TrafficFlowGAN: Physics-informed Flow based Generative Adversarial Network for Uncertainty Quantification

Zhaobin Mo[1], Yongjie Fu[1], Daran Xu[2], and Xuan Di[1,3]

[1] Department of Civil Engineering and Engineering Mechanics, Columbia University, New York, USA
`{zm2302,yf2578,xd2187}@columbia.edu`

[2] Department of Statistics, Columbia University, New York, USA
`dx2207@columbia.edu`

[3] Data Science Institute, Columbia University, New York, USA

Abstract. This paper proposes the TrafficFlowGAN, a physics-informed flow based generative adversarial network (GAN), for uncertainty quantification (UQ) of dynamical systems. TrafficFlowGAN adopts a normalizing flow model as the generator to explicitly estimate the data likelihood. This flow model is trained to maximize the data likelihood and to generate synthetic data that can fool a convolutional discriminator. We further regularize this training process using prior physics information, so-called physics-informed deep learning (PIDL). To the best of our knowledge, we are the first to propose an integration of normalizing flow, GAN and PIDL for the UQ problems. We take the traffic state estimation (TSE), which aims to estimate the traffic variables (e.g. traffic density and velocity) using partially observed data, as an example to demonstrate the performance of our proposed model. We conduct numerical experiments where the proposed model is applied to learn the solutions of stochastic differential equations. The results demonstrate the robustness and accuracy of the proposed model, together with the ability to learn a machine learning surrogate model. We also test it on a real-world dataset, the Next Generation SIMulation (NGSIM), to show that the proposed TrafficFlowGAN can outperform the baselines, including the pure flow model, the physics-informed flow model, and the flow based GAN model. Source code and data are available at https://github.com/ZhaobinMo/TrafficFlowGAN.

Keywords: Uncertainty Quantification (UQ) · Normalizing Flow · Generative Adversarial Networks (GAN) · Physics-informed Deep Learning (PIDL).

1 Introduction

Uncertainty quantification (UQ) is the process of characterizing the uncertainty of system dynamics, accounting for two main sources of uncertainty [4]. The *aleatoric uncertainty* (or data uncertainty) refers to uncertainty arising from external factors, such as measurement noise, and random initial or boundary conditions. The *epistemic uncertainty* arises from the inadequate knowledge of the underlying model, such as inherent stochasticity in human behavior. With these random factors, UQ of dynamical systems is crucial to avoid potential system oscillation or cascading errors.

Two classes of methods are developed to characterize the aforementioned sources of uncertainty. The **physics-based** method assumes that the observations are generated from underlying physics imposed by Gaussian noise; thus filtering methods or Bayesian inference can be applied to propagate uncertainty. However, the physics-based method suffers from limitations such as non-Gaussian likelihoods and high-dimensional posterior distributions [18]. In contrast, the **data-driven** method, such as generative adversarial networks (GAN) [7], tries to characterize any distribution of data directly without making any assumption of noise. Recently, there is a growing trend in integrating physics-based models into the data-driven framework, namely, **physics-informed deep learning** (PIDL) [12]. PIDL-based UQ methods can characterize generic data distribution while ensuring physics consistency.

Among all PIDL models for UQ, the physics-informed GAN is the most widely used, which has been applied to solve stochastic differential equations [17,18,5] and quantify uncertainty in various domains [15,14]. Although GAN generates high-quality samples [9] through adversarial training, it has stability and convergence issues. Moreover, as GAN cannot calculate the model likelihood, it may miss important modes of the data distribution, namely, *mode collapse* [16]. In contrast, normalizing flow [6] calculates the exact data likelihood and is trained using maximum likelihood estimation (MLE), which is an effective way to avoid mode collapse. However, applying PIDL to the normalizing flow is still at its nascent stage and we only find one relevant work [10].

Leveraging the pros of both the MLE and adversarial training, Flow-GAN, a combination of normalizing flow and GAN, is first introduced in [9], which can achieve both high data likelihood and good sample qualities. Flow-GAN has been applied to manifold learning [3] and image-to-image translation [8]. Little research has been documented that applies Flow-GAN to UQ problems.

In this paper, we propose TrafficFlowGAN that leverages likelihood training, adversarial training, and PIDL for the UQ problems. To the best of our knowledge, we are the first to integrate these three methods for the UQ problems. Main contributions of this paper include:

- We propose a hybrid generative model, TrafficFlowGAN, combining normalizing flow and GAN to achieve both high likelihoods and good sample qualities and to avoid mode collapse.
- We incorporate physics information into the TrafficFlowGAN model for estimation accuracy and data efficiency, and use neural network surrogate models to learn the inter-relations between the physics variables at the same time.
- We apply the TrafficFlowGAN model to learn solutions of second-order stochastic partial differential equations (PDEs), and demonstrate the perfor-

mance of TrafficFlowGAN by applying it to a traffic state estimation (TSE) problem with real-world data.

The rest of this paper is organized as follows: Section 2 introduces the background and related work. Section 3 introduces the structure of TrafficFlowGAN for the UQ problems. Section 4 demonstrates how TrafficFlowGAN learns solutions of a PDE and the relations between physics variables using a neural network surrogate model. Section 5 demonstrates how TrafficFlowGAN characterizes uncertainty from the real-world data in the TSE problem, where two traffic models, i.e. the Aw–Rascle–Zhang (ARZ) [2] and the Lighthill-Whitham-Richards (LWR) [11] models, are used as the physics components. Section 6 concludes our work and projects future directions in this promising arena.

2 Background and Related Work

2.1 Normalizing Flow

The flow model aims to learn an invertible function $\boldsymbol{z} = f_\theta(\boldsymbol{u}) : \mathbb{R}^D \mapsto \mathbb{R}^D$, where data $\boldsymbol{u}$ is sampled from a distribution $p_{\text{data}}(\boldsymbol{u})$ and $\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})$ is a random noise of the same dimension as the data. The data likelihood $p_\theta(\boldsymbol{u})$ can be explicitly expressed by using the *change of variable formula*:

$$p_\theta(\boldsymbol{u}) = p_{\boldsymbol{z}}(\boldsymbol{z}) \left| \det \left(\frac{\partial f_\theta^{-1}(\boldsymbol{z})}{\partial \boldsymbol{z}} \right) \right|^{-1}. \tag{1}$$

To compute $p_\theta(\boldsymbol{u})$, it is nontrivial to choose a latent variable $\boldsymbol{z}$ that has an easy form and to design the invertible function f_θ so that the Jacobian determinant can be easily computed. A common selection of the latent variable $\boldsymbol{z}$ is the standard Gaussian, i.e. $p_{\boldsymbol{z}}(\boldsymbol{z}) \sim \mathcal{N}(0, \boldsymbol{I}_D)$. To compute the Jacobian determinant, RealNVP [6] designs the invertible function f_θ as an *affine coupling transformation* following the equations below:

$$f_\theta := \begin{cases} \boldsymbol{z}_{1:d} = \boldsymbol{u}_{1:d} \\ \boldsymbol{z}_{d+1:D} = \boldsymbol{u}_{d+1:D} \odot e^{k_\theta(\boldsymbol{u}_{1:d})} + b_\theta(\boldsymbol{u}_{1:d}) \end{cases}, \tag{2}$$

where $\boldsymbol{u}$ and $\boldsymbol{z}$ are split into two partitions at the dth elements. The *scale function* k_θ and the *translation function* b_θ are neural networks to be learned, which constitute the affine transformation of the partition $\boldsymbol{u}_{1:d}$. $\odot$ is the Hadamard product or element-wise product. By this design of invertible function, the Jacobian determinant in Eq. 1 can be computed by

$$\left| \det \left(\frac{\partial f_\theta^{-1}(\boldsymbol{z})}{\partial \boldsymbol{z}} \right) \right|^{-1} = e^{\sum_j [k_\theta(\boldsymbol{z}_{1:d})]_j}, \tag{3}$$

where j is the index of the element of $k_\theta(\boldsymbol{z}_{1:d})$. The inverse function f_θ^{-1} can also be obtained by

$$f_\theta^{-1} := \begin{cases} \boldsymbol{u}_{1:d} = \boldsymbol{z}_{1:d} \\ \boldsymbol{u}_{d+1:D} = (\boldsymbol{z}_{d+1:D} - b_\theta(\boldsymbol{z}_{1:d}))/e^{k_\theta(\boldsymbol{z}_{1:d})} \end{cases}. \tag{4}$$

To better accommodate the complex data distribution, f_θ is further modeled as a sequence of affine coupling transformations: $f_\theta = f_L \circ ... \circ f_1$, where L is the total number of transformations. Let f_l be the lth invertible mapping and $\boldsymbol{h}^{(l)}$ be the lth latent variable that satisfies $\boldsymbol{h}^{(l)} = f_l(\boldsymbol{h}^{(l-1)})$, where $\boldsymbol{h}^{(0)} = \boldsymbol{u}$ and $\boldsymbol{h}^{(L)} = \boldsymbol{z}$. Then the log-likelihood of $\boldsymbol{u}$ can be computed by:

$$\log p_\theta(\boldsymbol{u}) = \log p_{\boldsymbol{z}}(\boldsymbol{z}) + \sum_{l=0}^{L-1} \log \left| \det \left(\frac{\partial f_l^{-1}(\boldsymbol{h}^{(l)})}{\partial \boldsymbol{h}^{(l)}} \right) \right|^{-1}. \tag{5}$$

The computation of the log-likelihood of $\boldsymbol{z}$ is straightforward as $\boldsymbol{z}$ is assumed to follow a standard Gaussian distribution, and each Jacobian determinant can be calculated following Eq. 3. Thus, the exact data likelihood is tractable and the flow model can be trained by the MLE.

2.2 Generative Adversarial Network (GAN)

GAN aims to train a generator G_θ to learn the mapping from a random noise $\boldsymbol{z}$ to the corresponding state variables $\boldsymbol{u}$, i.e. $G_\theta : \boldsymbol{z} \to \boldsymbol{u}$. The objective of the generator G_θ is to fool an adversarially trained discriminator D_ϕ. Different GAN variants use different metrics to evaluate the divergence between the prediction distribution and the data distribution, such as the Kullback-Leibler (KL) divergence, the Jensen-Shannon divergence, and the Wasserstein distance [1]. Among these metrics, the Wasserstein distance has received growing popularity for its stability, which optimizes the following objective:

$$\min_\theta \max_{\phi \in \mathcal{F}} \mathbb{E}_{p_{\text{data}}(\boldsymbol{u})} \left[D_\phi(\boldsymbol{u}) \right] - \mathbb{E}_{p_{\boldsymbol{z}}(\boldsymbol{z})} \left[D_\phi(G_\theta(\boldsymbol{z})) \right], \tag{6}$$

where θ and ϕ are the parameters of the generator and the discriminator, respectively. $\mathcal{F}$ is defined such that D_ϕ is 1-Lipschitz.

3 Framework of TrafficFlowGAN

3.1 Problem Statement

Define the spatial and temporal domains as $\mathcal{X}$ and $\mathcal{T}$, respectively. $(x,t) \in \mathcal{X} \times \mathcal{T}$ is the spatio-temporal coordinate ("coordinate" for short). It is assumed that the state variable $\boldsymbol{u}$ can only be observed by limited number of sensors placed at fixed locations and at a specific frequency. Thus, we further define the *observed (labeled) region* $O \subseteq \mathcal{X} \times \mathcal{T}$ as the spatio-temporal region where the state variable $\boldsymbol{u}$ is observed, and thereby the *unobserved (unlabeled) region* $C = \mathcal{X} \times \mathcal{T} \setminus O$. We represent the continuous domain in a discrete manner using grid points. Thus, the observed region O and the unobserved region C can be represented as collections of discrete coordinates: $O = \{(x_o^{(i)}, t_o^{(i)})\}_{i=1}^{N_o}$ and $C = \{(x_c^{(j)}, t_c^{(j)})\}_{j=1}^{Nc}$, where i and j are the indices of observed and unobserved coordinates, respectively; N_o, N_c are the numbers of observed and unobserved coordinates, respectively.

The state variable $\boldsymbol{u}$ is a random variable for each coordinate, i.e. $\boldsymbol{u} \sim p_{data}(\boldsymbol{u}|x,t)$. Our goal is to train a generator such that its prediction distribution distribution $p_\theta(\hat{\boldsymbol{u}}|x,t)$ matches the data distribution $p_{data}(\boldsymbol{u}|x,t)$. Below we will introduce how to achieve this goal with our proposed TrafficFlowGAN.

3.2 Overview of TrafficFlowGAN Structure

An overview of TrafficFlowGAN is illustrated in Fig. 1, which consists of three main components, namely, a conditional flow f_θ, a physics-based computational graph, and a convolutional discriminator D_ϕ. The data is illustrated as a heatmap in the spatio-temporal domain. We assume the data is measured by sensors at fixed locations. Due to limited range each sensor can cover, the observation region consists of separate horizontal "strips." The observed and unobserved coordinates are fed into the conditional flow model to generate predictions $\hat{\boldsymbol{u}}_o$ and $\hat{\boldsymbol{u}}_c$, respectively. Those predictions bifurcate into two branches. In the upper branch, $\hat{\boldsymbol{u}}_c$ are fed into a physics-based computational graph, which encodes physics laws, to calculate the physics loss function. This process of calculating physics loss from the unobserved coordinates is illustrated by grey arrows. In the lower branch, the prediction states $\hat{\boldsymbol{u}}_o$ and the observed states $\boldsymbol{u}_o$ are then reshaped to constitute the prediction matrix $\hat{M}$ and the observation matrix M, respectively. These two matrices are then fed into the convolutional discriminator. The process of calculating the adversarial loss from observed coordinates and states is illustrated by blue arrows.

We will detail each component sequentially and explain how we integrate those components in the following subsections.

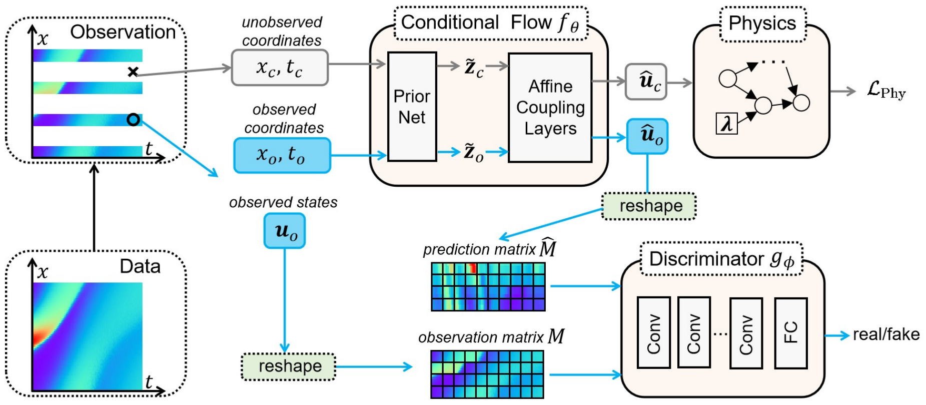


Fig. 1: Structure of the TrafficFlowGAN.

3.3 Convolutional Neural Network as the Discriminator

Existing physics-informed GAN models construct the discriminator as a *fully connected network*. By explicitly adding the spatio-temporal coordinates into the input space, the discriminator is expected to make decisions based on the *spatio-temporal pattern*, which will be represented better by the convolutional neural network (CNN). In this work, we propose to use CNN as the discriminator.

The discriminator D_ϕ consists of a sequence of convolutional layers (Conv) followed by a fully connected layer (FC). This FC layer outputs a 1×1 scalar indicating if the input matrix is from observations or predictions. The pooling layer is not used in this structure, as there is no requirement for compression in our task.

The reshape of the observation $\boldsymbol{u}$ is straightforward. As we represent the spatio-temporal domain in a discrete manner, $\boldsymbol{u}(x,t)$ for each coordinate can be viewed as a "pixel", and the dimension of $\boldsymbol{u}$ is its number of channels. This is the same for reshaping the prediction $\hat{\boldsymbol{u}}$ to get the prediction matrix $\hat{M}$. Note that due to randomness in data and predictions, we can sample multiple observation matrices $\{M^{(i)}\}_{i=1}^{N_\omega}$ and prediction matrices $\{\hat{M}^{(i)}\}_{i=1}^{N_\omega}$, where N_ω is the total number of sampling.

The discriminator D_ϕ can be updated by minimizing the Wasserstein loss:

$$\mathcal{L}_D(\phi) = -\frac{1}{N_\omega}\sum_{i=1}^{N_\omega} D_\phi(M^{(i)}) - D_\phi(\hat{M}^{(i)}). \tag{7}$$

3.4 Conditional Flow as the Generator

We construct a conditional flow as our generator, as illustrated in Fig. 2. Assume $\boldsymbol{u}$ has two elements, i.e. u_1 and u_2. Different from the tradition normalizing flow, we add a *prior network* (p-net) to transform the standard Gaussian prior $\boldsymbol{z} = (z_1, z_2)$ to $\tilde{\boldsymbol{z}} = (\tilde{z}_1, \tilde{z}_2)$ with shifting and scaling, considering that the magnitude of uncertainty at different (x,t) coordinates can be different. The prior network takes as input the coordinate (x,t) and outputs the prior mean $\boldsymbol{\mu} = (\mu_1, \mu_2)$ and prior standard deviation $\boldsymbol{\sigma} = (\sigma_1, \sigma_2)$; thus $\tilde{\boldsymbol{z}} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$. The prior network is followed by affine coupling layers. Each affine coupling layer consists of a scale function (k-net) and a translation function (b-net), as introduced in Section 2.1.

Based on [19] and our experiment, the exponential operation in Eq. 2 is numerically unstable, which may result in gradient explosion. Instead of using the RealNVP, we replace the exponential operation in Eq. 2 with a Sigmoid operation:

$$f_\theta := \begin{cases} \tilde{\boldsymbol{z}}_{1:d} = \boldsymbol{u}_{1:d} \\ \tilde{\boldsymbol{z}}_{d+1:D} = \boldsymbol{u}_{d+1:D} \odot \text{Sigmoid}(k_\theta(\boldsymbol{u}_{1:d}; x, t)) + b_\theta(\boldsymbol{u}_{1:d}; x, t) \end{cases}, \tag{8}$$

and the calculation of Jacobian determinant in Eq. 5 is thus changed to

$$\left|\det\left(\frac{\partial f_\theta^{-1}(\tilde{\boldsymbol{z}})}{\partial \tilde{\boldsymbol{z}}}\right)\right|^{-1} = \sum_j \left[k_\theta(\tilde{\boldsymbol{z}}_{1:d}; x, t)\right]_j, \tag{9}$$

where j is the element index of $k_\theta(\tilde{\boldsymbol{z}}_{1:d}; x, t)$. We define a likelihood loss function for the generator as:

$$\begin{aligned}\mathcal{L}_{\mathrm{NLL}}(\theta) &= -\sum_{i=1}^{N_\omega} \sum_{(x_o,t_o)\in O} \log p_\theta(\boldsymbol{u}|x_o, t_o, \omega^{(i)}) \\ &= -\sum_{i=1}^{N_\omega} \sum_{(x_o,t_o)\in O} \log p_{\tilde{\boldsymbol{z}}}(\tilde{\boldsymbol{z}}|x_o, t_o) + \sum_{l=0}^{L-1} \log \left|\det\left(\frac{\partial f_l^{-1}(\boldsymbol{h}^{(l)})}{\partial \boldsymbol{h}^{(l)}}\right)\right|^{-1},\end{aligned} \tag{10}$$

which is the summation of negative log-likelihood (NLL) over all observed coordinates and random events.

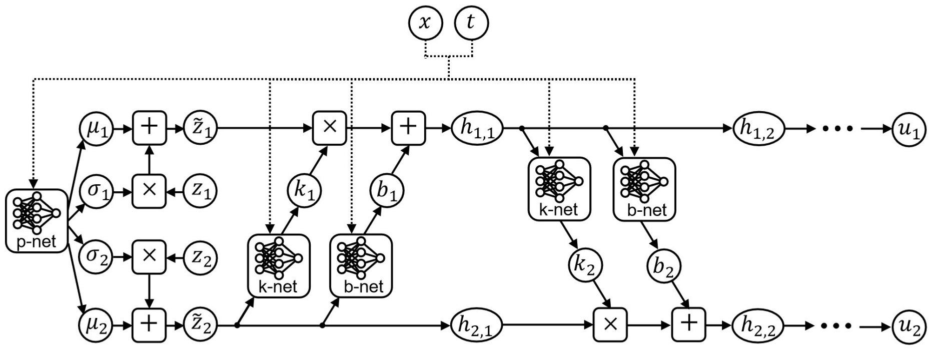


Fig. 2: Structure of the conditional flow with a prior network.

Apart from the likelihood loss $\mathcal{L}_{\mathrm{NLL}}$, the flow generator f_θ can also be trained with the discriminator D_ϕ through adversarial training. The adversarial loss for the generator is depicted as:

$$\mathcal{L}_{\mathrm{Adv}}(\theta) = -\frac{1}{N_\omega}\sum_{i=1}^{N_\omega} D_\phi(\hat{M}^{(i)}), \tag{11}$$

which uses the Wasserstein objective defined in Eq. 6; $\hat{M}$ is the prediction matrix and N_ω is the total number of sampling.

Using the adversarial loss $\mathcal{L}_{\mathrm{Adv}}$ alone is prone to mode collapse. We demonstrate below how the likelihood loss $\mathcal{L}_{\mathrm{NLL}}$ can mitigate the mode collapse by one example. Suppose the data is generated from a mixture model of two Gaussian distributions $\mathcal{N}(-1, 1)$ and $\mathcal{N}(1, 1)$. By adversarial training, the generator may end up only generating one mode, say $\mathcal{N}(-1, 1)$. In this case, the discriminator cannot distinguish between the samples and the ground truth. If the MLE is used, the likelihood of the missing mode is very low, resulting in a high overall NLL. Thus, the likelihood loss $\mathcal{L}_{\mathrm{NLL}}$ can guide the generator to leave the current local optimum.

3.5 Physics Regularization

The conditional flow model is further regularized by the physics-informed computational graph, which encodes physics prior knowledge like partial differential equations (PDE).

Suppose data follows laws that can be depicted as stochastic PDEs below:

$$\begin{aligned} \boldsymbol{u}_t(x,t;\omega)+\mathcal{N}_x[\boldsymbol{u}(x,t;\omega);\lambda(\omega)]&=\mathbf{0}, \quad (x,t)\in\mathcal{X}\times\mathcal{T},\omega\in\Omega,\\ \mathcal{B}[\boldsymbol{u}(x,t;\omega)]&=\mathbf{0}, \quad (x,t)\in\partial\mathcal{X}\times\mathcal{T},\\ \mathcal{I}[\boldsymbol{u}(x,0;\omega)]&=\mathbf{0}, \quad x\in\mathcal{X}, \end{aligned} \tag{12}$$

where, $\boldsymbol{u}_t$ is its partial derivative of $\boldsymbol{u}$ with regard to t; $\partial\mathcal{X}$ is the boundary of the space domain $\mathcal{X}$; $\mathcal{N}_x$ is a non-linear differential operator; $\mathcal{B}$ is a boundary condition operator; $\mathcal{I}$ is an initial condition operator; λ is the parameters of the PDEs. ω is a random event sampled from the probability space Ω, which represents uncertainties residing in the PDE parameters or the boundary and initial conditions.

By encoding physics information, the physics-informed flow generator has an additional learning objective on the unobserved region C, which encourages the prediction of the generator to follow the physics defined by the PDE. The physics loss function is defined as:

$$\mathcal{L}_{\text{Phy}}(\theta,\lambda)=\mathbb{E}_{q(x_c,t_c)}\left|\mathbb{E}_{p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(\hat{\boldsymbol{u}}_c)_t+\mathcal{N}_x[\hat{\boldsymbol{u}}_c;\lambda]\right]\right|^2, \tag{13}$$

where $\hat{\boldsymbol{u}}_c=f_\theta(x_c,t_c,\boldsymbol{z})$ is the prediction of the generator on the unobserved region. This physics loss function serves as a regularization term for the generator. If the flow generator f_θ is well trained, the physics loss needs to be as close to zero as possible.

3.6 Training of TrafficFlowGAN

The loss function of the flow model is a weighted sum of the likelihood loss, adversarial loss, and physics loss:

$$\mathcal{L}_f(\theta)=\alpha\mathcal{L}_{\text{NLL}}(\theta)+\beta\mathcal{L}_{\text{Adv}}(\theta)+\gamma\mathcal{L}_{\text{Phy}}(\theta,\lambda), \tag{14}$$

where $\alpha,\beta,\gamma\in(0,1]$ are hyperparameters that determine the contribution of each loss component. With the generator loss $\mathcal{L}_f(\theta)$, the discriminator loss $\mathcal{L}_D(\phi)$ defined in Eq. 7, and physics loss $\mathcal{L}_{\text{Phy}}(\theta,\lambda)$ defined in Eq. 13, we are ready to introduce the training algorithm as shown in Algorithm 1.

4 Numerical Experiment: Learning Solutions of A Known Second-order PDE

In this experiment, we apply TrafficFlowGAN to learn solutions of a known PDE and also to learn the relations of the PDE's parameters.

Algorithm 1 TrafficFlowGAN Training Algorithm.

Initialization:
Initialized physics parameters λ^0; Initialized networks parameters θ^0, ϕ^0; Training iterations $Iter$; Batch size m; Learning rate lr; Weights of loss functions α, β, and γ.

Input: The observation data $\{(x_o^{(i)}, t_o^{(i)}, \boldsymbol{u}_o^{(i)})\}_{i=1}^{N_o}$ and unobserved coordinates $\{x_c^{(j)}, t_c^{(j)}\}_{j=1}^{N_c}$.

```
1: for $k \in \{0, ..., Iter\}$ do
2:     Sample batches $\{(x_o^{(i)}, t_o^{(i)}, \boldsymbol{u}_o^{(i)})\}_{i=1}^{m}$ and $\{x_c^{(j)}, t_c^{(j)}\}_{j=1}^{m}$ from the observation data and unobserved coordinates, respectively
       // update the discriminator
3:     Calculate $\mathcal{L}_D$ by Eq. 7
4:     $\phi^{k+1} \leftarrow \phi^k - lr \cdot \text{Adam}(\phi^k, \nabla_\phi \mathcal{L}_D)$
       // update the generator
5:     Calculate $\mathcal{L}_{\text{NLL}}$ by Eq. 10, $\mathcal{L}_{\text{Adv}}$ by Eq. 11, and $\mathcal{L}_{\text{Phy}}$ by Eq. 13
6:     Calculate $\mathcal{L}_f$ by Eq. 14
7:     $\theta^{k+1} \leftarrow \theta^k - lr \cdot \text{Adam}(\theta^k, \nabla_\theta \mathcal{L}_f)$
       // update the physics
8:     $\lambda^{k+1} \leftarrow \lambda^k - lr \cdot \text{Adam}(\lambda^k, \nabla_\lambda \mathcal{L}_{\text{Phy}})$
9: end for
```

4.1 Numerical Data

The numerical data is generated from the ARZ model [2], which is a second-order PDE that is used to describe the traffic dynamics. It is depicted as

$$\begin{cases} \rho_t + (\rho u)_x = 0, \\ (u + h(\rho))_t + u(u + h(\rho))_x = (U_{eq}(\rho) - u)/\tau, \end{cases} \tag{15}$$

where,

$$h(\rho) = U_{eq}(0) - U_{eq}(\rho) \tag{16}$$

is the hesitation function and

$$U_{eq}(\rho) = u_{max}(1 - \rho/\rho_{max}) \tag{17}$$

is the equilibrium traffic velocity; traffic density ρ and traffic velocity u are the state variables, i.e. $\boldsymbol{u} = (\rho, u)$; τ is the relaxation parameter; ρ_{max} and u_{max} are the maximum traffic density and the maximum traffic velocity, respectively. In this experiment, we study a "ring road" in $t \in [0, 3]$ and $x \in [0, 1]$ with a boundary condition $\boldsymbol{u}(0, t) = \boldsymbol{u}(1, t)$. We set the parameters as $\rho_{max} = 1.13$, $u_{max} = 1.02$, and $\tau = 0.02$. We set the initial conditions of ρ and u as bell-shaped functions shown in Fig. 3(a). The x-axis is the space domain, and the y-axis is the initial value of ρ (blue line) and u (red line). We solve Eq. 15 using the Lax-Friedrichs scheme on a spatio-temporal grid of sizes 240×960, and the solutions $\rho(x, t)$ and

$u(x,t)$ are shown in Fig. 3(b) and Fig. 3(c), respectively. The dashed black lines indicate 4 locations where data is observed (the top and bottom dashed black lines indicate the same position as it is a ring road). We then add a white noise $\epsilon \sim \mathcal{N}(0, 0.02)$ to the solution to represent the uncertainty.

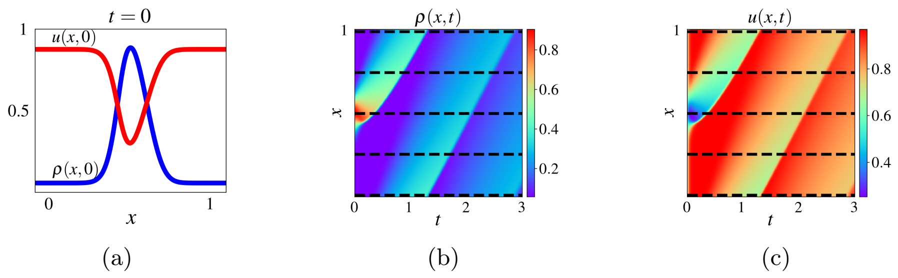


Fig. 3: Numerical data generator from the ARZ model. (a) is the bell-shaped initial ρ and u over $x \in [0,1]$; (b) and (c) are numerical solutions for ρ and u, respectively.

4.2 Physics-based Computational Graph

Fig. 4(a) illustrates the physics-based computational graph assosciated with the ARZ. We assume that the exact form of U_{eq} is unknown, and we use a *surrogate network* (s-net) to learn an approximate $\hat{U}_{eq}$. The corresponding physics loss for each line of Eq. 15 is as below:

$$\begin{cases} \mathcal{L}^{(1)}_{\text{ARZ}} = \left|\mathbb{E}_{\boldsymbol{z}}\left[\hat{\rho}_t + (\hat{\rho}\hat{u})_x\right]\right|^2 \\ L^{(2)}_{\text{ARZ}} = \left|\mathbb{E}_{\boldsymbol{z}}\left[(\hat{u} + h(\hat{\rho}))_t + \hat{u}(\hat{u} + h(\hat{\rho}))_x - (\hat{U}_{eq}(\hat{\rho}) - \hat{u})/\tau\right]\right|^2. \end{cases} \tag{18}$$

In implementation, derivatives with regard to x and t can be easily calculated by the Pytorch module `torch.autograd`.

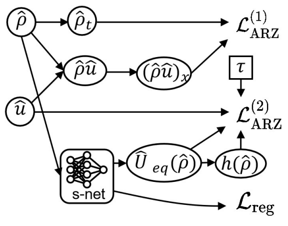
Additionally, we add a *shape constraint* to regularize the s-net to be monotonically decreasing for the ARZ physics loss, given the domain knowledge that the equilibrium speed U_{eq} decreases as the density ρ increases. This shape constraint is depicted as follows:

$$\mathcal{L}_{\text{reg}} = \int_a^b \max\left(0, \frac{\partial \hat{U}_{eq}(\rho)}{\partial \rho}\right) d\rho, \tag{19}$$

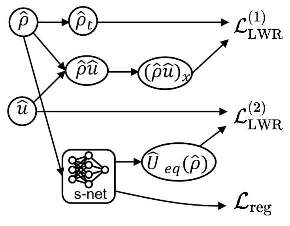
where hyperparameters a and b determine the interval where the shape constraint takes effect, e.g. $a = 0$ and $b = 1.13$ in this ring road experiment. Summarizing the loss terms defined in Eq. 18 and Eq. 19, the final physics loss can thus be written as:

$$\mathcal{L}_{\text{Phy}} = \eta\mathcal{L}^{(1)}_{\text{ARZ}} + (1-\eta)\mathcal{L}^{(2)}_{\text{ARZ}} + \xi\mathcal{L}_{\text{reg}}, \tag{20}$$

where $\eta \in (0, 1]$ and $\xi \in (0, \infty)$ are hyperparameters that control the weights.


$\hat{\rho}$
$\hat{\rho}_t$
$\hat{\rho}\hat{u}$
$(\hat{\rho}\hat{u})_x$
$\hat{u}$
τ
$\mathcal{L}^{(1)}_{\text{ARZ}}$
$\mathcal{L}^{(2)}_{\text{ARZ}}$
s-net
$\hat{U}_{eq}(\hat{\rho})$
$h(\hat{\rho})$
$\mathcal{L}_{\text{reg}}$

(a)


$\hat{\rho}$
$\hat{\rho}_t$
$\hat{\rho}\hat{u}$
$(\hat{\rho}\hat{u})_x$
$\hat{u}$
$\mathcal{L}^{(1)}_{\text{LWR}}$
$\mathcal{L}^{(2)}_{\text{LWR}}$
s-net
$\hat{U}_{eq}(\hat{\rho})$
$\mathcal{L}_{\text{reg}}$

(b)

Fig. 4: Physics with surrogate models. (a) is the physics for ARZ and (b) is the physics for LWR, which will be introduced in Section 5.2.

4.3 Experiment Setting

Experiments are conducted on a Google cloud workstation with 8 Intel Xeon E5-2686 v4 processors and an NVIDIA V100 Tensor Core GPU with 16 GB memory in Ubuntu 18.04.3. The learning rate for the Adam optimizer is 0.0005, and other configurations are kept as default. The configuration of the discriminator is different for different loop detectors, which are detailed in the supplementary materials.

4.4 Results

The results of the TrafficFlowGAN are shown in Fig. 5. Fig. 5(a) is the prediction of the traffic density. It demonstrates that the TrafficFlowGAN can reconstruct the traffic density with observations from 4 sensors. Two prediction snapshots at $t = 0.078$ and $t = 1.0$ shown in Fig. 5(b) and Fig. 5(c), respectively. The blue line stands for the mean of the ground truth; the dashed red line represents the mean of the prediction, and the yellow band is the prediction interval. Fig. 5(d) illustrates the relation between traffic density and traffic velocity learned by the s-net, i.e. $\hat{U}_{eq}(\hat{\rho})$. The solid blue line is the ground-truth relation $U_{eq}(\rho)$ that is defined in Eq. 17. The dashed black line and the dashed red line are the $\hat{U}_{eq}(\hat{\rho})$ at the 1st and the 15000th epochs, respectively. We can see that s-net manages to recover the underlying traffic density-velocity relation. The reason for the relatively poor performance for $\rho > 0.9$ is that the numerical data does not contain ρ that is bigger than 0.9, as indicated by the colorbar of Fig. 3(b).

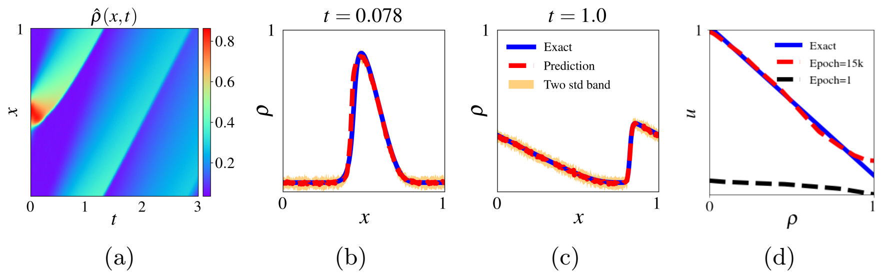


Fig. 5: Results of the TrafficFlowGAN on the numerical data generated by ARZ.

5 Case Study: Traffic State Estimation

Traffic state estimation (TSE) is an important traffic engineering problem that aims to infer traffic state variables represented by traffic density and velocity along a road segment from partial observations. In a nutshell, the goal of TSE is to learn a mapping from a spatio-temporal domain to traffic states, i.e. $f : (x,t) \rightarrow (\rho, u)$, using partial observations from fixed sensors (e.g. loop detectors).

5.1 Dataset

The Next Generation SIMulation (NGSIM)[4] is a real-world dataset that collects vehicle trajectory every 0.1 seconds. We focus on a 15-minutes data fragments collected on highway US 101. The traffic density and velocity are shown in Fig. 6.

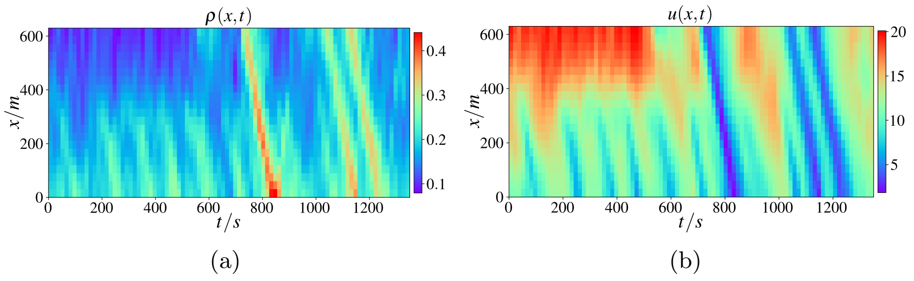


Fig. 6: NGSIM dataset, where (a) is the traffic density and (b) is the traffic velocity.

[4] www.fhwa.dot.gov/publications/research/operations/07030/index.cfm

5.2 Physics-based Computational Graph

As the underlying physics for the real-world scenario is unknown, in addition to the ARZ, we also adopt the LWR model as our physics. LWR is depicted as below:

$$\begin{cases} \rho_t + (\rho u)_x = 0, \\ u = U_{eq}(\rho) \triangleq u_{max}(1 - \rho/\rho_{max}), \end{cases} \tag{21}$$

which shares the same physics parameters, i.e. ρ_{max} and u_{max}, as the ARZ. The physics computational graph associated with LWR is illustrated in Fig. 4(b). The corresponding physics losses are as below:

$$\begin{cases} \mathcal{L}_{\mathrm{LWR}}^{(1)} = \left|\mathbb{E}_{\boldsymbol{z}}\left[\hat{\rho}_t + (\hat{\rho}\hat{u})_x\right]\right|^2 \\ L_{\mathrm{LWR}}^{(2)} = \left|\mathbb{E}_{\boldsymbol{z}}\left[(\hat{U}_{eq}(\hat{\rho}) - \hat{u})\right]\right|^2 \end{cases} . \tag{22}$$

5.3 Baselines and Metrics

We adopt the following baselines for comparison: the pure flow model, the physics-informed flow with ARZ as the physics (PhysFlow-ARZ), the physics-informed flow with LWR as the physics (PhysFlow-LWR), FLowGAN, and the ARZ-based extended Kalman filter (EKF) [13]. EKF applies a non- linear version of the Kalman filter and is widely used in nonlinear systems like the TSE.

We use the $\mathbb{L}^2$ relative percentage error (RE) to measure the difference between the mean of the prediction and that of the ground truth. The reason for choosing this metric is to mitigate the influence of the scale of the ground truth. In addition, the reverse Kullback-Leibler (KL) divergence is used to measure the difference between the prediction distribution and the sample distribution.

5.4 Results

Fig. 7 shows the REs (left two) and KL divergences (right two) of traffic density ρ and velocity u of TrafficFlowGAN and the baselines. The x-axis is the number of loop detectors. Different scatter types and colors are used to distinguish with different models. From this figure, we can see that TrafficFlowGAN-ARZ outperforms others across nearly all numbers of loop detectors for REs, and TrafficFlowGAN-LWR achieves the best performance for KL divergences. We also record the training time of each model when the number of loops is 10. The Flow-based models, including Flow and PhysFlow, cost 0.11 second per epoch. This means that the extra computational time from calculating the physics loss is negligible. The training time of the FlowGAN-based models, including TrafficFlowGAN and FlowGAN, is 0.31 second per epoch.

Fig. 8 show the predictions of traffic density (top row) and traffic velocity (bottom row). Fig. 8(a) and Fig. 8(d) present the heatmaps of traffic density and traffic velocity in spatio-temporal space. Those two predictions are close to the ground truth shown in Fig. 6. The other 4 subfigures show the snapshots of the prediction intervals of the traffic density and velocity.

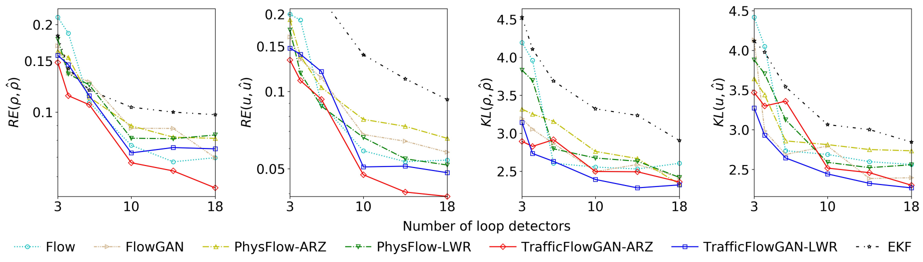


Fig. 7: The REs and KL of our proposed TraficFlowGAN and the baselines.

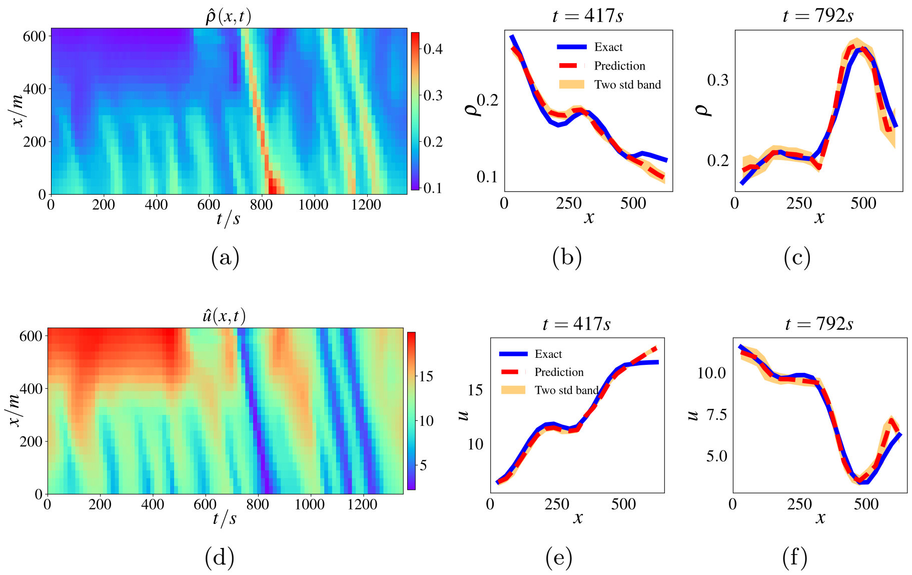


Fig. 8: Predictions of the traffic density (top row) and the traffic velocity (bottom row) of the TrafficFlowGAN.

Fig. 9 presents the comparison between the ground-truth traffic density distribution and that predicted by the TrafficFlowGAN model, each subfigure for a randomly sampled spatio-temporal coordinates. Most parts of the predicted and ground-truth distributions overlap with each other, which demonstrates that our proposed model can estimate the real-world traffic states uncertainties well.

Fig. 10 shows the learned traffic density-velocity relation by TrafficFlowGAN. When the number of loop detectors is 4, TrafficFlowGAN can already capture this relation well. Increasing the number of loop detectors helps TrafficFlowGAN learn a subtler pattern.

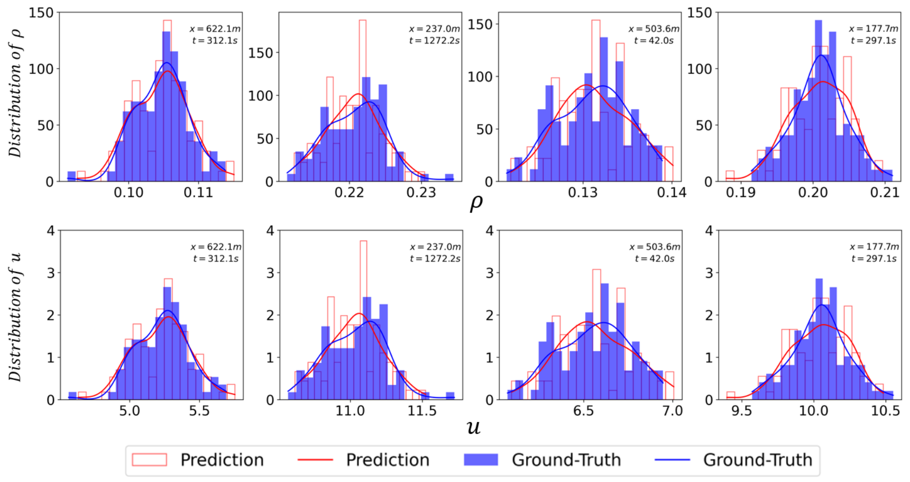


Fig. 9: Prediction distributions of TrafficFlowGAN for the NGSIM dataset.

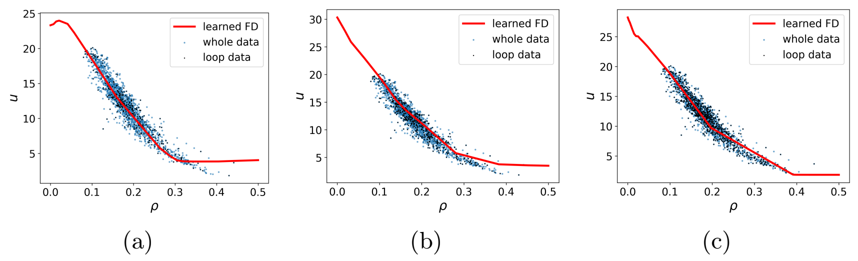


Fig. 10: Traffic density-velocity relations learned by the surrogate model (s-net) for different loop detector numbers. (a) loop detector number is 4. (b) loop detector number is 6. (c) loop detector number is 10.

6 Conclusions

This paper proposes TrafficFlowGAN to quantify the uncertainty in dynamical systems. TrafficFlowGAN leverages MLE, adversarial training, and PIDL to generate high-quality samples with exact data likelihood and efficient data usage. To verify that TrafficFlowGAN can learn the solutions of a second-order PDE, we conduct a numerical experiment where data is generated from a known ARZ equation. Numerical results show that TrafficFlowGAN manages to reconstruct the PDE solutions and recover the underlying relation between state variables. We further apply TrafficFlowGAN to a TSE problem using real-world data to demonstrate its performance. Results show that TrafficFlowGAN can better capture the real-world uncertainty than baselines, including the pure flow, the physics-informed flow, and the flow based GAN. We also show that TrafficFlow-GAN can learn the real-world traffic density-velocity relation simultaneously.

This work can be further improved in two directions. First, apart from the weighted sum, other approaches to integrating the likelihood loss, adversarial loss, and physics loss can be proposed. Second, TrafficFlowGAN needs to be re-trained if applied to other roads or to the same road but within a new time slot. We will work on the generalizability of TrafficFlowGAN in the future.

Acknowledgements

This work is sponsored by NSF under CPS-2038984.

References

1. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein generative adversarial networks. In: International conference on machine learning. pp. 214–223. PMLR (2017)
2. Aw, A., Rascle, M.: Resurrection of "second order" models of traffic flow. SIAM Journal on Applied Mathematics **60**(3), 916–938 (2000)
3. Brehmer, J., Cranmer, K.: Flows for simultaneous manifold learning and density estimation. Advances in Neural Information Processing Systems **33**, 442–453 (2020)
4. Council, N.R., et al.: Assessing the reliability of complex models: mathematical and statistical foundations of verification, validation, and uncertainty quantification. National Academies Press (2012)
5. Daw, A., Maruf, M., Karpatne, A.: Pid-gan: A gan framework based on a physics-informed discriminator for uncertainty quantification with physics. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. pp. 237–247 (2021)
6. Dinh, L., Sohl-Dickstein, J., Bengio, S.: Density estimation using real nvp. arXiv preprint arXiv:1605.08803 (2016)
7. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. Advances in neural information processing systems **27** (2014)
8. Grover, A., Chute, C., Shu, R., Cao, Z., Ermon, S.: Alignflow: Cycle consistent learning from multiple domains via normalizing flows. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 4028–4035 (2020)
9. Grover, A., Dhar, M., Ermon, S.: Flow-gan: Combining maximum likelihood and adversarial learning in generative models. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32 (2018)
10. Guo, L., Wu, H., Zhou, T.: Normalizing field flows: Solving forward and inverse stochastic differential equations using physics-informed flow models. arXiv preprint arXiv:2108.12956 (2021)
11. Lighthill, M.J., Whitham, G.B.: On kinematic waves II. A theory of traffic flow on long crowded roads. Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences **229**(1178), 317–345 (1955)
12. Raissi, M.: Deep hidden physics models: Deep learning of nonlinear partial differential equations. Journal of Machine Learning Research **19**(1), 932–955 (Jan 2018)

13. Seo, T., Bayen, A.M.: Traffic state estimation method with efficient data fusion based on the aw-rascle-zhang model. In: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC). pp. 1–6. IEEE (2017)
14. Shi, R., Mo, Z., Huang, K., Di, X., Du, Q.: A physics-informed deep learning paradigm for traffic state and fundamental diagram estimation. IEEE Transactions on Intelligent Transportation Systems (2021)
15. Siddani, B., Balachandar, S., Moore, W.C., Yang, Y., Fang, R.: Machine learning for physics-informed generation of dispersed multiphase flow using generative adversarial networks. Theoretical and Computational Fluid Dynamics **35**(6), 807–830 (2021)
16. Theis, L., Oord, A.v.d., Bethge, M.: A note on the evaluation of generative models. arXiv preprint arXiv:1511.01844 (2015)
17. Yang, L., Zhang, D., Karniadakis, G.E.: Physics-informed generative adversarial networks for stochastic differential equations. SIAM Journal on Scientific Computing **42**(1), A292–A317 (2020)
18. Yang, Y., Perdikaris, P.: Adversarial uncertainty quantification in physics-informed neural networks. Journal of Computational Physics **394**, 136–152 (2019)
19. Zang, C., Wang, F.: Moflow: an invertible flow model for generating molecular graphs. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (2020)