}^{[m]}. \tag{3}$$

The VI problem in Eq. (3) requires the knowledge of feasible path set $\mathcal{P}$. This is a common assumption for path-based UE formulation (Guarda and Qian, 2022; Bekhor and Toledo, 2005) with well-developed methods for generating the feasible path set in the literature (Frejinger et al., 2009). If one believes that path costs are link additive, the following link-based UE formulation can be used instead:

$$\left\langle T_\theta(v^{\star[m]}, x), v^{[m]} - v^{\star[m]} \right\rangle \geq 0, \quad \forall v^{[m]} \in \mathcal{V}^{[m]}, \tag{4}$$

where $v^{[m]} \in \mathcal{V}^{[m]}$ and $T_\theta(v^{\star[m]}, x)$ are link flows and link cost functions respectively; $\mathcal{V}^{[m]} = \{v \in \mathbb{R}^{|\mathcal{A}|} : \ v \geq 0, \Sigma^\top \Lambda v = \bar{q}^{[m]}\}$ is the set of feasible link flows and $\Lambda \in \mathbb{R}^{|\mathcal{P}|\times|\mathcal{A}|}$ denotes the path-link incidence matrix, Λ_{pa} equals one if link $a \in \mathcal{A}$ belongs to path $p \in \mathcal{P}$, and zero otherwise. In this paper, we consider that path cost may not be link additive and thus use the path-based formulation.

Theorem 1 (*Existence of Equilibrium*). *There exists at least one solution to the multi-class user equilibrium problem in Eq.* (3).

Proof. Path flow $h^{[m]}$ is a solution to $\mathbb{VI}\left(C_\theta(h, x), \mathcal{H}^{[m]}\right)$ if and only if it is the fixed point of the projection operator $\mathbb{P}_{\mathcal{H}^{[m]}}(\cdot)$ for any $\alpha > 0$, defined as:

$$h^{\star[m]} \in \mathbb{VI}\left(C_\theta(h, x), \mathcal{H}^{[m]}\right) \Longleftrightarrow h^{\star[m]} = \mathbb{P}_{\mathcal{H}^{[m]}}\left(h^{\star[m]} - \alpha\, C_\theta(h^{\star[m]}, x)\right), \tag{5}$$

where the projection operator is defined as $\mathbb{P}_{\mathcal{H}^{[m]}}(y) = \text{argmin}_{y' \in \mathcal{H}^{[m]}} \|y' - y\|$.

The cost function $C_\theta(h^{[m]}, x)$ is approximated by neural networks and thus is continuous. The fixed point operator $\mathbb{P}_{\mathcal{H}^{[m]}}(\cdot)$ is a projection operator that is continuous. Because the feasible flow set $\mathcal{H}^{[m]}$ is convex and compact, as per Brouwer's fixed point theorem, there exists at least one solution to the fixed point problem in Eq. (5).

3.2. 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 desired properties to enable efficient training. Hereinafter, we highlight that features/attributes are the concentation of single features/attributes for all elements within one set. For example, the link feature is $x^{\mathcal{A}} = \left\{x_1^{\mathcal{A}}, \ldots, x_a^{\mathcal{A}}, \ldots, x_{|\mathcal{A}|}^{\mathcal{A}}\right\} = \{x_a^{\mathcal{A}}\}_{a\in\mathcal{A}}$.

3.2.1. Attribute net

One may construct the Attribut Net G_θ 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 Fig. 3, 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.

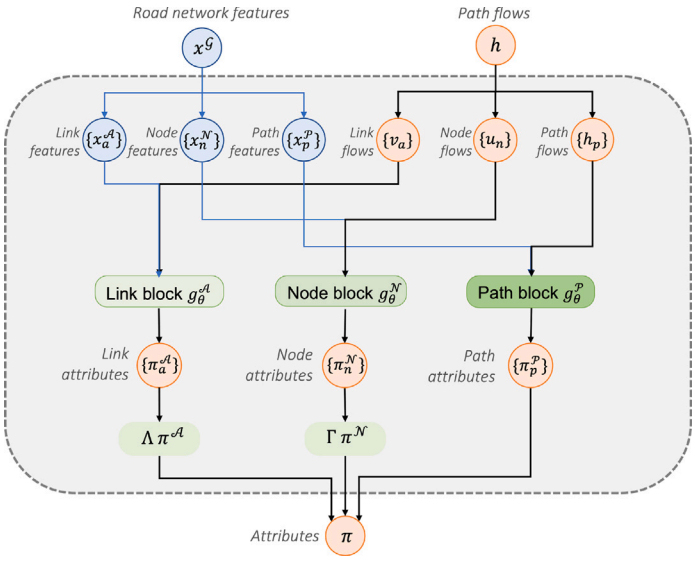

Fig. 3. Illustration of Attribute Net. The feature/attribute subscripts for enumerating the elements within a set and the superscripts for a sample m are omitted to facilitate presentation.

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* $g_\theta^{\mathcal{A}}$ takes the single link flow $v_a \in \mathbb{R}_+$ and single link features $x_a^{\mathcal{A}} \in \mathcal{X}_a^{\mathcal{A}}$ of one link $a \in \mathcal{A}$ as input and outputs the corresponding link attributes $\pi_a^{\mathcal{A}} \in \Pi_a^{\mathcal{A}}$, defined as:

$$g_\theta^{\mathcal{A}} : \mathbb{R}_+ \times \mathcal{X}_a^{\mathcal{A}} \to \Pi_a^{\mathcal{A}},$$

where $\mathcal{X}_a^{\mathcal{A}} \subset \mathbb{R}^{|\mathcal{I}^{\mathcal{A}}|}$ is the feasible set of single link features; $|\mathcal{I}^{\mathcal{A}}|$ is the number of features associated with one link; $\Pi_a^{\mathcal{A}} \subset \mathbb{R}^{|\mathcal{S}^{\mathcal{A}}|}$ 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* $\pi^{\mathcal{A}} \in \mathbb{R}^{|\mathcal{A}|\times|\mathcal{S}^{\mathcal{A}}|}$ are the concatenation of single link attributes, defined as:

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

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). As shown in Fig. 4, 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 $(1 + |\mathcal{I}^{\mathcal{A}}| = 3)$ and the output dimension $(|\mathcal{S}^{\mathcal{A}}| = 1)$ 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 flow h (denoted as the slash box in Fig. 4) whereas the input and output dimensions of the link block remain the same.

Similarly, let node flow u_n be the sum of link flows from all approaches at node $n \in \mathcal{N}$. To capture the interactions among link flows, node block $g_\theta^{\mathcal{N}} : \mathbb{R}_+ \times \mathcal{X}_n^{\mathcal{N}} \to \Pi_n^{\mathcal{N}}$ maps the single node flow $u_n \in \mathbb{R}_+$ and single node features $x_n^{\mathcal{N}} \in \mathcal{X}_n^{\mathcal{N}} \subset \mathbb{R}^{|\mathcal{I}^{\mathcal{N}}|}$ of one node $n \in \mathcal{N}$ to its local node attributes $\pi_n^{\mathcal{N}} \in \Pi_n^{\mathcal{N}} \subset \mathbb{R}^{|\mathcal{S}^{\mathcal{N}}|}$. The *node attributes* $\pi^{\mathcal{N}} \in \mathbb{R}^{|\mathcal{N}|\times|\mathcal{S}^{\mathcal{N}}|}$ are the concatenation of single node

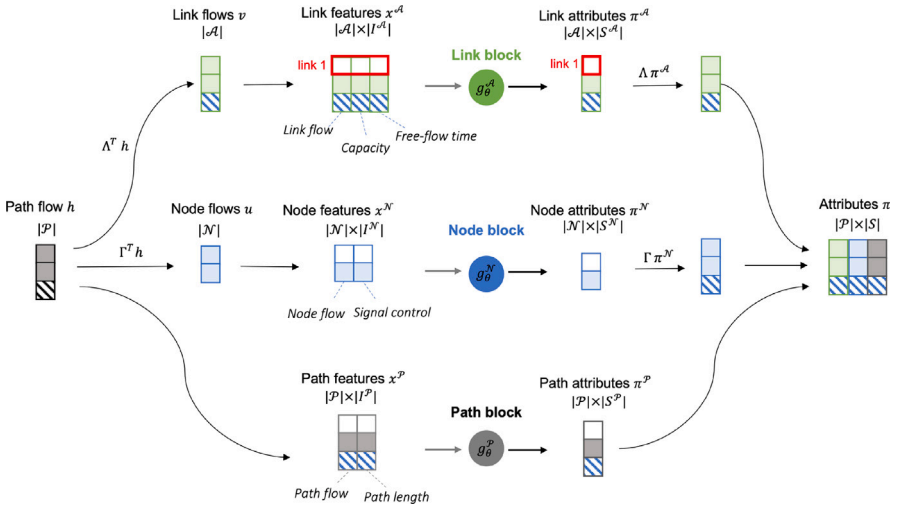

Fig. 4. Illustrations of link, node, and path blocks. Slash boxes denote the change in variables when another parallel link is added to the road network.

attributes, defined as:

$$\pi^{\mathcal{N}} = \left\{ g_\theta^{\mathcal{N}}(u_n, x_n^{\mathcal{N}})^\top \right\}_{n\in\mathcal{N}}.$$

And the path block $g_\theta^{\mathcal{P}} : \mathbb{R}_+ \times \mathcal{X}_p^{\mathcal{P}} \to \Pi_p^{\mathcal{P}}$ maps the single path flows $h_p \in \mathbb{R}_+$ and single path features $x_p^{\mathcal{P}} \in \mathcal{X}_p^{\mathcal{P}} \subset \mathbb{R}^{|I^{\mathcal{P}}|}$ of one path $p \in \mathcal{P}$ to its path attributes $\pi_p^{\mathcal{P}} \in \Pi_p^{\mathcal{P}} \subset \mathbb{R}^{|S^{\mathcal{P}}|}$. The *path attributes* are:

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

Finally, the attributes π are the concatenation of link attributes $\pi^{\mathcal{A}}$, node attributes $\pi^{\mathcal{N}}$ and path attributes $\pi^{\mathcal{P}}$, defined as:

$$\pi = \left\{ \Lambda\pi^{\mathcal{A}},\ \Gamma\pi^{\mathcal{N}},\ \pi^{\mathcal{P}} \right\}, \tag{6}$$

where $\Gamma \in \mathbb{R}^{|\mathcal{P}|\times|\mathcal{N}|}$ is the path-node incidence matrix.

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.

3.2.2. Weight net

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 f_θ that maps the single traveler characteristics $x_r^{\mathcal{R}} \in \mathcal{X}_r^{\mathcal{R}} \subset \mathbb{R}^{|I^{\mathcal{R}}|}$ of one OD pair to its OD-specific weights $w_r \in \mathcal{W}_r \subset \mathbb{R}^{|S|}$, defined as:

$$f_\theta(x_r^{\mathcal{R}}) : \mathcal{X}_r^{\mathcal{R}} \to \mathcal{W}_r,$$

where $|I^{\mathcal{R}}|$ is the number of traveler characteristics. The weights $w \in \mathcal{W}$ is represented as:

$$w = F_\theta(x^{\mathcal{R}}) = \left\{ f_\theta(x_r^{\mathcal{R}})^\top \right\}_{r\in\mathcal{R}}.$$

The parameters of neural network f_θ 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 2.2, can be incorporated into the proposed framework and guide the design of neural network architectures, particularly when behavior interpretability is desired.

3.2.3. Regularization of cost function

As shown in Theorem 1, the continuity of cost function $C_\theta(h, x)$ 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 (see Section 4).

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

Theorem 2 (*Monotonicity and Lipschitz Continuity of Cost Function*). *The cost function $C_\theta(h, x)$ defined in Eq. (2) is maximal monotone and Lipschitz continuous with respect to path flows h if weight w is positive and link block $g_\theta^{\mathcal{A}}$, node block $g_\theta^{\mathcal{N}}$ and path block $g_\theta^{\mathcal{P}}$ are column-wise monotone and Lipschitz continuous.*

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 $y^{(l)} = \sigma^{(l)}(W^{(l)}y^{(l-1)} + b^{(l)})$, where $W^{(l)}$ and $b^{(l)}$ are learnable parameters of linear layers. We constrain the sign of weights w as strict positive by using SoftPlus[1] as the last layer of Weight Net F_θ. The column-wise monotonicity and Lipschitz continuity of attribute blocks, however, are more challenging to obtain. Most activation 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.

4. End-to-end learning with an implicit layer

We are now ready to discuss how to learn the parameters θ associated with the cost function $C_\theta(h, x)$ in the VI problem defined in Eq. (3). To train the neural networks, the VI is encapsulated as an implicit layer in a learning framework, which takes the context features x and demands (demand $\overline{q}^{[m]}$ encapsulated in feasible set $\mathcal{H}^{[m]}$) as input and outputs user equilibrium flows. Then the neural networks are trained by comparing the computed equilibrium flows with observed flows. Consider a smooth loss function $\mathcal{L} : \mathcal{V} \times \overline{\mathcal{V}} \to \mathbb{R}$ that measures the distance between predicted equilibrium flows and real-world observations, the end-to-end learning problem can be formulated as the following Mathematical Program with Equilibrium Constraints (MPEC).

$$\begin{aligned} \min_{\theta} \quad & \mathbb{E}_{\overline{q}\sim Q}\left[\mathcal{L}(v^{\star[m]}, \overline{v}^{[m]})\right] \\ s.t. \quad & h^{\star[m]} \in \mathbb{VI}\left(C_\theta(h, x), \mathcal{H}^{[m]}\right), \quad \forall m \in \mathcal{M} \\ & v^{\star[m]} = \Lambda^\top h^{\star[m]}, \quad \forall m \in \mathcal{M} \end{aligned} \tag{7}$$

where the demand $\overline{q}$ is assumed to follow a distribution Q; $h^{\star[m]}$ and $v^{\star[m]}$ are the equilibrium path and link flows on sample m respectively. This MPEC is nonconvex in general (Lawphongpanich and Hearn, 2004).

Remark 1. If the cost function is independent of input feature x and equals the sum of link travel times and an entropy term (i.e., $c(h) = \Lambda t(h) + \ln(h)$), the learning problem defined in Eq. (7) 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 remainder of the section deals 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.

4.1. Forward: solving a batch of constrained VIs

Batched operation is essential for training neural networks with massive empirical data. Instead of solving one constrained VI problem, the forward propagation in the proposed framework requires simultaneously solving a batch of VIs. 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 efficiently solve a batch of constrained VIs, we adopt the recent developments in operator-splitting methods and reformulate the original VI problem as an auxiliary fixed point problem, which can be iteratively solved with closed-form update rules. More specifically, we decompose the polyhedron constraint set $\mathcal{H}^{[m]}$ into two simpler sets $\mathcal{H}_1 = \{h \in \mathbb{R}^{|\mathcal{P}|} : h \geq 0\}$ and $\mathcal{H}_2^{[m]} = \{h \in \mathbb{R}^{|\mathcal{P}|} : \Sigma^\top h = \overline{q}^{[m]}\}$. Projection operators onto $\mathcal{H}_1$ and $\mathcal{H}_2^{[m]}$ have closed-form solutions[2,3] that can be encoded with computational graphs and efficiently implemented in a batched manner (Heaton et al., 2021).

[1] $\sigma(y^{(l)}) = \log(1 + \exp y^{(l)})$

[2] $\mathbb{P}_{\mathcal{H}_1}(z^{[m]}) = [z^{[m]}]_+$

[3] $\mathbb{P}_{\mathcal{H}_2^{[m]}}(z^{[m]}) = z^{[m]} - UE^{-1}V^\top(\Sigma z^{[m]} - \overline{q}^{[m]})$, where $\Sigma = UEV$ is the compact singular value decomposition of Σ such that U and V have orthonormal columns and E is invertible.

We consider an auxiliary variable $z^{[m]} \in \mathcal{Z} \subset \mathbb{R}^{|\mathcal{P}|}$ and auxiliary fixed point operator $\mathbb{T}_\theta(z^{[m]}, x)$ for any $\alpha > 0$, defined as:

$$\mathbb{T}_\theta(z^{[m]}, x) \triangleq z^{[m]} - \mathbb{P}_{\mathcal{H}_1}(z^{[m]}) + \mathbb{P}_{\mathcal{H}_2^{[m]}}\left(2\mathbb{P}_{\mathcal{H}_1}(z^{[m]}) - z^{[m]} - \alpha\, C_\theta(\mathbb{P}_{\mathcal{H}_1}(z^{[m]}), x)\right).$$

The VI problem in Eq. (3) can be equivalently formulated as an *auxiliary fixed point problem*:

$$z^{\star[m]} = \mathbb{T}_\theta(z^{\star[m]}, x),$$

and the equilibrium path flows are calculated as $h^{\star[m]} = \mathbb{P}_{\mathcal{H}_1}(z^{\star[m]})$. It is equivalent to say:

$$h^{\star[m]} \in \mathbb{VI}\left(C_\theta(h, x), \mathcal{H}^{[m]}\right) \iff z^{\star[m]} = \mathbb{T}_\theta(z^{\star[m]}, x), \quad h^{\star[m]} = \mathbb{P}_{\mathcal{H}_1}(z^{\star[m]}). \tag{8}$$

The equivalence can be established when the cost function $C_\theta(h, x)$ is maximal monotone (Heaton et al., 2021), which holds, as we have proved in Theorem 2, if the weight w is positive and all three blocks are column-wise monotone. Appendix C presents a brief proof of the equivalence between the VI problem and the auxiliary fixed point problems in Eq. (8). Interested readers can refer to Ryu and Yin (2021) and Heaton et al. (2021) for more details.

We explore two iterative methods, fixed point iteration and root-finding, to solve the auxiliary fixed point problem. Two criteria are considered to measure the convergence at iteration k. To facilitate understanding, the superscripts for a sample m are omitted for the rest of this section.

Definition 1 (*Residual and Relative Residual*)**.** The residual ϕ measures the absolute change of the auxiliary variable between two consecutive iterations, defined as:

$$\phi^{(k)} \triangleq \|z^{(k+1)} - z^{(k)}\|.$$

The relative residual φ measures the relative absolute change of the auxiliary variable between two consecutive iterations, defined as:

$$\varphi^{(k)} \triangleq \frac{\|z^{(k+1)} - z^{(k)}\|}{\|z^{(k)}\|}.$$

4.1.1. Fixed point iteration

Starting with an initial point $\tilde{z}$, the fixed point iteration repeats $z^{(k+1)} = \mathbb{T}_\theta(z^{(k)}, x)$ for each step until the relative residual is smaller than a threshold ε_1 or the iteration step k exceeds the maximum number of iterations κ_1. The algorithm heuristically adjusts the step size α_1 when the current step size fails to reduce the relative residual. Algorithm 1 shows the algorithm details. The initial point $\tilde{z}$ is not necessarily feasible and will be projected onto the feasible region during the iteration. $z^{(k+1/2)}$ denotes auxiliary path flows that only satisfy nonnegativity constraints. The threshold and rule to heuristically adjust step sizes are represented as β_1 and γ_1 respectively.

Algorithm 1 Fixed point iteration

Input: x, $\bar{q}$, $\tilde{z}$
Output: $h^\star$

$z^{(1)} \leftarrow \tilde{z}$, $z^{(0)} \leftarrow \tilde{z}$, $k \leftarrow 1$, $\varphi^{(0)} = \varepsilon_1 + 1$ ▷ Initialization
while $\varphi^{(k)} > \varepsilon_1$ and $k < \kappa_1$ **do**
  $z^{(k+1/2)} \leftarrow \mathbb{P}_{\mathcal{H}_1}(z^{(k)})$ ▷ Update auxiliary path flow
  $y^{(k+1)} \leftarrow \mathbb{P}_{\mathcal{H}_2}\left(2z^{(k+1/2)} - z^{(k)} - \alpha_1^{(k)} C_\theta(z^{(k+1/2)}, x)\right)$ ▷ Projection
  $z^{(k+1)} \leftarrow z^{(k)} - z^{(k+1/2)} + y^{(k+1)}$ ▷ Update auxiliary variable
  $\varphi^{(k+1)} \leftarrow \varphi(z^{(k)}, z^{(k+1)})$ ▷ Update relative residual
  if $\varphi^{(k+1)}/\varphi^{(k)} \geq \beta_1$ **then** ▷ If fails to decrease the relative residual
    $\alpha_1^{(k+1)} \leftarrow \gamma_1 \alpha_1^{(k)}$ ▷ Adjust step size
  end if
  $k \leftarrow k + 1$ ▷ Update iteration step
end while
$h^\star \leftarrow \mathbb{P}_{\mathcal{H}_1}(z^{(k)})$ ▷ Output equilibrium path flow

Proposition 1. *Supposing that the cost function $C_\theta(h, x)$ is L-Lipschitz continuous and step size $\alpha_1 \in (0, 2/L)$, the fixed point iteration $z^{(k+1)} = \mathbb{T}_\theta(z^{(k)}, x)$ converges linearly to the fixed point if one exists.*

The proof of Proposition 1 is shown in Appendix D. The selection of step size is vital. If the step size is too large, the fixed point iteration may diverge; if too small, the convergence can be extremely slow. The optimal step size depends on an unknown Lipschitz constant L, the exact computation of which is NP-hard (Virmaux and Scaman, 2018). We thus explore two variants of fixed point 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 $z^{(k+1)}$ as an optimal linear combination of τ previous iterations, i.e., $z^{(k+1)} = \sum_{i=1}^{\tau} \alpha^{(i)} \mathbb{T}_\theta\left(z^{(k-i+1)}, x\right)$. The optimal step size $\alpha^{(i)}$ solves a quadratic program:

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

where the objective function is to minimize the sum of residual norms over τ iterations.

Another variant, *weighted ergodic iteration*, heuristically adjusts the step size at each iteration and updates $z^{(k)}$ as a linear combination of previous steps:

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

Weighted ergodic iteration improves the convergence rate of fixed point iteration from $O\left(\frac{1}{\sqrt{k+1}}\right)$ to $O\left(\frac{1}{k+1}\right)$ when the Jacobian matrix of $C_\theta(h, x)$ is symmetric (Davis and Yin, 2017), which holds, as shown in Appendix B, if the Attribute Net is constructed following Eq. (6).

4.1.2. Root-finding

Solving for the auxiliary fixed point is equivalent to finding the root of $z^\star - \mathbb{T}_\theta(z^\star, x) = 0$ via a root-finding method. The projection operator $\mathbb{P}_{\mathcal{H}_1}(\cdot)$ is non-differentiable at the boundary of a set and thus Newton's method may diverge. Therefore, we use Broyden's method, a quasi-Newton's method that does not require differentiability. Broyden's method approximates Newton's direction and updates the point as $z^{(k+1)} = z^{(k)} - s^{(k)}$. Let the initial guess be $s^{(0)} = -I$ and the direction is updated as:

$$s^{(k+1)} = s^{(k)} + \frac{\Delta z^{(k+1)} - s^{(k)} \Delta\phi^{(k+1)}}{\Delta z^{(k+1)\top} s^{(k)} \Delta\phi^{(k+1)}} \Delta z^{(k+1)\top} s^{(k)}, \tag{9}$$

where $\Delta z^{(k+1)} = z^{(k+1)} - z^{(k)}$ and $\Delta\phi^{(k+1)} = \phi^{(k+1)} - \phi^{(k)}$. The details of the root-finding method are shown in Algorithm 2, where the step size α_2 is heuristically adjusted when the current step size fails to reduce the relative residual.

Algorithm 2 Root-finding

Input: x, $\bar{q}$, $\tilde{z}$
Output: h

$z^{(1)} \leftarrow \tilde{z}$, $z^{(0)} \leftarrow \tilde{z}$, $k \leftarrow 1$, $s^{(0)} \leftarrow -I$, $\varphi^{(0)} = \varepsilon_2 + 1$ ▷ Initialization
while $\varphi^{(k)} > \varepsilon_2$ and $k < \kappa_2$ **do**
  $s^{(k)} \leftarrow s^{(k-1)} + \frac{\Delta z^{(k)} - s^{(k)} \Delta\phi^{(k)}}{\Delta z^{(k)\top} s^{(k-1)} \Delta\phi^{(k)}} \Delta z^{(k)\top} s^{(k-1)}$ ▷ Update direction
  $z^{(k+1)} \leftarrow z^{(k)} - \alpha_2^{(k)} s^{(k)}$ ▷ Update auxiliary variable
  $\varphi^{(k+1)} \leftarrow \varphi(z^{(k)}, z^{(k+1)})$. ▷ Update relative residual
  if $\varphi^{(k+1)}/\varphi^{(k)} \geq \beta_2$ **then** ▷ If fails to decrease the relative residual
    $\alpha_2^{(k+1)} \leftarrow \gamma_2 \alpha_2^{(k)}$ ▷ Adjust step size
  end if
  $k \leftarrow k + 1$ ▷ Update iteration step
end while
$h^\star = \mathbb{P}_{\mathcal{H}_1}(z^{(k)})$ ▷ Output equilibrium path flow

4.2. Backward: differentiate through the fixed point solution

In this section, we consider a multivariate function $f(x, y) : \mathbb{R}^p \times \mathbb{R}^m \to \mathbb{R}^n$ and denote the differentiation with respect to one argument x, as $\frac{\partial f}{\partial x} \triangleq \partial g(x)$where $g(x) = f(x, y)$. Note that $\frac{\partial f}{\partial x} \in \mathbb{R}^{n \times p}$ is a matrix consistent with the input and output dimensions. For backpropagation, we need to efficiently differentiate through the fixed point solution $z^\star$ and adjust the parameter θ with $\frac{\partial \mathcal{L}}{\partial \theta}$:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial z^\star} \frac{\partial z^\star}{\partial \theta}.$$

Solving the auxiliary fixed point problem usually entails many iterations and thus explicitly backpropagating through each iteration to calculate $\partial z^\star / \partial\theta$ can be computationally expensive and prone to vanishing or exploding gradients. Using the implicit function theorem, $z^\star$ is a continuous function of θ near the fixed point and $\partial z^\star / \partial\theta$ can be expressed as follows (Bai et al., 2019):

$$\frac{\partial z^\star}{\partial \theta} = \left(I - \frac{\partial \mathbb{T}_\theta(z^\star, x)}{\partial z^\star}\right)^{-1} \frac{\partial \mathbb{T}_\theta(z^\star, x)}{\partial \theta}, \tag{10}$$

where $\partial \mathbb{T}_\theta(z^\star, x)/\partial z^\star$ and $\partial \mathbb{T}_\theta(z^\star, x)/\partial\theta$ can be computed with automated differentiation.

We further reduce the computational difficulty by approximating the matrix inversion in Eq. (10). Three methods are explored. First, the *Jacobian-free backpropagation* replaces $\left(I - \frac{\partial \mathbb{T}_\theta(z^\star, x)}{\partial z^\star}\right)^{-1}$ 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*,[4] reducing the computational cost from matrix inversion to matrix–matrix multiplications. Third, the gradient of our interest can be reformulated and solved with another fixed point iteration, on which we elaborate in Appendix E. Interested readers can refer to Bai et al. (2019) for more details.

Remark 2. Calculating the gradients of equilibrium flows $h^\star$ 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 $h^\star$. These conditions may not hold in a general setting. However, the aforementioned numerical methods work well in our numerical experiments.

5. Numerical experiments

In this section, we conduct a proof-of-concept of the proposed framework in the Sioux Falls network with $|\mathcal{A}| = 76$ links, $|\mathcal{N}| = 28$ nodes, and $|\mathcal{R}| = 528$ OD pairs. The prediction accuracy and robustness of the proposed framework are tested in different scenarios.

5.1. Experiment setting

5.1.1. Training set generation

Each OD pair $r \in \mathcal{R}$ 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 $a \in \mathcal{A}$ follows the BPR function, i.e., $t_a(v_a) = \bar{t}_a \left(1 + 0.15 \left(v_a/\bar{c}_a\right)^4\right)$, where $\bar{t}_a$ and $\bar{c}_a$ represent free-flow time and link capacity respectively. The node delay on node $n \in \mathcal{N}$ follows an exponential form as proposed by Jeihani et al. (2006), i.e., $d_n(u_n) = \bar{d}_n \left(u_n/\bar{c}_n\right)^{\beta_n} + \gamma_n$, where $\bar{d}_n$, $\bar{c}_n$, β_n, and γ_n are parameters depending on intersection layout. 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 e_p on bad-condition links. Let $0 \le x_p \le 1$ 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 good-condition links.

The "ground-truth" cost for travelers of OD pair r to use path $p \in \mathcal{P}_r$ is a weighted sum of link travel times, node delays, and a discomfort constant:

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

where the class-specific weights for the node delays and the discomfort constant are $w_r^d = (5.5 + x_r^1) \cdot \exp(1 - x_r^2)$ and $w_r^e = 10^{(x_r^1 \cdot x_r^2)}$, respectively. 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 $\mathcal{P}_r$ 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 and one training sample is $\left((x, \bar{q}^{[m]}), \bar{v}^{[m]}\right)$. The training and test sets include 1536 and 512 samples respectively. So far all links are assumed to be observable.

5.1.2. Neural network architecture

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: $|\mathcal{I}^{\mathcal{N}}| = 4$, $|\mathcal{I}^{\mathcal{P}}| = 1$, $|\mathcal{I}^{\mathcal{R}}| = 2$, $|\mathcal{S}^{\mathcal{N}}| = |\mathcal{S}^{\mathcal{P}}| = 1$, and $|\mathcal{S}| = 3$. Weighted ergodic iteration and fixed point approximation 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

[4] $\mathrm{A}^{-1} \approx \sum_{i=0}^{r-1} (\mathrm{I} - \mathrm{A})^i$

Table 1
Forward algorithm hyperparameters.

Name	κ_1	α_1	β_1	γ_1	κ_2	α_2	β_2	γ_2
F	1e3	1e−3	1	0.9	–	–	–	–
FA	1e3	1e−3	1	0.9	–	–	–	–
FW	1e3	1e−3	1	0.9	–	–	–	–
R	–	–	–	–	1e2	1e−3	1	0.9
F-R	1e3	1e−3	1	0.9	1e2	1e−3	1	0.9
FA-R	10	1e−3	1	0.9	10	1e−3	1	0.9
FW-R	1e3	1e−3	1	0.9	1e2	1e−3	1	0.9

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 fixed point 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 fixed point iterations initially and switch to the root-finding when the relative residential is sufficiently small. Details of algorithm hyperparameters are shown in Table 1. The stopping thresholds are $\varepsilon_1 = \varepsilon_2 = 1e{-}5$.

5.1.3. Performance measurement

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 $\mathcal{G}$ and tested on the same road network $\mathcal{G}$. By contrast, in *out-of-distribution* tests, the trained model is tested on a partially changed road network $\mathcal{G}'$. 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.

Hereinafter, Mean Absolute Percentage Error (MAPE) is used to measure the percentage differences in link flow predictions. MAPE of one sample is:

$$\eta \triangleq \frac{1}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} \frac{|v_a^\star - \overline{v}_a|}{\overline{v}_a} \times 100\%.$$

5.2. Performance comparisons

Table 2 compares the MAPE of different network equilibrium models. The proposed end-to-end learning framework is denoted as "Implicit" in Table 2. We use DUE as the baseline and denote its MAPE as η_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.

Fig. 5 shows the training processes of different forward algorithms. R converges the fastest within nine epochs whereas FW converges the slowest after 27 epochs. Combining F or FA with R slows down the convergence and hurts the training performance.

As shown in Table 3, 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 it consistently shrinks the step size during iterations.

Fig. 6 compares the performance of three backpropagation methods: Jacobian-Free (JF) approximation, Newman Approximation (NA), and Fixed point Approximation (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 Fig. 7. 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.

Table 2
MAPE of different network equilibrium models. MAPEs are shown in percentage.

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%)

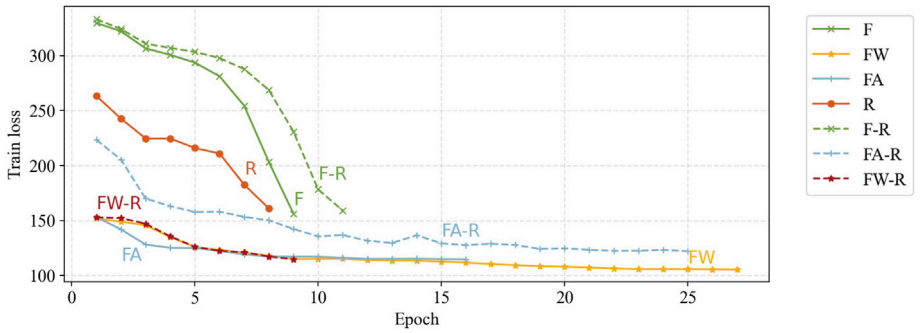

Fig. 5. Training process of different forward algorithms.

Table 3
MAPE of proposed forward algorithms. MAPEs are shown in percentage and superscript.

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								
Demand	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

*Denotes the best performance of each scenario.

We conclude this section by sharing some tips on training the proposed framework. The weighted ergodic iteration and fixed point approximations are recommended as default forward and backward methods. The step size turns out to be the most important

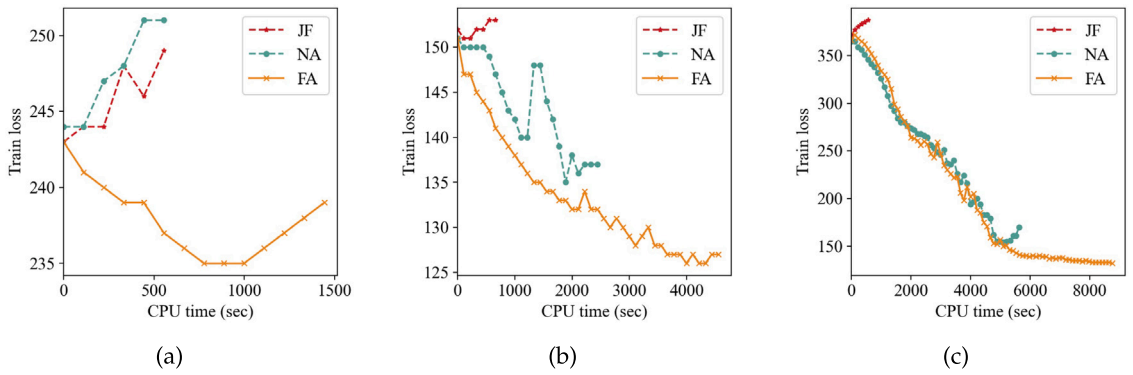

Fig. 6. Performances of different backpropagation methods under (a) base, (b) uncongested, and (c) congested demand.

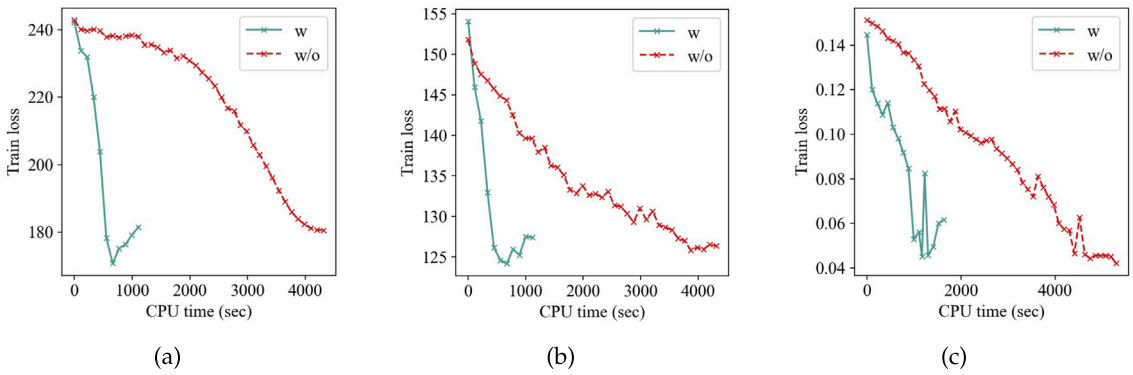

Fig. 7. Effects of spectral normalization under (a) base, (b) uncongested, and (c) congested demand. "w" suggests "with spectral normalization" and "w/o" suggests "without spectral normalization".

hyperparameter and one should always start by fine-tuning it. Spectral normalization is recommended to constrain the Lipshitz constant and speed up the convergence.

5.3. Robustness analysis

In this section, we examine the robustness of the proposed framework by relaxing model assumptions. FW, R, and FW-R have the best performance among fixed point iterations, root-finding, and combined methods 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.

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 (see Fig. 8). For example, Fig. 8(b) 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-truth, are added to all demands. As shown in Fig. 9, 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.

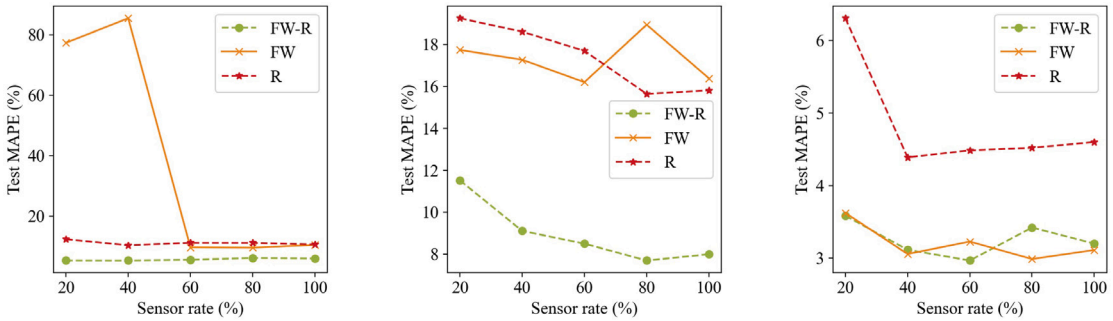

Fig. 8. Model performances with different sensor coverage rates under (a) base, (b) uncongested, and (c) congested demand.

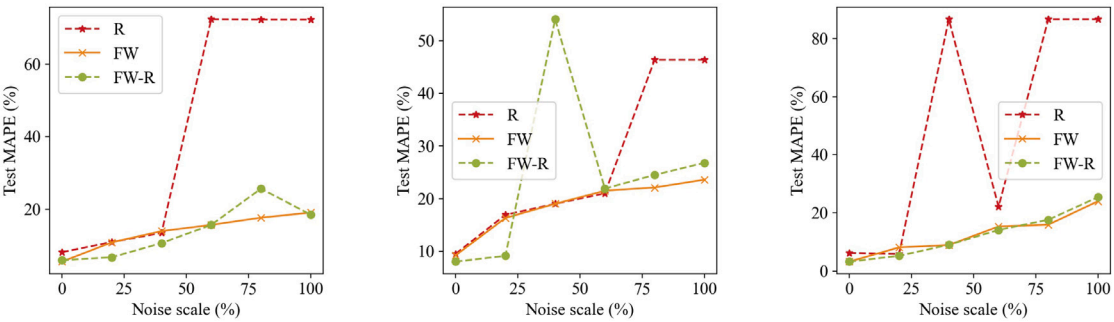

Fig. 9. Model performances with demand noises under (a) base, (b) uncongested, and (c) congested demand.

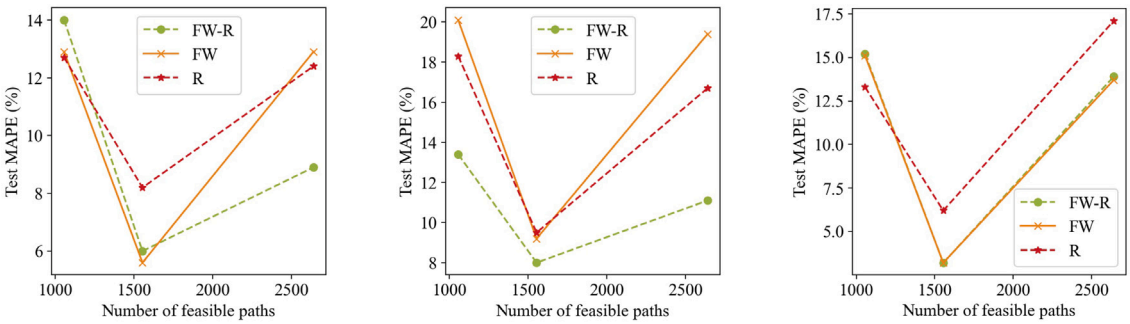

Fig. 10. Effects of inaccurate feasible path sets under (a) base, (b) uncongested, and (c) congested demand.

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 1587 paths in the ground-truth path set and we consider two scenarios: one with an incomplete path set of 1058 paths and the other with a redundant path set of 2645 paths. FW-R has the best performance when the selection of feasible paths is inaccurate (see Fig. 10). As shown in Fig. 10(a), 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 reduces it during training.

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. Conclusion

This study has proposed an end-to-end framework for transportation network equilibrium analysis, which directly learns travel choice preferences and the equilibrium state from multi-day link flow observations. Travelers' route choice preferences are represented with deep neural networks and embedded in a VI that prescribes the user equilibrium flow distribution. The neural network architecture is designed to simultaneously entail the cost function with monotonicity and Lipschitz continuity and accommodate the change of road network topology in the subsequent "what-if" planning analysis. To enable efficient batch operations in forward propagation, the VI is reformulated as an auxiliary fixed point problem and is then embedded as an implicit layer in a learning framework. Two iterative algorithms, i.e., fixed point iteration and root-finding method, are explored to solve the auxiliary fixed point problem. The proposed end-to-end framework is tested in the Sioux Falls network. Our numerical experiments show that the framework achieves 94.0% accuracy in link flow prediction when the road network topology changes and is robust to incomplete observations and input noises.

The proposed framework is flexible and can be applied to model various travel choices and learn supply-side components. We plan to extend the proposed framework to enable learning of another travel choice (e.g., trip generation, which implies learning travel demand functions), feasible path set, or combined choices (e.g., simultaneous choice of destination and route), by considering the availability of multi-source data (e.g., trajectories and observations of travel time, link flow, OD demand or trip productions/attractions). For each scenario, we will investigate appropriate architectures to facilitate end-to-end learning, and tailor efficient training algorithms for each learning problem.

We also 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. 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.

CRediT authorship contribution statement

Zhichen Liu: Methodology, Coding, Visualization, Writing – original draft, Review & editing. **Yafeng Yin:** Conceptualization, Methodology, Review & editing. **Fan Bai:** Funding acquisition, Review & editing. **Donald K. Grimm:** Funding acquisition, Review & editing.

Acknowledgments

The work described in this paper was partly supported by research grants from General Motors, United States, the USDOT Center for Connected and Automated Transportation, United States and National Science Foundation, United States.

Appendix A. Notations

Name	Notation	Description
Sets		
Node set	$\mathcal{N}$	
Link set	$\mathcal{A}$	
OD pair set	$\mathcal{R}$	
Path set	$\mathcal{P}$	
Path set for OD pair r	$\mathcal{P}_r$	
Sample set	$\mathcal{M}$	
Feasible demand set	$\overline{\mathcal{Q}}$	$\overline{\mathcal{Q}} = \{\overline{q} \in \mathbb{R}^{\lvert\mathcal{R}\rvert} : \overline{q} \geq 0\}$
Link flow observation set	$\overline{\mathcal{V}}$	$\overline{\mathcal{V}} = \{v \in \mathbb{R}^{\lvert\mathcal{A}\rvert} : v \geq 0\}$
Traveler characteristic set	$\mathcal{X}^{\mathcal{R}}$	$\mathcal{X}^{\mathcal{R}} \subset \mathbb{R}^{\lvert\mathcal{R}\rvert \times \lvert\mathcal{I}^{\mathcal{R}}\rvert}$
Characteristic set of a single traveler	$\mathcal{X}_r^{\mathcal{R}}$	$\mathcal{X}_r^{\mathcal{R}} \subset \mathbb{R}^{\lvert\mathcal{I}^{\mathcal{R}}\rvert}$
Feature set of a single link	$\mathcal{X}_a^{\mathcal{A}}$	$\mathcal{X}_a^{\mathcal{A}} \subset \mathbb{R}^{\lvert\mathcal{I}^{\mathcal{A}}\rvert}$
Feature set of a single node	$\mathcal{X}_n^{\mathcal{N}}$	$\mathcal{X}_n^{\mathcal{N}} \subset \mathbb{R}^{\lvert\mathcal{I}^{\mathcal{N}}\rvert}$
Feature set of a single path	$\mathcal{X}_p^{\mathcal{P}}$	$\mathcal{X}_p^{\mathcal{P}} \subset \mathbb{R}^{\lvert\mathcal{I}^{\mathcal{P}}\rvert}$
Attribute set	Π	$\Pi \subset \mathbb{R}^{\lvert\mathcal{P}\rvert \times \lvert\mathcal{S}\rvert}$
Attributes set of a single link	$\Pi_a^{\mathcal{A}}$	$\Pi_a^{\mathcal{A}} \subset \mathbb{R}^{\lvert\mathcal{S}^{\mathcal{A}}\rvert}$
Attribute set of a single node	$\Pi_n^{\mathcal{N}}$	$\Pi_n^{\mathcal{N}} \subset \mathbb{R}^{\lvert\mathcal{S}^{\mathcal{A}}\rvert}$
Attribute set of a single path	$\Pi_p^{\mathcal{P}}$	$\Pi_p^{\mathcal{P}} \subset \mathbb{R}^{\lvert\mathcal{S}^{\mathcal{A}}\rvert}$
Feasible link flow set	$\mathcal{V}^{[m]}$	$\mathcal{V}^{[m]} = \{v \in \mathbb{R}^{\lvert\mathcal{A}\rvert} : \ v \geq 0, \Sigma^{\top}\Lambda v = \overline{q}^{[m]}\}$
Feasible node flow set	$\mathcal{U}^{[m]}$	$\mathcal{U}^{[m]} = \{u \in \mathbb{R}^{\lvert\mathcal{N}\rvert} : \ u \geq 0, \Sigma^{\top}\Gamma u = \overline{q}^{[m]}\}$
Feasible path flow set	$\mathcal{H}^{[m]}$	$\mathcal{H}^{[m]} = \{h \in \mathbb{R}^{\lvert\mathcal{P}\rvert} : h \geq 0, \Sigma^{\top} h = \overline{q}^{[m]}\}$
Feasible auxiliary variable set	$\mathcal{Z}$	$\mathcal{Z} \subset \mathbb{R}^{\lvert\mathcal{P}\rvert}$
Feasible cost set	$\mathcal{C}$	$\mathcal{C} = \{c \in \mathbb{R}^{\lvert\mathcal{P}\rvert} : c \geq 0\}$
Parameters		
Path-link incidence matrix	Λ	$\Lambda \in \mathbb{R}^{\lvert\mathcal{P}\rvert \times \lvert\mathcal{A}\rvert}$, $\Lambda_{pa} = 1$ if link a is on path p
Path-node incidence matrix	Γ	$\Gamma \in \mathbb{R}^{\lvert\mathcal{P}\rvert \times \lvert\mathcal{N}\rvert}$, $\Gamma_{pn} = 1$ if node n is on path p
Path-OD incidence matrix	Σ	$\Sigma \in \mathbb{R}^{\lvert\mathcal{P}\rvert \times \lvert\mathcal{R}\rvert}$, $\Sigma_{pr} = 1$ if path p connects OD pair r
OD demands	$\overline{q}^{[m]}$	$\overline{q}^{[m]} \in \overline{\mathcal{Q}}$
Context features	x	$x \in \mathcal{X}$
Traveler characteristics	$x^{\mathcal{R}}$	$x^{\mathcal{R}} \in \mathcal{X}^{\mathcal{R}}$
Characteristics of a single traveler	$x_r^{\mathcal{R}}$	$x_r^{\mathcal{R}} \in \mathcal{X}_r^{\mathcal{R}}$
Road network features	$x^{\mathcal{G}}$	$x^{\mathcal{G}} \in \mathcal{X}^{\mathcal{G}}$
Features of a single link	$x_a^{\mathcal{A}}$	$x_a^{\mathcal{A}} \in \mathcal{X}_a^{\mathcal{A}}$
Features of a single node	$x_n^{\mathcal{N}}$	$x_n^{\mathcal{N}} \in \mathcal{X}_n^{\mathcal{N}}$
Features of a single path	$x_p^{\mathcal{P}}$	$x_p^{\mathcal{P}} \in \mathcal{X}_p^{\mathcal{P}}$
Cost function parameters	θ	
Variables		
Path flows	$h^{[m]}$	$h^{[m]} \in \mathcal{H}^{[m]}$
Link flows	$v^{[m]}$	$v^{[m]} \in \mathcal{V}^{[m]}$
node flows	$u^{[m]}$	$u^{[m]} \in \mathcal{U}^{[m]}$
Auxiliary variables	$z^{[m]}$	$z^{[m]} \in \mathcal{Z}$
Costs	$c^{[m]}$	$c^{[m]} \in \mathcal{C}$
Weights	w	$w \in \mathcal{W}$
Attributes	$\pi^{[m]}$	$\pi^{[m]} \in \Pi$
Link attributes	$\pi^{\mathcal{A}[m]}$	$\pi^{\mathcal{A}[m]} \in \Pi^{\mathcal{A}}$
Node attributes	$\pi^{\mathcal{N}[m]}$	$\pi^{\mathcal{N}[m]} \in \Pi^{\mathcal{N}}$
Path attributes	$\pi^{\mathcal{P}[m]}$	$\pi^{\mathcal{P}[m]} \in \Pi^{\mathcal{P}}$
Attributes of a single link	$\pi_a^{\mathcal{A}[m]}$	$\pi_a^{\mathcal{A}[m]} \in \Pi_a^{\mathcal{A}}$
Attributes of a single node	$\pi_n^{\mathcal{N}[m]}$	$\pi_n^{\mathcal{N}[m]} \in \Pi_n^{\mathcal{N}}$
Attributes of a single path	$\pi_p^{\mathcal{P}[m]}$	$\pi_p^{\mathcal{P}[m]} \in \Pi_p^{\mathcal{P}}$
Residual	ϕ	
Relative residual	φ	

Functions		
Cost function	C_θ	$C_\theta : \mathcal{H}^{[m]} \times \mathcal{X} \to \mathcal{C}$
Weight Net	F_θ	$F_\theta : \mathcal{X}^{\mathcal{R}} \to \mathcal{W}$
Attribute Net	G_θ	$G_\theta : \mathcal{H}^{[m]} \times \mathcal{X}^{\mathcal{G}} \to \Pi$
Link block	$g_\theta^{\mathcal{A}}$	$g_\theta^{\mathcal{A}} : \mathbb{R}_+ \times \mathcal{X}_a^{\mathcal{A}} \to \Pi_a^{\mathcal{A}}$
Node block	$g_\theta^{\mathcal{N}}$	$g_\theta^{\mathcal{N}} : \mathbb{R}_+ \times \mathcal{X}_n^{\mathcal{N}} \to \Pi_n^{\mathcal{N}}$
Path block	$g_\theta^{\mathcal{P}}$	$g_\theta^{\mathcal{P}} : \mathbb{R}_+ \times \mathcal{X}_p^{\mathcal{P}} \to \Pi_p^{\mathcal{P}}$
Loss function	$\mathcal{L}$	$\mathcal{L} : \mathcal{V} \times \overline{\mathcal{V}} \to \mathbb{R}$
Forward algorithm hyperparameters		
Maximum iterations	κ	κ_1 (κ_2) for fixed point iteration (root-finding method)
Stopping threshold	$\varepsilon > 0$	ε_1 (ε_2) for fixed point iteration (root-finding method)
Step size	$\alpha > 0$	α_1 (α_2) for fixed point iteration (root-finding method)
Step size adjust threshold	$\beta > 0$	β_1 (β_2) for fixed point iteration (root-finding method)
Step size adjust factor	$\gamma > 0$	γ_1 (γ_2) for fixed point iteration (root-finding method)

Appendix B. 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. $\|A\| = \sup_{x \neq 0} \frac{\|Ax\|}{\|x\|}$ represents the spectral norm of matrix A. The Jacobian matrix of a vector-to-vector function $F(x) : \mathbb{R}^n \to \mathbb{R}^m$ is denoted as $J_F(x) \in \mathbb{R}^{m \times n}$.

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 2 (*Monotonicity*)**.** A function $F(x) : \mathbb{R}^n \to \mathbb{R}^m$ is monotone if

$$\langle F(x) - F(y), x - y \rangle \geq 0, \quad \forall x, y \in \mathbb{R}^n.$$

A differentiable function $F(x) : \mathbb{R}^n \to \mathbb{R}^n$ is monotone if and only if its Jacobian matrix $\mathrm{J}_F(x) \in \mathbb{R}^{n \times n}$ is positive-semidefinite for all $x \in \mathbb{R}^n$.

Definition 3 (*Lipschitz Continuity*)**.** A function $F(x) : \mathbb{R}^n \to \mathbb{R}^n$ is L-Lipschitz continuous if there exists $L > 0$, such that

$$\|F(x) - F(y)\| \leq L \|x - y\|, \quad \forall x, y \in \mathbb{R}^n.$$

A differentiable function $F(x) : \mathbb{R}^n \to \mathbb{R}^n$ is L-Lipschitz continuous if and only if its Jacobian matrix $\mathrm{J}_F(x) \in \mathbb{R}^{n \times n}$ has finite spectral norms for all $x \in \mathbb{R}^n$.

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., $g^{\mathcal{A}}(v_a) : \mathbb{R}_+ \to \mathbb{R}$. By assumption, $g^{\mathcal{A}}(v_a)$ is monotone and Lipschitz continuous with respect to v_a, i.e., $0 \leq \frac{dg^{\mathcal{A}}}{dv_a} \leq L$ with $L > 0$.

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

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

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

Recall that the attributes π are the product of path-link incidence matrix Λ and link attributes. The Attribute Net $G(h) : \mathbb{R}^{|\mathcal{P}|} \to \mathbb{R}^{|\mathcal{P}|}$ is now defined as a self-mapping with respect to path flows, i.e., $G(h) = \Lambda G^{\mathcal{A}}(\Lambda^\top h)$. It follows that the Jacobian matrix of $G(h)$,

$$\mathrm{J}_G(h) = \Lambda \, \mathrm{J}_{G^{\mathcal{A}}}(v) \, \Lambda^\top,$$

is symmetric and positive-semidefinite. Path-link incidence matrix Λ is a 0–1 matrix with bounded spectral norm. As per Cauchy–Schwarz inequality, the spectral norm $\|\mathrm{J}_G(h)\| \leq L\|\Lambda\|^2$.

The cost function $C(h) : \mathbb{R}^{|\mathcal{P}|} \to \mathbb{R}^{|\mathcal{P}|}$ is formulated as $C(h) = \Sigma\, w \odot G(h)$. Suppose the weights are positive ($w > 0$), the Jacobian matrix of the cost function,

$$\mathrm{J}_C(h) = \mathrm{Diag}(\Sigma\, w)\, \mathrm{J}_G(h),$$

is the product of two symmetric positive-semidefinite matrices and thus symmetric positive-semidefinite with spectral norm bounded by $\|\mathrm{Diag}(\Sigma\, w)\| \, \|\mathrm{J}_G(h)\|$. It is equivalent to say the cost function $C(h)$ is monotone and Lipschitz continuous with respect to the path flows h. 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 ith column of weights and G_i denote the ith column of attributes. The cost function $C(h)$ is:

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

Suppose each block is column-wise monotone and Lipschitz continuity, $w_i \odot G_i(h)$ is monotone and Lipschitz continuous following previous proof for one-column scenarios. Monotonicity and Lipschitz continuity are preserved under summation, its follows that the cost function $C(h)$ is monotone and Lipschitz continuous with respect to the path flows h.

Additionally, it is straightforward to show that the Jacobian matrix of the cost function $\mathrm{J}_C(h)$ is the sum of $|S|$ symmetric matrix and thus is symmetric. Suppose $\mathrm{J}_C(h)$ is real everywhere, there exists a scalar function $y(h) : \mathbb{R}^{|\mathcal{P}|} \to \mathbb{R}$ such that $C(h)$ is the gradient of y (Emberton, 2008). Under mild assumptions that y is closed and proper, the monotonicity of $C(h)$ is equivalent to maximal monotonicity (Ryu and Yin, 2021). This completes the proof.

Appendix C. Proof sketch of the auxiliary fixed point reformulation

To begin with, we formally state three relevant definitions.

Definition 4 (*Resolvent and Reflected Resolvent of a Monotone Operator*)**.** Consider a maximal operator $\mathbb{A} : \mathbb{R}^n \to \mathbb{R}^n$ and $\alpha > 0$, the resolvent of $\alpha\mathbb{A}$ is:

$$\mathbb{J}_{\alpha\mathbb{A}} \triangleq (\mathrm{I} + \alpha\mathbb{A})^{-1},$$

and the reflected resolvent of $\alpha\mathbb{A}$ is:

$$\mathbb{R}_{\alpha\mathbb{A}} \triangleq 2\mathbb{J}_{\alpha\mathbb{A}} - \mathrm{I}.$$

Definition 5 (*Indicator Function*)**.** For $\Omega \subset \mathbb{R}^k$, the indicator function is

$$\delta_\Omega(u) = \begin{cases} 0 & \text{if } u \in \Omega \\ \infty & \text{otherwise} \end{cases}$$

The resolvent of the subgradient of indicator function $\partial\delta_\Omega$ is just the projection operator i.e., $\mathbb{J}_{\partial\delta_\Omega} = \mathbb{P}_\Omega$.

Definition 6 (*Cocoercivity*)**.** A single-valued operator $\mathbb{A} : \mathbb{R}^n \to \mathbb{R}^n$ is β-cocoercive if $\beta > 0$ and

$$\langle \mathbb{A}x - \mathbb{A}y, x - y\rangle \geq \beta\|x - y\|^2 \quad \forall x, y \in \mathbb{R}^n.$$

Cocoercivity is the dual property of strong monotonicity. When $\mathbb{A}$ is β-cocoercive, $\mathbb{A}$ is $(1/\beta)$-Lipschitz continuous. The converse is not necessarily true.

Two theorems from Ryu and Yin (2021) are cited without proof.

Theorem 3 (*Three Operator Splitting*)**.** *Consider three maximal monotone operators A, B, and C, with C single-valued. Then for any $\alpha > 0$, h is a solution to the operator inclusion problem if and only if there is a z is the fixed point of operator $(\frac{1}{2}\mathbb{I} + \frac{1}{2}\mathbb{T})$ and $h = \mathbb{J}_{\alpha\mathbb{B}} z$.*

$$0 \in (\mathbb{A} + \mathbb{B} + \mathbb{C})h \quad \Leftrightarrow \quad \left(\frac{1}{2}\mathbb{I} + \frac{1}{2}\mathbb{S}\right) z = z, \quad h = \mathbb{J}_{\alpha\mathbb{B}} z \tag{11}$$

where $\mathbb{S} = \mathbb{R}_{\alpha\mathbb{A}}\left(\mathbb{R}_{\alpha\mathbb{B}} - \alpha\mathbb{C}\mathbb{J}_{\alpha\mathbb{B}}\right) - \alpha\mathbb{C}\mathbb{J}_{\alpha B}$.

Theorem 4 (*Convergence of Average Operator*)**.** *Suppose $\mathbb{C}$ is β-cocoercive and $\alpha \in (0, 2\beta)$, the fixed point iteration defined as,*

$$\begin{aligned} z^{(k+1/2)} &= \mathbb{J}_{\alpha\mathbb{B}}(z^{(k)}) \\ y^{(k+1)} &= \mathbb{J}_{\alpha\mathbb{A}}\left(2z^{(k+1/2)} - z^{(k)} - \alpha\mathbb{C}z^{(k+1/2)}\right) \\ z^{(k+1)} &= z^{(k)} + y^{(k+1)} - z^{(k+1/2)} \end{aligned}$$

converges to a fixed point if one exists.

Let $\partial\delta_{\mathcal{H}}(h)$ denote the subgradient of the indicator function on the feasible path flow set $\mathcal{H}$. Solving a constrained VI problem is equivalent to solving an operator inclusion problem (Heaton et al., 2021).

$$h^\star \in \mathbf{VI}(C_\theta(h, x), \mathcal{H}) \Longleftrightarrow 0 \in C_\theta(h, x) + \partial\delta_{\mathcal{H}}(h)$$

It is straightforward to show that $\delta_{\mathcal{H}}(u) = \delta_{\mathcal{H}_1}(u) + \delta_{\mathcal{H}_2}(u)$ for $\mathcal{H} = \mathcal{H}_1 \cap \mathcal{H}_2$. And in our case, it follows:

$$h^\star \in \mathbf{VI}(C_\theta(h, x), \mathcal{H}_1 \cap \mathcal{H}_2) \Longleftrightarrow 0 \in C_\theta(h, x) + \partial\delta_{\mathcal{H}_1}(h) + \partial\delta_{\mathcal{H}_2}(h)$$

Let $\mathbb{A} = \partial\delta_{\mathcal{H}_2}$, $\mathbb{B} = \partial\delta_{\mathcal{H}_1}$, and $\mathbb{C} = C_\theta(h, x)$ which is maximal monotone and $1/L$-co-coercivity as shown in Appendices B and D. Apply it to Theorem 3, it follows:

$$\begin{aligned}\mathbb{T} = \frac{1}{2}\mathbb{I} + \frac{1}{2}\mathbb{S} &= \mathbb{I} - \mathbb{J}_{\alpha\mathbb{B}} + \mathbb{J}_{\alpha\mathbb{A}}\left(\mathbb{R}_{\alpha\mathbb{B}} - \alpha\mathbb{C}\mathbb{J}_{\alpha\mathbb{B}}\right)\\ &= \mathbb{I} - \mathbb{J}_{\alpha\partial\delta_{\mathcal{H}_1}} + \mathbb{J}_{\alpha\partial\delta_{\mathcal{H}_2}}\left(\mathbb{R}_{\alpha\partial\delta_{\mathcal{H}_1}} - \alpha\, C_\theta\left(\mathbb{J}_{\alpha\partial\delta_{\mathcal{H}_1}}\right)\right)\\ &= \mathbb{I} - \mathbb{P}_{\mathcal{H}_1} + \mathbb{P}_{\mathcal{H}_2}\left(2\mathbb{P}_{\mathcal{H}_1} - \mathbb{I} - \alpha\, C_\theta\left(\mathbb{P}_{\mathcal{H}_1}\right)\right)\end{aligned}$$

Appendix D. Proof of Proposition 1

Appendix B shows the cost function $C(h)$ is the gradient of a differentiable convex function $y(h)$. In this case, its L-Lipschitz continuity is equivalent to $1/L$-co-coercivity (Ryu and Yin, 2021). According to Theorem 4, given $C_\theta(h, x)$ is $1/L$-co-coercive and $\alpha \in (0, \frac{2}{L})$, $\frac{1}{2}\mathbb{I} + \frac{1}{2}\mathbb{S}$ is an average operator. Starting with any point $z^0 \in \mathbb{R}^n$, the following fixed point iteration converges to the fixed point of operator $\mathbb{T}$.

$$\begin{aligned}z^{(k+1/2)} &= \mathbb{P}_{\mathcal{H}_1}(z^{(k)})\\ y^{(k+1)} &= \mathbb{P}_{2}\left(2z^{(k+1/2)} - z^{(k)} - \alpha\, C_\theta\left(z^{(k+1/2)}\right)\right)\\ z^{(k+1)} &= z^{(k)} - z^{(k+1/2)} + y^{(k+1)}\end{aligned}$$

And the equilibrium flow $h^\star = \mathbb{J}_{\alpha\mathbb{B}} z^\star = \mathbb{P}_{\mathcal{H}_1}(z^\star)$.

Appendix E. Discussion about fixed point approximation for implicit differentiation

In backpropagation, the gradient of interest is:

$$\left(\frac{\partial z^\star}{\partial \theta}\right)^\top \left(\frac{\partial \mathcal{L}}{\partial z^\star}\right) = \left(\frac{\partial \mathbb{T}_\theta(z^\star, x)}{\partial \theta}\right)^\top \underbrace{\left(I - \frac{\partial \mathbb{T}_\theta(z^\star, x)}{\partial z^\star}\right)^{-\top}\left(\frac{\partial \mathcal{L}}{\partial z^\star}\right)}_{g}, \tag{12}$$

where $\frac{\partial \mathcal{L}}{\partial z^\star}$ denotes the input gradient of the implicit layer. It turns out that g can be rearranged as the solution of another fixed point problem, defined as:

$$g = \left(\frac{\partial \mathbb{T}_\theta(z^\star, x)}{\partial z^\star}\right)^\top g + \left(\frac{\partial \mathcal{L}}{\partial z^\star}\right).$$

After solving for g with fixed point iteration, the gradient of interest in Eq. (12) can be calculated with via typical automatic differentiation. The fixed point iteration converges if the Jacobian $\frac{\partial \mathbb{T}_\theta(z^\star, x)}{\partial \theta}$ is a stable matrix with maximum eigenvalue that has a magnitude less than one. Previous studies show that these iterations typically are convergent in practice (Bai et al., 2019).

References

Bai, S., Kolter, J.Z., Koltun, V., 2019. Deep equilibrium models. Adv. Neural Inf. Process. Syst. 32.

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

Bekhor, S., Toledo, T., 2005. Investigating path-based solution algorithms to the stochastic user equilibrium problem. Transp. Res. B 39 (3), 279–295.

Bibi, A., Ghanem, B., Koltun, V., Ranftl, R., 2019. Deep Layers as Stochastic Solvers. OpenReview. net.

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. Transp. Res. C 68, 285–299.

Davis, D., Yin, W., 2017. A three-operator splitting scheme and its optimization applications. Set-Valued Var. Anal. 25 (4), 829–858.

Emberton, J., 2008. An Elucidation of Vector Calculus Through Differential Forms. Citeseer.

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, Vol. 34, No. 01. pp. 630–637.

Frejinger, E., Bierlaire, M., Ben-Akiva, M., 2009. Sampling of alternatives for route choice modeling. Transp. Res. B 43 (10), 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.

Gouk, H., Frank, E., Pfahringer, B., Cree, M.J., 2021. Regularisation of neural networks by enforcing lipschitz continuity. Mach. Learn. 110 (2), 393–416.

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. J. Transp. Eng. 130 (2), 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. Adv. Neural Inf. Process. Syst. 34, 9639–9650.

Jeihani, M., Lawe, S., Connolly, J., 2006. Improving Traffic Assignment Model Using Intersection Delay Function. Technical Report.

Lawphongpanich, S., Hearn, D.W., 2004. An MPEC approach to second-best toll pricing. Math. Program. 101 (1), 33–55.

Li, J., Yu, J., Nie, Y., Wang, Z., 2020. End-to-end learning and intervention in games. Adv. Neural Inf. Process. Syst. 33, 16653–16665.

Ma, W., Pi, X., Qian, S., 2020. Estimating multi-class dynamic origin-destination demand through a forward-backward algorithm on computational graphs. Transp. Res. C 119, 102747.

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. Transp. Sci. 38 (3), 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. Transp. Res. B 140, 236–261.

Simon, H.A., 1955. A behavioral model of rational choice. Q. J. Econ. 69 (1), 99–118.

Spana, S., Du, L., 2022. Optimal information perturbation for traffic congestion mitigation: Gaussian process regression and optimization. Transp. Res. C 138, 103647.

Tobin, R.L., Friesz, T.L., 1988. Sensitivity analysis for equilibrium network flow. Transp. Sci. 22 (4), 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. J. Environ. Econ. Manag. 62 (1), 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.

Tversky, A., Kahneman, D., 1992. Advances in prospect theory: Cumulative representation of uncertainty. J. Risk Uncertain. 5 (4), 297–323.

Van Der Pol, M., Currie, G., Kromm, S., Ryan, M., 2014. Specification of the utility function in discrete choice experiments. Value Health 17 (2), 297–301.

Virmaux, A., Scaman, K., 2018. Lipschitz regularity of deep neural networks: analysis and efficient estimation. Adv. Neural Inf. Process. Syst. 31.

Walker, H.F., Ni, P., 2011. Anderson acceleration for fixed-point iterations. SIAM J. Numer. Anal. 49 (4), 1715–1735.

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 (1), e0146850.

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

Wardrop, J.G., 1952. Road paper. some theoretical aspects of road traffic research. Proc. Inst. Civ. Eng. 1 (3), 325–362.

Wehenkel, A., Louppe, G., 2019. Unconstrained monotonic neural networks. Adv. Neural Inf. Process. Syst. 32.

Wong, M., Farooq, B., 2021. ResLogit: A residual neural network logit model for data-driven choice modelling. Transp. Res. C 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. Transp. Res. C 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. Transp. Res. B 45 (2), 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. Transp. Sci. 35 (2), 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, Vol. 33, No. 01. pp. 5668–5675.

Yin, Y., Lawphongpanich, S., Lou, Y., 2008. Estimating investment requirement for maintaining and improving highway systems. Transp. Res. C 16 (2), 199–211.