Learning-based solutions to nonlinear hyperbolic PDEs: Empirical insights on generalization errors

Bilal Thonnam Thodi[1,2], **Sai Venkata Ramana Ambadipudi**[2], **Saif Eddin Jabari**[1,2]
[1]New York University Tandon School of Engineering, Brooklyn, NY
[2]New York University Abu Dhabi, Abu Dhabi, UAE
{btt1, sa183, sej7}@nyu.edu

Abstract

We study learning weak solutions to nonlinear hyperbolic partial differential equations (H-PDE), which have been difficult to learn due to discontinuities in their solutions. We use a physics-informed variant of the Fourier Neural Operator (π-FNO) to learn the weak solutions. We empirically quantify the generalization/out-of-sample error of the π-FNO solver as a function of input complexity, i.e., the distributions of initial and boundary conditions. Our testing results show that π-FNO generalizes well to unseen initial and boundary conditions. We find that the generalization error grows linearly with input complexity. Further, adding a physics-informed regularizer improved the prediction of discontinuities in the solution. We use the Lighthill-Witham-Richards (LWR) traffic flow model as a guiding example to illustrate the results.

1 Introduction

Hyperbolic partial differential equations (H-PDEs) arise in the study of nonlinear wave motion in applications such as the behavior of water waves, vehicular traffic flow, and even in high-speed physics like blast waves and sonic booms. Typically, (inviscid) H-PDEs are of the form $u_t + cu_x = 0$, where u is a conserved quantity. Wave motion in H-PDEs is characterized by finite propagation speeds c. The solutions are characteristic trajectories emanating from the datum (e.g., the initial conditions). Nonlinear H-PDEs, where the nonlinearity results in different wave propagation speeds, e.g., $c = c(u)$, result in characteristic lines that may cross somewhere in the domain, and at those points, the solution is multi-valued, which means that the problem does not have a solution in the classical sense. Here, weak solutions that permit discontinuities are introduced, which are conventionally solved using finite volume or finite difference-based numerical schemes [1, 2].

Recently, deep learning-based (DL) methods have appeared that aim to overcome the limitations of conventional numerical solvers, namely, high computational cost, grid dependence, and knowledge of complete initial and boundary conditions. The DL solvers have shown remarkable results for both forward and inverse problems, especially where the PDE solutions are smooth [3, 4]. However, DL solutions having high irregularities (e.g., discontinuities), such as those arising in the weak solutions of H-PDEs, are only partially successful [5–8]. This is partly due to the ill-posedness of H-PDEs, i.e., derivatives are not defined everywhere, and the H-PDE residuals do not form a correct physics loss metric. Further, these studies lacked a systematic procedure to evaluate the out-of-sample performance or the generalization error.

To this end, we explore Fourier Neural Operators (FNO) [4] for learning weak solutions of H-PDEs. We propose a systematic training and testing experiment where the FNO solver is trained with solutions of elementary input conditions and evaluated for solutions of general input conditions. We quantify the empirical generalization error of the FNO solver as a function of the input complexity, i.e., how the out-of-sample error grows as the distribution of input conditions becomes more general.

Machine Learning and the Physical Sciences workshop, NeurIPS 2022.

To capture physically consistent weak solutions (e.g., shocks), we propose using an integral form of the H-PDE (written in discrete form) as the physics loss function instead of the H-PDE residual. We demonstrate these results using vehicular traffic flow as an example.

2 Methods

Problem setting We consider the LWR traffic flow model [9, 10] as the guiding example for nonlinear scalar H-PDEs. The LWR model is a continuum description of the flow of vehicles on a road. Consider a space-time domain $\Omega \subset \mathbb{R} \times \mathbb{R}_+$. Denote by $u(x,t) : \Omega \to [0, u_{\max}]$ the density of traffic at position $x \in \mathbb{R}$ and time $t \in \mathbb{R}_+$. Here density refers to the average number of vehicles per unit length. Let $q(x,t) : \Omega \to [0, q_{\max}]$ the traffic flux, which is the number of vehicles crossing a point per unit time. The LWR model describes the evolution of traffic density based on the principle of vehicular conservation:

$$u_t + q_x = 0; \;\; u(x,0) = \bar{u}_0; \;\; u(x_b,t) = \bar{u}_b; \;\; (x,t) \in \Omega; \tag{1}$$

where $u_t \equiv \frac{\partial u}{\partial t}$, $\bar{u}_0$ is an initial condition, and $\bar{u}_b$ is a boundary condition. For the LWR model, one typically prescribes a flux function $q = f(u)$, where $f(u) : [0, u_{\max}] \to [0, q_{\max}]$, which is concave in the context of traffic flow. The LWR model is given as

$$u_t + f'(u)u_x = 0; \;\; u(x,0) = \bar{u}_0; \;\; u(x_b,t) = \bar{u}_b; \;\; (x,t) \in \Omega; \tag{2}$$

We refer to (2) as the *forward* problem if $\bar{u}_b$ corresponds to solutions at the boundary points $(x_b,t) \in \partial\Omega$ and as the *inverse* problem if $\bar{u}_b$ is replaced with densities at random points in the domain Ω i.e., $(x_b,t) \in \Omega$. We use $f(u) = uv_{\max}(1 - u/u_{\max})$, where $u_{\max}$ is the maximum traffic density and $v_{\max}$ is the maximum traffic speed. A major difficulty in solving the *forward* problem is handling discontinuities when they appear in the solution due to nonlinearity in the system. The *inverse* problem poses the additional challenge that the boundary condition is unknown, and the solution needs to be inferred from partially observed measurements. We tackle both challenges using a deep learning-based solver, discussed below.

Fourier Neural Operator solver We are interested in learning the solution operator that maps the input function $a := (\bar{u}_0, \bar{u}_b)$ to the weak solution $u(x,t)$ over Ω. Let the input function be $a \in \mathcal{A}$ and output function be $u \in \mathcal{U}$. The problem (2) can be rephrased as one of learning an operator $\mathcal{G} : \mathcal{A} \to \mathcal{U}$. We approximate $\mathcal{G}$ using the Fourier Neural Operator (FNO) of Li et al. [4], represented by the parametric model $\mathcal{G}_\Theta$, and given by

$$\widehat{u} = \mathcal{G}_\Theta(a) = \big(\mathcal{Q} \circ \mathcal{F}^{(L)} \circ \mathcal{F}^{(L-1)} \circ \cdots \circ \mathcal{F}^{(2)} \circ \mathcal{F}^{(1)} \circ \mathcal{P}\big)(a) \tag{3}$$

where $\{\mathcal{F}^{(l)}\}_{l=1}^L$ is a set of Fourier operators while $\mathcal{P}$ and $\mathcal{Q}$ are projection operators. A single Fourier operator is defined as

$$\mathcal{F}(z) = \sigma\Big(W \cdot z + \mathrm{IFFT}\big(R \cdot \mathrm{FFT}(z)\big)\Big) \tag{4}$$

for any latent input z, where FFT and IFFT denote the Fourier transform and its inverse. $\Theta = \{W^{(l)}, R^{(l)}\}_{l=1}^L$ is the set of trainable parameters of the FNO operator $\mathcal{G}_\Theta$. Our motivation for using the FNO operator (3) is its efficient approximation in the Fourier domain and it ability to learn complex dynamics [4]. The parameter complexity involved in learning $\mathcal{G}_\Theta$ depends on the size of R, which is independent of the domain size $|\Omega|$.

Physics-informed training The FNO model (3) can be trained end-to-end in a supervised learning framework over an appropriately defined loss function. We perform physics-informed training where the loss function has two parts – an empirical training data loss L_{data} and a physics constraint loss L_{phys} to emulate the PDE operator. L_{data} is simply $\sum_{n \in N} ||\mathcal{G}_\Theta(a^{(n)}) - u^{(n)}||_2$, where N is the number of samples and $u^{(n)}$ is the n^{th} sample solution.

One could use the PDE residual of (2) to form the physics loss L_{phys}, as in the conventional physics-informed neural networks [3]. However, (2) is not a well-posed PDE, i.e., derivatives are not defined everywhere, especially near discontinuities and hence not a well-defined loss metric. Thus, we resort to the integral form of (2) to form the physics loss L_{phys} as follows:

$$L_{\text{phys}} = \Big\| \big\langle u(x,t+\Delta t) - u(x,t) + \frac{\Delta t}{\Delta x}\left[q(x - \Delta x/2, t) - q(x + \Delta x/2, t)\right] \big\rangle_{(x,t)} \Big\|_2 \tag{5}$$

where $u(x,t)$ and $q(x,t)$ are defined over a discretization of Ω, and $\langle\cdot\rangle_{(x,t)}$ is a concatenation operator. Accordingly, we train two different FNO models with two different objective functions, as shown below:

$$\text{(i) FNO model : } \min_{\Theta} L_{\text{data}} \quad \text{(ii) } \pi\text{-FNO model : } \min_{\Theta} L_{\text{data}} + \lambda L_{\text{phys}} \tag{6}$$

Data and training experiments We obtain the training and testing dataset by numerically simulating (2) for different input conditions. The numerical scheme used for generating the datasets and the FNO training codes is described in Appendix A and Appendix C.

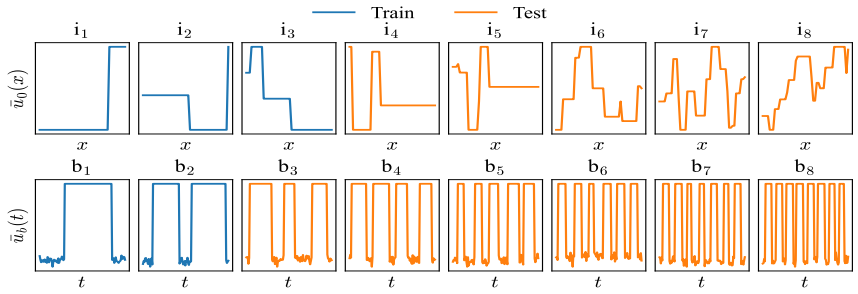

Figure 1: Initial and boundary conditions used for training and testing the FNO solver.

We perform systematic training and testing experiments to quantify the generalization performance of the FNO solver. During training, the FNO solver is shown solutions of simple dynamics, for instance, generated from step-wise initial conditions (a single vehicle queue) and one or two stepped wavelet-like boundary conditions (emulating vehicles stopping at traffic lights). The FNO solver is then tested with solutions of complex dynamics generated from general initial conditions (multiple vehicle queues) and multi-stepped wavelet-like boundary conditions (multiple stops at traffic lights). The set of initial and boundary conditions used in training and testing are shown in Figure 1. Training data consists of solutions using inputs (i_0-i_3, b_0-b_2). Testing data are made of (i_0-i_9, b_0-b_2) for evaluating initial conditions, and (i_0-i_3, b_0-b_8) for evaluating boundary conditions.

The goal is to train the FNO solver with simple solutions and assess the out-of-sample error as input conditions become complex. For the LWR traffic flow example, the input complexity refers to (a) how the vehicles are distributed spatially at time $t = 0$ (i.e., $\bar{u}_0$) and (b) the impact of traffic signals at the road exit $x = x_{\max}$ (i.e., boundary condition $\bar{u}_b$). These two input factors put together can generate complex dynamics u. Training details are summarized in Appendix B.

3 Results

Generalization error v/s input complexity The out-of-sample errors as a function of input conditions are summarized in Figure 2. Each data point is the average mean absolute error (MAE) of 50 samples. A piece-wise linear trendline is fitted to the error plots. The trendline shows that MAE is nearly constant for input conditions seen during training (i_0-i_3, b_0-b_2) and steadily increases for input conditions at testing (i_4-i_9, b_3-b_9).

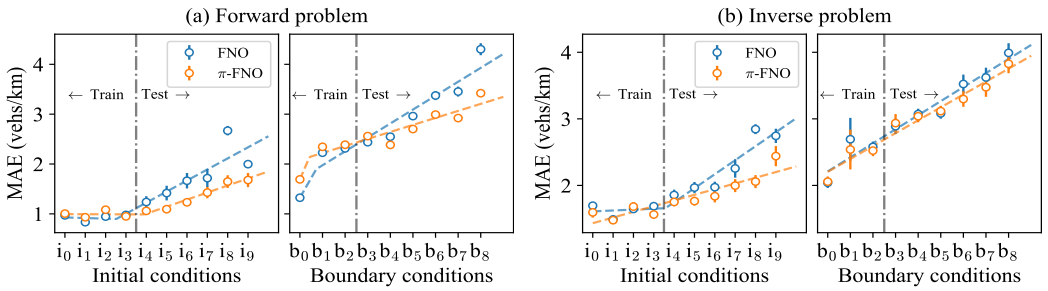

Figure 2: Empirical generalization (out-of-sample) error as function of input conditions.

We define the generalization error as the slope of this trendline, summarized in Table 1. For instance, Figure 2 says that MAE increases by 0.141 vehs/km ($+0.12\%$) for every additional traffic light at the road exit $\bar{u}_b$ using π-FNO model. Similarly, an additional non-uniformity in the initial condition $\bar{u}_0$ increases MAE by 0.142 vehs/km ($+0.12\%$).

Table 1: Error rates (vehs/km)

	FNO		π-FNO	
	$\bar{u}_0$	$\bar{u}_b$	$\bar{u}_0$	$\bar{u}_b$
Forward	0.222	0.279	0.142	0.141
Inverse	0.210	0.211	0.085	0.194

Figure 2 concludes that the generalization error for the testing set grows linearly with the input complexity for both the forward and inverse problems. Also, π-FNO incurs lower error rates compared to the FNO model.

Sample predictions The predicted and true solutions for four different input conditions are shown in Figure 3. The π-FNO solver is only shown density dynamics similar to Figure 3a at the training stage. Figure 3b-3d are the density dynamics that π-FNO solver generalizes flawlessly. This implies that the π-FNO solver learned to capture the traffic queuing dynamics (i.e., vehicle queue formation and dissipation) as a function of the boundary flows. Also, the inverse problem results in Figure 3c, and 3d shows that π-FNO can qualitatively recover solution without knowledge of boundary conditions but only with sparse trajectory measurements.

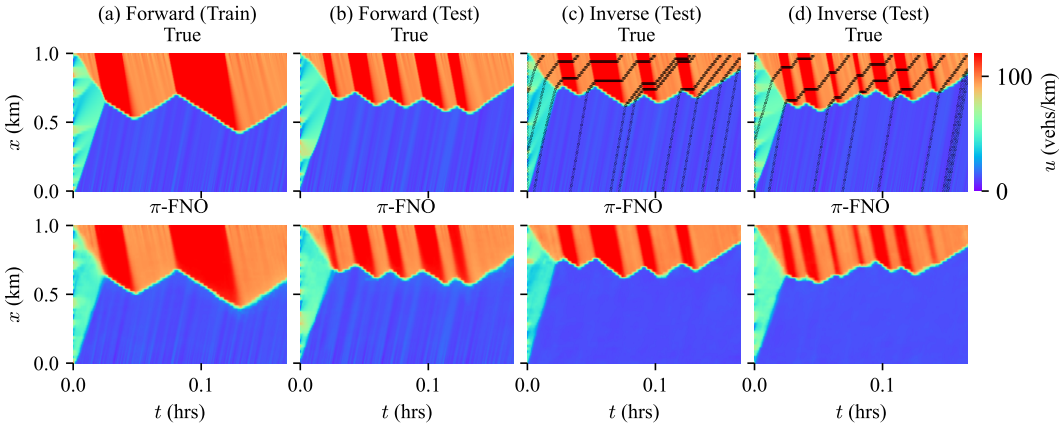

Figure 3: Comparison of true and π-FNO predicted solutions. Sub-figure (a) is a training scenario, and (b)-(d) are test scenarios. The dotted curve is the input locations of $\bar{u}_b$ for the inverse problem.

Physics-informing on the behavior of shock solutions In Figure 4, we compare solutions (zoomed-in) for a constant and a step-wise initial conditions. We see that the FNO model (physics-uninformed) produces noisy predictions, whereas the π-FNO model (physics-informed) smoothens these artifacts. This suggests the benefits of physics-informing in producing physically consistent solutions.

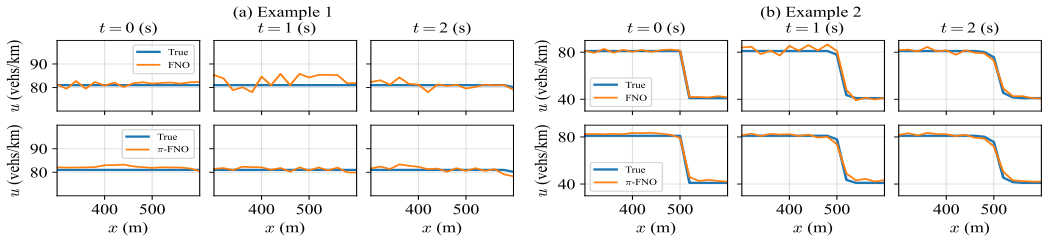

Figure 4: Solution profiles as a function of x

4 Summary and Discussion

We explored the Fourier Neural Operator (FNO) in learning the weak solutions of scalar non-linear hyperbolic partial differential equations (H-PDEs), which has seen limited success in the deep

learning-based computational literature. We focused on quantifying the generalization error of the FNO solver as a function of input complexity, taking vehicular traffic flow as an example. We found that the FNO solver can be trained using simple solutions and can easily generalize to complex inputs with an acceptable error tolerance – the out-of-sample errors grew linearly with input complexity. We also showed the benefits of physics-informing in predicting physically consistent solutions, e.g., correct shock behavior. To the authors' knowledge, this is the first empirical study on generalization capabilities of learning-based solvers for non-linear H-PDEs.

A limitation of the current solver is that it requires a regular grid-like computational domain, partly due to the Fourier Transform operator. Our efforts continue to extend these solvers to irregular graph-like computational domains, e.g., to solve traffic flow on a city network.

5 Broader Impact

The modeling and control of dynamical systems such as road traffic, water supply, and communication networks are fundamental in functioning large-scale urban cities, which contribute to one-third of the global carbon footprint. The techniques developed in this study aid in building low-resource computational tools for controlling these dynamical systems, which are often modeled as partial differential equations. The data-driven learning paradigms studied in this work help advance the development of edge computing infrastructures such as connected vehicles, smart personal gadgets, and even augmented/virtual reality applications.

References

[1] Randall J LeVeque. *Numerical methods for conservation laws*, volume 214. Springer, 1992.

[2] G B Whitham. *Linear and nonlinear waves*. John Wiley & Sons, New York, NY, 1974.

[3] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[4] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=c8P9NQVtmnO.

[5] Ruben Rodriguez-Torrado, Pablo Ruiz, Luis Cueto-Felgueroso, Michael Cerny Green, Tyler Friesen, Sebastien Matringe, and Julian Togelius. Physics-informed attention-based neural network for hyperbolic partial differential equations: application to the buckley–leverett problem. *Scientific Reports*, 12(1):7557, 2022. doi: 10.1038/s41598-022-11058-2. URL https://doi.org/10.1038/s41598-022-11058-2.

[6] Xiaoping Zhang, Tao Cheng, and Lili Ju. Implicit form neural network for learning scalar hyperbolic conservation laws. In Joan Bruna, Jan Hesthaven, and Lenka Zdeborova, editors, *Proceedings of the 2nd Mathematical and Scientific Machine Learning Conference*, volume 145 of *Proceedings of Machine Learning Research*, pages 1082–1098. PMLR, 16–19 Aug 2022. URL https://proceedings.mlr.press/v145/zhang22a.html.

[7] Ravi G. Patel, Indu Manickam, Nathaniel A. Trask, Mitchell A. Wood, Myoungkyu Lee, Ignacio Tomas, and Eric C. Cyr. Thermodynamically consistent physics-informed neural networks for hyperbolic systems. *Journal of Computational Physics*, 449:110754, 2022. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2021.110754. URL https://www.sciencedirect.com/science/article/pii/S0021999121006495.

[8] Ameya D. Jagtap, Ehsan Kharazmi, and George Em Karniadakis. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 365:113028, 2020. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2020.113028. URL https://www.sciencedirect.com/science/article/pii/S0045782520302127.

[9] M. Lighthill and G. Whitham. On kinematic waves. II. A theory of traffic flow on long crowded roads. In *Royal Society of London. Series A, Mathematical and Physical Sciences*, volume 229, pages 317–345, 1955.

[10] Paul I. Richards. Shock Waves on the Highway. *Operations Research*, 4(1):42–51, February 1956. ISSN 0030-364X, 1526-5463.

Acknowledgments and Disclosure of Funding

This work was supported in part by the NYUAD Center for Interacting Urban Networks (CITIES), funded by Tamkeen under the NYUAD Research Institute Award CG001, and in part by the NYUAD Research Center on Stability, Instability, and Turbulence (SITE), funded by Tamkeen under the NYUAD Research Institute Award CG002. The views expressed in this article are those of the authors and do not reflect the opinions of CITIES, SITE, or their funding agencies.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 4
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Appendix A-C.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] The hyperparameters and training details are discussed in Appendix B
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] The error bars are plotted in Figure 2
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section B
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See Section C
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Godunov scheme for LWR simulation

The LWR traffic flow model can be solved using Godunov's numerical scheme. Let i and j be the space and time index, denote $u_{(i,j)}$ as the average traffic density for each cell (i, j). The traffic density is updated as:

$$u_{(i,j+1)} = u_{(i,j)} + \frac{\Delta t}{\Delta x}\left[q_{(i-1/2,j)} - q_{(i+1/2,j)}\right] \tag{7}$$

where $q_{(i-1/2,j)}$ is the cell boundary flux from cell $(i-1, j)$ to cell (i, j). Δt and Δ x denote the temporal and spatial width. The boundary flux is given by,

$$q_{(i-1/2,j)} = \min\left\{Q_{\text{dem}}^{(i,j-1)},\ Q_{\text{sup}}^{(i,j)}\right\}$$
$$q_{(i+1/2,j)} = \min\left\{Q_{\text{dem}}^{(i,j)},\ Q_{\text{sup}}^{(i,j+1)}\right\}$$

where

$$Q_{\text{dem}}^{(i,j)} = \begin{cases} f\left(u^{(i,j)}\right) & \text{if } u^{(i,j)} \leq u_{\text{cr}} \\ q_{\max} & \text{otherwise} \end{cases}$$
$$Q_{\text{sup}}^{(i,j)} = \begin{cases} f\left(u^{(i,j)}\right) & \text{if } u^{(i,j)} > u_{\text{cr}} \\ q_{\max} & \text{otherwise} \end{cases}$$

where $f(u)$ is the traffic flux function (also called Fundamental relation in traffic flow literature)

$$f(u) = u(1 - u/u_{max})v_{max}.$$

where v_{max} is maximum speed, u_{max} is maximum density and $q_{\max}$ is the maximum traffic flux. The above numerical scheme is run for different sets of initial and boundary conditions shown in Figure 1. For the inverse problem, we draw random vehicle trajectories to represent $\bar{u}_b$. For all the testing, we used 10 random vehicle trajectories as the input.

B Hyperparameters and training details

The hyperparameters used in the study are summarized in Table 2. The training hyperparameters are chosen by an independent trial-and-error experiment. We modified the original FNO implementation from Li et al. [4] to implement the π-FNO model in python using the PyTorch machine learning library; see Appendix C. The training was performed on a GPU cluster (NVIDIA Tesla V100 32GB) and the total run time was around ~ 45 min for a single training experiment.

Table 2: Hyper-parameters used in the study

LWR simulation		**FNO model**		**FNO training**	
space dimension	1000 m	# Fourier layers L	4	coefficient λ	2.0
time dimension	600 sec	# modes in x dimension	24	# epochs	500
discretization size	(50×600)	# modes in t dimension	128	batch size	128
cell width Δx	20 m	# latent width	64	learning rate	$1e-3$
cell width Δt	1 sec	lifting operator $\mathcal{P}$	Linear layer with depth 128	learning rate scheduler	step-wise
max density $u_{\max}$	120 vehs/km			optimizer	Adam GD
max flow $q_{\max}$	1800 vehs/hr	lifting operator $\mathcal{Q}$	2-layer FNN with depth 128	# training samples	5200
				# testing samples	400

The objective function coefficient λ is obtained by training a series of π-FNO models for $\lambda = \{0, 0.05, 0.1, \ldots, 0.95, 1.0\}$. The λ value corresponding to the least validation error is considered optimal.

C Datasets and codes

The datasets, codes, and pretrained models are shared at https://github.com/bilzinet/pifno under the MIT license.