Mathematics of Control, Signals, and Systems (2022) 34:185–214
https://doi.org/10.1007/s00498-021-00309-8

ORIGINAL ARTICLE

Parameter calibration with stochastic gradient descent for interacting particle systems driven by neural networks

Simone Göttlich[1] · Claudia Totzeck[1]

Received: 29 January 2021 / Accepted: 4 October 2021 / Published online: 27 October 2021

Abstract
We propose a neural network approach to model general interaction dynamics and an adjoint-based stochastic gradient descent algorithm to calibrate its parameters. The parameter calibration problem is considered as optimal control problem that is investigated from a theoretical and numerical point of view. We prove the existence of optimal controls, derive the corresponding first-order optimality system and formulate a stochastic gradient descent algorithm to identify parameters for given data sets. To validate the approach, we use real data sets from traffic and crowd dynamics to fit the parameters. The results are compared to forces corresponding to well-known interaction models such as the Lighthill–Whitham–Richards model for traffic and the social force model for crowd motion.

Keywords Optimal control · Neural networks · Parameter identification · Data analysis

AMS Classification 34H05 · 92B20 · 82C32

1 Introduction

In the recent years, many models for interaction dynamics with various applications such as swarming, sheep and dogs, crowd motion, traffic and opinion dynamics have been proposed, see, e.g. [1,3–5,8,9,21,26,27] for an overview. Typically, the models are based on ordinary differential equations (ODEs) which describe the dynamics of each particle (or agent) in the system by interaction forces. A common approach is

Published in the topical collection *Machine Learning for Control Systems and Optimal Control*.

✉ Simone Göttlich
goettlich@uni-mannheim.de

Claudia Totzeck
totzeck@uni-mannheim.de

1 University of Mannheim, Mannheim, Germany

to model forces that replicate observations made in nature. For example in swarming, there exists the well-known three-phase model that consists of a short-range repulsion, a neutral/comfort zone and a third zone with attraction for long-range interactions [5]. The ranges can be adjusted with the help of parameters that need to be fitted for every application. Certainly, this is a smart approach and leads to promising results. Nevertheless, it is interesting to overboard all those assumptions and instead use a neural network to model the interactions with parameters based on data. This is the path we will follow in this paper. Actually, a similar viewpoint was taken for example in [2,19,28] where interaction kernels were inferred with a least squares approach from given data. The theoretical basis was discussed in [2], where well-posedness under a certain coercivity condition was shown. Moreover, the convergence of the approach w.r.t. to the mean-field limit is discussed. In [28], the methodology was generalized and applications with complicated dynamics allowing for pattern formation such as clustering and milling were studied. The learned interaction kernels are compared to well-known physical models with the help of confusion matrices and pattern indicator scores.

Clearly, the idea of using neural networks as substitutes for static/dynamic models or observers is well-known, see e.g [6,14] in the general context of ODEs. Furthermore, in [23] neural networks are discussed as alternative to estimate friction in automotive brakes. See [20] for a survey of artificial neural networks in energy systems. Moreover, in [12] deep neural networks are used to approximate Lyapunov functionals for systems of ordinary differential equations and in [13] the same author proposes to store approximate Lyapunov functions in a deep neural network in order to overcome the curse of dimensionality.

We intend to follow a similar approach for interacting particle systems in this paper. We first propose a framework to model very general particle interactions with the help of neural networks. Then, we state the corresponding parameter calibration problem. It enables us to identify the parameters using techniques from optimal control. We prove the well-posedness of the identification problem and derive the corresponding first-order optimality conditions that are used to compute the gradient for the stochastic descent algorithm.

In addition to the neural network model, the proposed parameter calibration approach can be used for general interacting particle systems with explicitly given, differentiable interaction forces, for example in terms of gradients of a potential. For our experiments, we apply the stochastic descent algorithm to real data sets from traffic and crowd experiments [18,24] to fit the neural network parameters. For comparison, we use the same algorithm to fit as well the parameters of well-known interaction models, such as the Lighthill–Whitham–Richards (LWR) model for traffic [17] and the social force model for crowd dynamics [15]. The two applications discussed here are prototypical examples for a first-order approach in 1d and second-order approach in 2d, respectively. Moreover, in both cases real data are available for our calibration purposes.

Similar parameter identification studies for pedestrian models have been recently introduced in [7,10] using a Bayesian probabilistic method and in [11,25] using neural networks. In contrast to [11,25], where the pedestrian speed or unknown interaction forces have been estimated, we address a more general setting that also allows for theoretical investigations and a rigorous numerical treatment. More precisely, for the neural networks we make some basic assumptions, but it is important to note that we do not prescribe any physical interaction assumptions. Using techniques from optimal control, we derive and analyse a parameter identification procedure that is based on stochastic gradient descent in Sect. 2. In Sect. 3, we begin with well-posedness results concerning a general interacting particle system that is driven by an artificial neural network. Then, we prove the existence of an optimal control for the parameter identification problem. Moreover, in Sect. 4, we derive the first-order optimality system for the identification problem in order to state the corresponding algorithm, an adjoint-based stochastic gradient descent method. A proof of existence of the adjoint concludes the theory part.

To validate our approach, we present an extended parameter estimation study for the traffic and the pedestrian model in Sects. 5 and 6, respectively. We apply the algorithm to a traffic scenario and estimate the interaction force as well as the speed of cars. The second application is based on pedestrian data, here again we train the artificial neural network with the help of our algorithm and compare the results to optimal parameters resulting from interactions that involve the social model for pedestrian interaction. The numerical results for both applications offer interesting insights and give rise for future considerations.

2 Optimal Control problem

The parameter identification is cast as optimal control problem. Let u denote the control variables, or, in terms of the application, the parameters to be identified and z the reference data set. We denote the space of controls by $\mathcal{U} = \mathbb{R}^K$. Then, we are interested in

$$\min_{u \in \mathcal{U}} J(x(u); z)$$

for some tailored cost function J. As the parameters enter the cost function implicitly through the state x, we propose to compute the gradient of the cost functional used for a stochastic gradient decent method with the help of an adjoint-based approach.

From now on, it is clear that x depends on the parameters u, so for notational convenience we drop the dependence in some equations. Let us assume that for each cost evaluation, we consider N agents with corresponding trajectories $z_i\colon [0, T] \to \mathbb{R}^d$ for $i = 1, \dots, N$. Based on the applications we have in mind, we obtain parameter-dependent trajectories $x_i^u\colon [0, T] \to \mathbb{R}^d$ for $i = 1, \dots, N$. We focus on the cost functional

$$J(x(u); z) = \frac{1}{2} \int_0^T \|z(t) - x(t)\|^2 \, dt = \frac{1}{2} \|z - x(u)\|^2_{L^2(0,T)} .$$

The state $x = x(u)$ is a solution to an ODE system that is driven by a feed-forward artificial neural network (NN) as follows

$$\frac{d}{dt}x_i = \sum_{j=1}^{N} W_u^{i,j}(x_j - x_i), \quad x_i(0) = z_0^i, \quad i = 1, \dots, N, \tag{1}$$

where W_u models the interaction of the agents and is the output of a neural network parametrized by u.

Remark 1 Throughout the work, we assume that every agent is driven by the same artificial neural network. We need the index $W^{i,j}$ only for the applications. For example in the traffic dynamic, the cars only interact with the car in front. This leads to $W^{i,j}$ being nonzero only for $j = i + 1$. In contrast, the car in front drives with fixed velocity, this can be represented with fixed weights, not involved in the parameter identification. Moreover, pedestrians interact with every other person which yields $W^{i,j} = W$; indeed, we can use the same artificial network to model the interaction forces for all the pedestrians.

To summarize, the optimal control problem we propose for the parameter identification driven by neural networks reads

Problem 1 *Find $\bar{u} \in \mathcal{U}_{ad} := [-1, 1]^K$ such that*

$$J(x(\bar{u}), z) = \min_{u \in \mathcal{U}} J(x(u), z) \quad \textit{subject to (1).}$$

Remark 2 We emphasize that in contrast to [11] the cost functional is not of the usual structure given by

$$\frac{1}{m}\left(\sum_{i=1}^{m} J(h_u(x_i), y_i)\right),$$

where $h_u(x_i)$ denotes the output of a neural network defined by the parameters u. Indeed, in the present article, the trajectories $z_i(u), i = 1, \dots, N$ are not the output of the neural network, but solutions of ODEs that are driven by the neural network.

As mentioned above, we consider feed-forward artificial neural networks. For later reference, we define these as follows.

Definition 1 A *feed-forward artificial neural network (NN)* is characterized by the following structure:

– Input layer:

$$a_1^{(1)} = 1, \quad a_k^{(1)} = x_{k-1}, \quad \text{for } k \in \{2, \dots, n^{(1)} + 1\},$$

where $x \in \mathbb{R}^{n^{(1)}}$ is the input (feature) and $n^{(1)})$ is the number of neurons without the bias unit $a_1^{(1)}$.

– Hidden layers:

$$a_1^{(\ell)} = 1, \quad a_k^{(\ell)} = g^{(\ell)} \left(\sum_{j=1}^{n^{(\ell-1)}+1} u_{j,k}^{(\ell-1)} a_j^{(\ell-1)} \right)$$

for $\ell \in \{2, \dots, L-1\}$ and $k \in \{2, \dots, n^{(\ell)} + 1\}$.

– Output layer:

$$a_k^{(L)} = g^{(L)} \left(\sum_{j=1}^{n^{(L-1)}+1} u_{j,k}^{(L-1)} a_j^{(L-1)} \right)$$

for $k \in \{1, \dots, n^{(L)}\}$

Note that the output layer has no bias unit. The entry $u_{j,k}^{\ell}$ of the weight matrix $u^{(\ell)} \in \mathbb{R}^{n^{(\ell-1)} \times n^{(\ell)}}$ describes the weight from neuron $a_j^{(\ell-1)}$ to the neuron $a_k^{(\ell)}$. For notational convenience, we assemble all entries $u_{j,k}^{(\ell)}$ in a vector $\mathbb{R}^K$ with

$$K := n^{(1)} \cdot n^{(2)} + n^{(2)} \cdot n^{(3)} + \cdots + n^{(L-1)} \cdot n^{(L)}.$$

For the numerical experiment, we use $g^{(\ell)}(x) = \log(1 + e^x)$ for $\ell = 2, \dots, N-1$ and $g^{(L)}(x) = x$. An illustration of an NN with $L = 3$, four inputs and six units in the hidden layer can be found in Fig. 1.

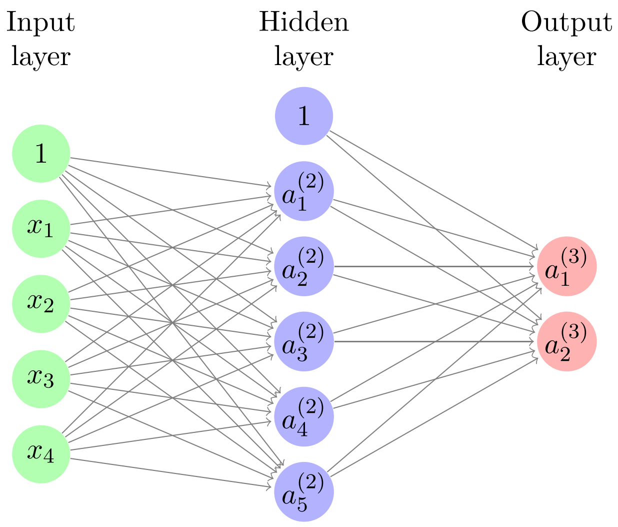

Fig. 1 Illustration of a feed-forward artificial network with 4 inputs and one bias input, one hidden layer with one bias unit and 5 neurons and two outputs. This corresponds to $n^{(1)} = 4, n^{(2)} = 5, n^{(3)} = 2, L = 3$

3 Analysis of the optimal control problem

In this section, we analyse the parameter estimation problem in terms of well-posedness. We begin with the state system and discuss then the well-posedness of the parameter identification problem. Finally, we show the existence of an adjoint state.

3.1 Well-posedness of the ODE system

Under some assumption on the activation functions $g^{(\ell)}$, $\ell = 2, \dots, L$ we can establish a well-posedness result for system (1). As the output of the neural network is the right-hand side of the ODE, we assume $n^{(L)} = d$ for the following considerations.

Theorem 1 *(Well-posedness of the state equation) Let the activation functions, $g^{(\ell)}$ of the* NN *defined in Definition 1 with $n^{(L)} = d$ be globally Lipschitz for $\ell = 2, \dots, L$. Then, there exists a unique solution $x \in \mathcal{C}^1([0,T], \mathbb{R}^d)$ to (1).*

Proof The proof exploits the recursive structure of the neural network defined in Definition 1. In fact, let $x \in \mathbb{R}^{n^{(1)}}$ and $a^{(L)} \in \mathbb{R}^d$ then

$$\begin{aligned} a_k^{(L)}(x) &= g^{(L)}\left(\sum_{j=1}^{n^{(L-1)}+1} u_{j,k}^{(L-1)} a_j^{(L-1)}(x)\right), \\ a_j^{(L-1)}(x) &= g^{(L-1)}\left(\sum_{m=1}^{n^{(L-2)}+1} u_{m,j}^{(L-2)} a_m^{(L-2)}(x)\right), \\ &\dots, \\ a_m^{(1)} &= x_{m-1}. \end{aligned}$$

Note that all relations are linear except for the activation functions $g^{(\ell)}$, $\ell = 2, \dots, L$. Using the globally Lipschitz assumption on g and the aforementioned linearity, we obtain

$$|a_k^{(L)}(x) - a_k^{(L)}(y)| \le C_g |x - y|,$$

where the constant C_g depends on all the Lipschitz constants of $g^{(\ell)}$, $\ell = 2, \dots, L$. This allows us to conclude the global Lipschitz property of the right-hand side of the ODE. An application of the Picard–Lindelöf theorem yields the well-posedness of the ODE as desired. □

Remark 3 The SmoothReLU activation function given by $g(x) = \ln(1 + e^x)$ is one example satisfying the assumptions of Theorem 1. Another suitable activation function is the identity $g(x) = x$.

The analysis of the optimal control problem is established in a Hilbert space framework. That is why we use the embedding $H^1([0,T], \mathbb{R}^{dN}) \hookrightarrow \mathcal{C}([0,T], \mathbb{R}^{dN})$ and define the control to state map

$$\mathcal{S}\colon \mathcal{U} \to H^1([0,T],\mathbb{R}^{dN}), \qquad \mathcal{S}\colon u \mapsto x(u),$$

which is well-defined thanks to Theorem 1. Moreover, we define the reduced cost functional

$$\hat{J}(u) := J(\mathcal{S}(u); z).$$

The next theorem provides us the continuous dependence on the data for the ODE solution. It will help us in the proof of the existence of a minimizer below (Theorem 3).

Theorem 2 (Continuous dependence on the data) *Let the assumptions of Theorem 1 hold and, additionally, let $\mathcal{U}$ be bounded and $g^{(\ell)}$ be bounded for $\ell = 2, \dots, L-1$. Then, the solution to (1) depends continuously on the data, i.e.*

$$\|x - \bar{x}\|_{H^1([0,T],\mathbb{R}^d)} \le M\left(\|x_0 - \bar{x}_0\| + \|u - \bar{u}\|_{\mathcal{U}}\right).$$

Proof Let us consider the i-th component of the difference of two solutions corresponding to different data u and $\bar{u}$ given by

$$\begin{aligned} x_i(t) - \bar{x}_i(t) \ &= (x_0 - \bar{x}_0)_i + \int_0^t \sum_{j=1}^N W_u^{i,j}(x_j(s) - x_i(s)) \\ &\quad - \sum_{j=1}^N W_{\bar{u}}^{i,j}(\bar{x}_j(s) - \bar{x}_i(s))ds. \end{aligned}$$

We therefore need to estimate the m-th entry of the difference of the interaction forces. Let $m \in \{1, \dots, d\}$ be arbitrary, we find

$$\begin{aligned} &|\left(W_u^{i,j}(x_j - x_i) - W_{\bar{u}}^{i,j}(\bar{x}_j - \bar{x}_i)\right)_m| \\ &= |g^{(L)}\left(\sum_{k=1}^{n^{(L-1)}+1} u_{k,m} a_k^{(L-1)}(x_j - x_i)\right) - g^{(L)}\left(\sum_{k=1}^{n^{(L-1)}+1} \bar{u}_{k,m} a_k^{(L-1)}(\bar{x}_j - \bar{x}_i)\right)| \\ &\le L_{g^{(L)}} \sum_{k=1}^{n^{(L-1)}+1} |u_{k,m}||a_k^{(L-1)}(x_j - x_i) - a_k^{(L-1)}(\bar{x}_j - \bar{x}_i)| + |u_{k,m} - \bar{u}_{k,m}||a_k^{(L-1)}(\bar{x}_j - \bar{x}_i)| \\ &\le L_{g^{(L)}} \sum_{k=1}^{n^{(L-1)}+1} M_u |a_k^{(L-1)}(x_j - x_i) - a_k^{(L-1)}(\bar{x}_j - \bar{x}_i)| + M_{g^{(L-1)}}|u_{k,m} - \bar{u}_{k,m}|, \end{aligned}$$

where M_u denotes the upper bound of $\mathcal{U}$, $M_{g^{(\ell-1)}}$ denotes the bound for the activation functions $g^{(\ell)}$ with $\ell = 2, \dots, L-1$ and $L_{g^{(\ell)}}$ denotes the global Lipschitz constants of the activation functions.

The last step of the previous estimate can be recursively applied to

$$|a_k^{(\ell-1)}(x_j - x_i) - a_k^{(\ell-1)}(\bar{x}_j - \bar{x}_i)|$$
$$\leq \sum_{p=1}^{n^{(\ell-2)}+1} u_{p,k}^{(\ell-2)} a_p^{(\ell-2)}(x_j - x_i) - \bar{u}_{p,k}^{(\ell-2)} a_p^{(\ell-2)}(\bar{x}_j - \bar{x}_i), \quad \ell = 3, \dots, L-1.$$

For the first layer, we have

$$|a_k^{(1)}(x_j - x_i) - a_k^{(1)}(\bar{x}_j - \bar{x}_i)| \leq |(x_j - \bar{x}_j)_k| + |(x_i - \bar{x}_i)_k|.$$

The two estimates together yield

$$|\left(W_u^{i,j}(x_j - x_i) - W_{\bar{u}}^{i,j}(\bar{x}_j - \bar{x}_i)\right)_m| \leq C_1 \|u - \bar{u}\| + C_2 \|x - \bar{x}\|. \tag{2}$$

This implies

$$\|x(t) - \bar{x}(t)\| = \sum_{i=1} |x_i(t) - \bar{x}_i(t)| \leq \|x_0 - \bar{x}_0\| + C_3 \|u - \bar{u}\|$$
$$+ C_4 \int_0^t \|x(s) - \bar{x}(s)\| ds.$$

An application of Gronwall's theorem gives us

$$\|x(t) - \bar{x}(t)\| \leq C\left(\|x_0 - \bar{x}_0\| + \|u - \bar{u}\|\right) e^{C_4 t}. \tag{3}$$

Using (2) and (3), we obtain for the time derivative

$$\|\frac{d}{dt}x_i(t) - \frac{d}{dt}\bar{x}_i(t)\| \leq \sum_j \|W_u^{i,j}(x_j - x_i) - W_{\bar{u}}^{i,j}(\bar{x}_j - \bar{x}_i)\|$$
$$\leq C_5 \sum_j \left(\|x_0 - \bar{x}_0\| + \|u - \bar{u}\|\right) e^{C_4 t}.$$

Summing over $i = 1, \dots, N$, we get the bound

$$\|\frac{d}{dt}x(t) - \frac{d}{dt}\bar{x}(t)\| \leq C\left(\|x_0 - \bar{x}_0\| + \|u - \bar{u}\|\right) e^{C_4 t}. \tag{4}$$

Combining the estimates in (3) and (4) leads to the desired result. □

In the following, we are concerned with the existence of a minimizer to the control problem and the existence of an adjoint state. Latter will help us to state the first-order optimality conditions and to compute the gradient for the descent algorithm. For notational convenience, we define the operator

$$e(x,u)\colon H^1([0,T],\mathbb{R}^{dN})\times\mathcal{U}\to Z,\qquad e(x,u)=\Big(e_i(x,u)\Big)_{i=1,\dots,N}$$

with

$$e_i(x,u)=\frac{d}{dt}x_i-\sum_{j=1}^{N}W_u^{i,j}(x_j-x_i).$$

Theorem 3 *(Existence of a minimizer) Let $W_u^{i,j}$ be weakly continuous for all $i,j=1,\dots,N$ and $g^{(L)}(0)=0$. Then, there exists a minimizer $u^*\in\mathcal{U}_{ad}$ for Problem 1.*

Proof Equipped with the results of Theorem 1 and Theorem 2, the proof can be established by standard arguments. See Theorem 1.45 in [16] for the main ideas. □

Remark 4 Note that the minimizer of the parameter identification problem may not be unique due to the nonlinearity in the state problem.

Remark 5 Again, the SmoothReLU activation functions satisfy the requirements of the existence result in Theorem 3. Indeed, $g(x)=\ln(1+e^x)$ and $g(x)=x$ have uniformly bounded derivatives. Hence, we apply the mean-value theorem to obtain

$$\begin{aligned}&\int_0^T(g((x_{n_j}-x_{n_i})_k)-g((x_j-x_i)_k))\phi(t)dt\\&=\int_0^T g'(\xi)(x_{n_j}-x_j+x_{n_i}-x_i)_k\phi(t)dt\longrightarrow 0\end{aligned}$$

as x_n converges weakly to x for some $\xi\in[(x_{n_j}-x_{n_i})_k,(x_j-x_i)_k]$. Moreover, choosing $g^{(L)}$ to be the identity leads to $g^{(L)}(0)=0$.

Theorem 4 (Existence of adjoint state) *Let $g^{(\ell)}\in\mathcal{C}^1$ with $(g^{(\ell)})'\in Lip_{loc}(\mathbb{R})$ globally bounded and let $(\bar{x},\bar{u})$ an optimal solution of Problem 1. Then, there exists an adjoint state, $\bar{p}$, such that the following optimality condition holds:*

$$\frac{d}{dt}\bar{x}_i=\sum_{j=1}^{N}W_{\bar{u}}^{i,j}(\bar{x}_j-\bar{x}_i),\quad \bar{x}_i(0)=z_0^i,\quad i=1,\dots,N,$$

$$\begin{aligned}\int_0^T\left(\frac{d}{dt}h_i+\sum_{i=1}^{N}d_{x_i}W_{\bar{u}}^{i,j}(\bar{x}_j-\bar{x}_i)h_i-d_{x_j}W_{\bar{u}}^{i,j}(\bar{x}_j-\bar{x}_i)h_j\right)\cdot\bar{p}_i\,dt+h_i(0)\cdot\eta\\=\int_0^T(\bar{x}_i(\bar{u})-z_i)\cdot h_i\,dt,\qquad\forall\,h_i\in H^1((0,T),\mathbb{R}^d),\ \eta\in\mathbb{R}^d,\end{aligned}$$

$$\int_0^T\sum_{i=1}^{N}\sum_{j=1}^{N}\nabla_u W_{\bar{u}}^{i,j}(\bar{x}_j-\bar{x}_i)\cdot\bar{p}_i\,dt\cdot(\bar{u}-u)\geq 0\ \ \textit{for all}\ \ u\in U_{ad}.$$

Proof We aim to apply Corollary 1.3 in [16] and check its four requirements.

We begin with the set of admissible controls. $\mathcal{U}_{\text{ad}} = [-1, 1]^K \subset U = \mathbb{R}^K$ is nonempty, convex and closed. This verifies the first requirement.

Next, we have to show that $J: Y \times \mathcal{U} \to \mathbb{R}$ and $e: Y \times \mathcal{U} \to Z$ are continuous Fréchet differentiable. We set $\mathcal{U} = \mathbb{R}^K, Y = H^1((0,T), \mathbb{R}^d)$ and $Z = L^2((0,T), \mathbb{R}^d)$. Note that these are all Banach spaces. The cost functional J is Fréchet differentiable by standard arguments, see for example [16], it holds

$$d_y J(y(u), u)[h] = \int_0^T (x(u) - z_i) \cdot h \, dt, \qquad d_u J(y(u), u)[h] = 0.$$

For the state operator e, we find

$$\begin{aligned} d_y e_i(y,u)[h] &= \frac{d}{dt} h_i + \sum_{j=1}^N d_{x_i} W_u^{i,j}(x_j - x_i) h_i - d_{x_j} W_u^{i,j}(x_j - x_i) h_j, \\ &\quad i = 1, \dots, N, \\ d_u e_i(y,u)[k] &= \int_0^T \sum_{i=1}^N \sum_{j=1}^N d_u W_u^{i,j}(x_j - x_i)[k] \cdot \bar{p}_i \, dt. \end{aligned}$$

Using the assumptions on the activation function $g^{(\ell)}$, this yields $e_y(y,u) \in \mathcal{L}(Y, Z)$.

Third, Theorem 1 assures that $e(y,u) = 0$ admits a unique solution

$$y = y(u) \in \mathcal{C}^1((0,T), \mathbb{R}^d) \subset H^1((0,T), \mathbb{R}^d).$$

We are left to show that $e_y(y(u), u) \in \mathcal{L}(Y, Z)$ has a bounded inverse for all $u \in V$ in a neighbourhood of $\mathcal{U}_{\text{ad}}$. In order to see that we consider

$$e_y(y(u), u)[h] = r$$

for arbitrary $r \in Z = L^2((0,T), \mathbb{R}^d)$. By the Caratheodory theory for ODEs, there exists a unique solution to this equation for every $r \in Z$. Using the explicit expression of $e_y(y(u), u)$, we find

$$\|h(t)\| = \|h(0)\| + \int_0^T \|r(s)\| ds + C_{DW} \int_0^T \|h(s)\| \quad \text{for all } t \in (0,T)$$

for some constant $C_{DW} > 0$ depending on the global bounds on the derivatives of the activation functions. An application of Gronwall's Lemma yields the boundedness of the inverse of $e_y(y(u), u)$.

Hence, the requirements of Corollary 1.3 in [16] are satisfied and we find an adjoint state $\bar{p}$ such that the optimality condition holds as desired. □

These results assure the well-posedness of the optimization problem and its first-order optimality system. We are well-equipped to state the stochastic gradient descent algorithm that we propose for the treatment of general neural network-driven optimization problems. The numerical results below are computed with this algorithm as well.

4 Stochastic gradient descent algorithm

Having the results of the previous section at hand, we can now propose the stochastic gradient descent algorithm. In this section, we assume that $p \in H^1((0,T),\mathbb{R}^d) \subset Z$. This allows us to formulate the adjoint system in strong form given by

$$-\frac{d}{dt}\bar{p}_i = \sum_{j=1}^N \nabla_{x_i} W_{\bar{u}}^{i,j}(\bar{x}_j - \bar{x}_i)(\bar{p}_i - \bar{p}_j) - (\bar{x}_i - z_i), \qquad i = 1,\dots,N,$$

supplemented with the terminal condition $p(T) = 0$, where $\bar{x}$ is a solution to (1) with initial condition $x_0 = z_0$. The gradient is based on this strong form of the adjoint.

4.1 Gradient of the reduced cost functional

We compute the gradient of the reduced cost functional as follows

$$\langle \hat{J}'(u), s\rangle_{\mathbb{R}^K} = \langle z - \mathcal{S}(u), \mathcal{S}'(u)[s]\rangle_{H^1,H^{-1}} = \langle \mathcal{S}'(u)^*(z - \mathcal{S}(u)), s\rangle_{\mathbb{R}^K}.$$

Let e be the operator defined above, we obtain

$$S'(u)^*(z - \mathcal{S}(u)) = -e_u(\mathcal{S}(u),u)^* e_y(\mathcal{S}(u),u)^{-*}(z - \mathcal{S}(u)) = e_u(\mathcal{S}(u),u)^* p,$$

where we used the adjoint equation $e_y(\mathcal{S}(u),u)^* p = -(z - \mathcal{S}(u))$. Altogether, the gradient of the reduced cost functional can be expressed as

$$\hat{J}'(u) = e_u(\mathcal{S}(u),u)^* p(u) = \int_0^T \sum_{i=1}^N \sum_{j=1}^N \nabla_u W_{\bar{u}}^{i,j}(\bar{x}_j - \bar{x}_i)\cdot \bar{p}_i \, dt.$$

Based on these considerations, we may establish a gradient descent algorithm. Moreover, for $\mathcal{U}_{\text{ad}}$ bounded, we may employ a projected gradient descent method. Note that in our application we face big data sets and therefore, the evaluation of the full cost function is very costly. This is why we use a mini-batch algorithm. Its details are discussed in the following section.

Remark 6 Note that we exploited the recursive structure of the artificial neural network in the previous derivations. The approach can be generalized to a wider class of neural networks, in fact, all neural networks that allow for backpropagation can be used. Certainly, the computation of $\nabla_x W_u^{i,j}$ and $\nabla_u W_u^{i,j}$ can be very complicated in general.

4.2 Stochastic descent

In many applications, we expect the cost functions to have several local minima, where usual gradient descent algorithms may get stuck. To prevent this issue when training the neural networks, we use a mini-batch gradient descent scheme. Indeed, we compute ADADELTA updates as proposed in [11]. This leads to the following algorithm:

Let α_k denote the k-th iterate of the gradient descent. We define

$$\begin{aligned}
\Delta\alpha_k &= \alpha_{k+1} - \alpha_k, \qquad E[g^2]_0 = 0, \qquad E[\Delta\alpha^2]_0 = 0,\\
E[g^2]_k &= \rho E[g^2]_{k-1} + (1-\rho)(\nabla J_{k-1}^{\tilde{m}}(\alpha_{k-1}))^2,\\
\Delta\alpha_k &= -\frac{\sqrt{E[\Delta\alpha^2]_{k-1} + \epsilon}}{\sqrt{E[g^2]_k} + \epsilon}\nabla J_k^{\tilde{m}}(\alpha_k),\\
E[\Delta\alpha^2]_k &= \rho E[\Delta\alpha^2]_{l-1} + (1-\rho)\Delta\alpha_k^2,
\end{aligned}$$

where $\rho \in (0, 1)$ is the rate for the adaption of the squared gradient information and $\epsilon > 0$ avoids singular values by division. Both are fixed parameters.

Note that these updates are still deterministic. In order to incorporate noise that help us to escape from local minima, we add a multivariate normal distributed random vector N_k with $N_k \sim \mathcal{N}(0, \Sigma_k)$ to the gradient in each iteration. Here, Σ_k denotes the variance matrix. We choose

$$(\Sigma_k)_{ii} = \frac{\eta_1}{(1+k)^{\eta_2}}$$

for some constants $\eta_1, \eta_2 > 0$ and $(\Sigma_k)_{ij} = 0$ whenever $i \neq j$. Note that the noise diminishes as the number of iteration, k, increases. For the numerical simulations, we set $\eta_1 = 1$ and $\eta_2 = 0.55$, the adaption rate $\rho = 0.95$ and $\epsilon = 10^{-6}$.

5 Parameter estimation based on traffic data

The stochastic gradient descent method proposed above can be used to treat general neural network driven optimization or parameter identification problem. In the following, we apply it to two applications for which we have real data available. We begin with a one-dimensional traffic dynamic modelled with the help of a first-order ODE system, then we consider a two-dimensional pedestrian dynamic modelled with a second-order dynamic. Moreover, we use the stochastic gradient descent algorithm to estimate parameters of well-known interaction forces for the two scenarios and compare the output and the cost. We want to emphasize that we treat both approaches, the NN and the physically inspired models, as equally admissible. In particular, we do not choose one of them to be the ground truth.

Remark 7 We noticed in discussions that often physical models are accepted as ground truth without question. Even though these models are well-motivated, we cannot blindly assume they represent the whole truth.

5.1 Traffic models

For the traffic dynamic, we assume to have a *follower–leader* dynamic. Therefore, we choose two microscopic versions of the well-known LWR-model [17] with logarithmic and linear velocity function, respectively, as reference models.

5.1.1 Traffic dynamic with LWR-model

Let $x_i(t)$ denote the position of the i-th car at time $t \in [0, T]$. Then, the evolution of the cars is given by

$$\frac{d}{dt} x_i(t) = W_u^{i,i+1}\left(\frac{x_{i+1}(t) - x_i(t)}{L}\right), \quad i = 1, \dots, N-1, \tag{5a}$$

$$\frac{d}{dt} x_N(t) = v_0. \tag{5b}$$

The parameters are v_0 the velocity of the leading car and L the length of the cars. We consider $W_u^{i,i+1}(z) = v_0 \log(z)$ and $W_u^{i,i+1}(z) = v_0(1 - 1/z)$for $z > 0$. The task of the parameter identification is to estimate $u = (L, v_0) \in \mathbb{R}^2$.

5.1.2 Traffic dynamic with artificial neural network

The model driven by the feed-forward neural network is given as follows. We parametrize the interactions of the follower–leader dynamic by W_u, where $u = (v_0, u_{\text{Net}})$ and u_{Net} is assumed to contain all the information of the neural network. The dynamic is then given by

$$\frac{d}{dt} x_i(t) = W_u^{i,i+1}(x_{i+1}(t) - x_j(t)), \quad i = 1, \dots, N-1, \tag{6a}$$

$$\frac{d}{dt} x_N(t) = v_0. \tag{6b}$$

supplemented with initial data $x(0) = x_0$.

Remark 8 Note that we have to prescribe some value for the first vehicle even in this case, as we assume that the behaviour of cars depends on the distance to the vehicle in front and the first vehicle has no one in front.

5.2 Data processing of traffic data set

We use the microscopic traffic data set that was recorded within the project ESIMAS [18]. It contains vehicle data from 5 cameras that were placed in a $1km$ tunnel section on the German motorway A3 nearby Frankfurt / Main. For the parameter estimation, we extract sequences from the data that contain three or more vehicles in one lane. For simplicity, we restrict our considerations to the middle lane data and neglect the data of the $y-$coordinate. Thus, we have one-dimensional traffic data for the parameter estimation.

First, we interpolate the position data that is supplemented with time stamps to a reference time discretization. Having all the data aligned to the reference time discretization, we filter sequences of data where two or more vehicles are present in the camera frame. This yields a database with various sequences of different length and with different number of vehicles that we use for the parameter identification.

Note that after this data extraction, the vectors containing the positions of the cars for each sequence are ordered. In fact, we pass this ordered vector to the neural network and hold on to the assumption that the interaction of the vehicles depends on the distance to the vehicle in front. This is why we have to prescribe some velocity for the first car in the data set, even in the case of the neural network without physical parameters, see (6).

The ideas behind this data processing are the following. First, to solve the ODE it is convenient to have a fixed number of particle in the scene. We therefore cut the data into smaller time slices to ensure that the cars in the scene do not change within one sequence. Another argument in favour of the small time slices is that the prediction of the trajectories is more likely to be successful on short time intervals than predicting the whole journey of the particles. In fact, the ODE interaction models do react to the current situation and therefore the learning based on short time interval is justified. The preprocessed data with position and timestamp information are available online.[1]

Remark 9 Note that even though the parameter set gives data on the vehicle type, we do not use this information in the approach. Therefore, we expect to obtain an averaged length L as output of the parameter estimation for the LWR-model with logarithmic and linear velocity function.

5.3 Numerical schemes

For both approaches, the LWR and the NN model, we solve all the parameter identification problems with the stochastic gradient descent method discussed in Sect. 4.2. The forward and adjoint models are integrated using an Explicit Euler scheme. The traffic data set has time step $dt_{\text{data}} = 0.2$ for the simulation we use a finer discretization, i.e. $dt = 0.002$.

5.4 Numerical results

For the numerical results, we use the two data sets of "day 1" of all five cameras. We test three different neural network settings with two, four and ten neurons in the hidden layer and call them, N2, N4, N10, respectively. The results corresponding to the LWR-model with logarithmic velocity function are denoted by "Log", and the ones obtained with the linear velocity function are denoted by "Lin".

[1] https://github.com/ctotzeck/NN-interaction.

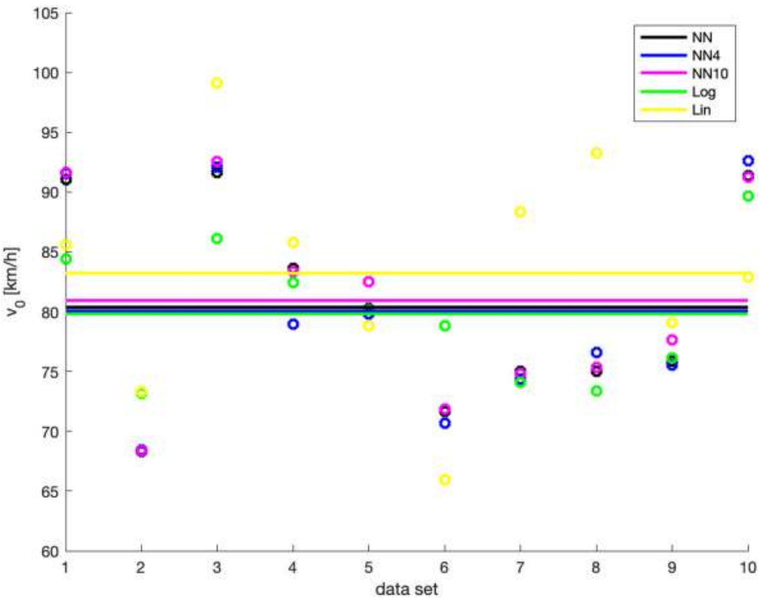

Fig. 2 The circles show the identified velocities of the leading car for the different methods (colour coded) and the different data sets. The lines show the mean velocities taken over all data sets for the different methods (Color figure online)

Table 1 Car lengths (in m) estimated with the algorithm for the 10 data sets with the LWR-model with linear and logarithmic velocity

	1	2	3	4	5	6	7	8	9	10	Average
Lin	3.47	3.95	5.64	4.50	2.25	2.54	8.33	7.07	2.00	7.16	4.69
Log	8.24	7.92	9.20	9.90	7.17	6.63	9.98	9.91	5.52	9.76	8.43

Note that the provided data units are meter m and seconds s. We initialize the velocity v_0 and car length L for the LWR cases with $v_0 = 30\frac{m}{s}$ and $L = 5m$. The same initial velocity is used for all neural network dynamics. The weights for the neural networks are initialized with random values uniformly distributed in $[-1, 1]$. We set a lower bound for the car length $L_{\min} = 2m$.

Figure 2 shows the results of the identification for the different models. The circles show the identified velocities of the leading car for the different methods and the different data sets. The lines show the mean velocities taken over all data sets for the different methods. The mean velocity of the LWR-model with linear velocity function is slightly larger than the other mean velocities which are close to $80\frac{km}{h}$. The maximal speed allowed on the highway is $100\frac{km}{h}$. We see differences between the models for single data sets. The car lengths identified for the two LWR approaches are shown in Table 1.

The average car length estimated with the linear velocity approach is $4.69m$, and the one for the logarithmic velocity function is $8.43m$. These numbers may indicate that the linear model estimates the true car length, while the logarithmic approach includes the distance to the next car into this value. For test case 9, the car length of the linear model hits the lower bound $L_{\min}$. Out of curiosity, we dropped the lower bound assumption in this case, leading to a value of $0.6m$ and an over all average velocity close to $80\frac{km}{h}$ similar to the other models.

Table 2 shows the cost for the different approaches using the interaction forces and parameters identified by the algorithm. In average, the neural network approach with four neurons in the hidden layer performs best. In fact, all the neural network approaches perform better than the LWR-model with linear or logarithmic velocity function. The least cost for each data set and for the average is highlighted. N4 has the least cost in 60% of the test cases. N10 approximates the third data set best, and Lin gives the best result for three of the data sets. The linear model is closer to the NN approaches and outperforms the logarithmic model. We can see that the smallest network with only two neurons in the hidden layer is not able to reproduce the interaction forces (see Table 2). On the other hand, the bigger the neural network, the more likely we run into problems of overfitting. This could be a reason why N4 outperforms N10. Nevertheless, this paper can only serve as a proof of concept of the methodology. An investigation of the robustness of the results with respect to noise in the data or the network architecture would be interesting future work.

Figure 3 illustrates the different interaction forces resulting from the parameter estimation. The interaction forces of the neural network models are rather linear except for the range $2m - 5m$. The interaction forces of the linear and the logarithmic LWR-model behave different in this region; in fact, they have very steep gradients and enter a negative regime, which corresponds to slowing down. For distances in the range of 15m-50m, the linear LWR-model is close to the interaction forces of the neural networks, whereas the logarithmic LWR-model admits larger values. Due to the lack of data in the short distance regime, we cannot expect the NN approaches to reproduce the behaviour of slowing down.

The learning process is visualized exemplarily for data set 4 in Fig. 4 which shows the norm of the gradient for different iterations of the learning process. The stochasticity of our algorithms is visible; at the first iterations, we see a high variance which is then diminishing when the learning process evolves. We choose a maximal number of 2500 iterations for the learning process.

6 Parameter estimation based on crowd data

The following parameter estimation is based on a data set provided by the Institute for Advanced Simulation: Civil Safety Research of Forschungszentrum Jülich [24]. In particular, the data set of a bidirectional flow in a corridor is used. We employ the stochastic gradient descent algorithm to fit the parameters of the neural network as well as the social force model.

Table 2 Evaluations of the cost functional with the interaction forces identified by the algorithm for the 10 data sets. The least cost value for each column is highlighted

	1	2	3	4	5	6	7	8	9	10	Average
N2	54.68	42.10	98.67	33.38	24.10	20.01	39.73	53.09	10.37	74.27	45.04
N4	54.25	**40.98**	108.80	**33.12**	**20.68**	**18.26**	40.43	**53.05**	**8.11**	72.02	**44.97**
N10	50.43	46.02	**94.69**	41.21	24.81	19.97	43.35	57.21	9.23	69.61	45.65
Lin	**49.11**	41.33	96.91	39.17	23.11	34.11	**39.39**	62.30	15.21	**63.78**	46.44
Log	58.88	50.04	113.69	71.31	32.40	59.59	39.68	57.72	19.84	89.24	59.24

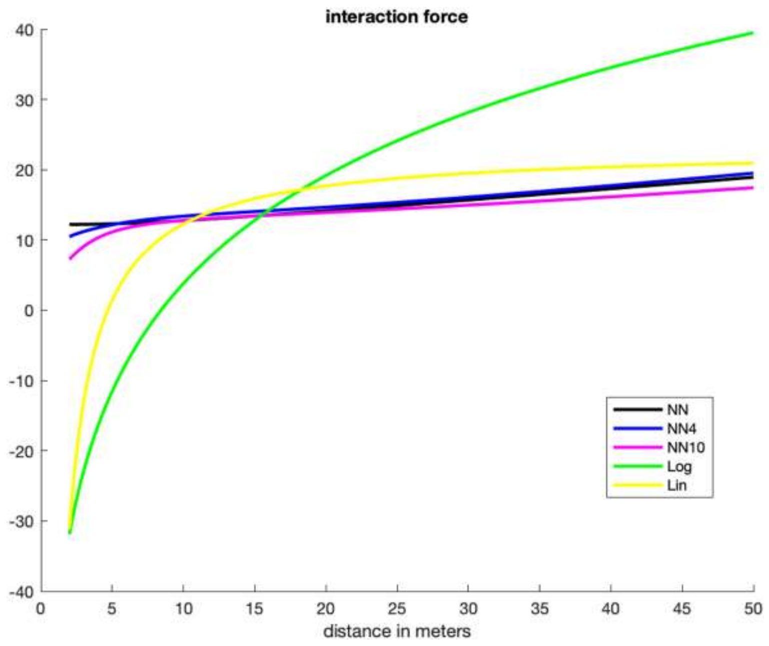

Fig. 3 Interaction forces resulting from the parameter identifications

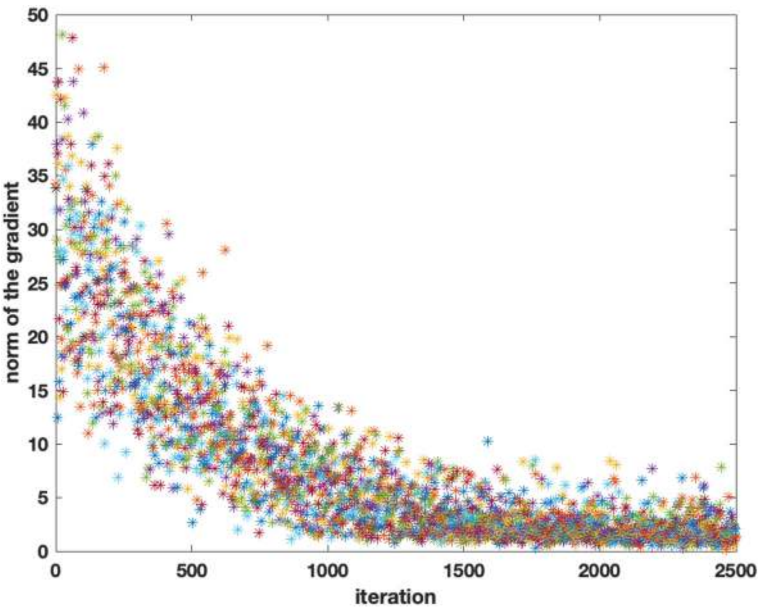

Fig. 4 Illustration of the learning process. At the first iterations, our learning algorithms has a high variance, and the variance diminishes as the learning process evolves. We stop at a maximal number of 2500 iterations

6.1 Crowd data with social force model

We assume that the social force model proposed by Helbing and Molnár [15] is a good fit to work with the crowd data set. It reads

$$\frac{d}{dt}x_i = v_i, \tag{7a}$$

$$\frac{d}{dt}v_i = v_i^D + \frac{1}{Nm}\sum_{j=1}^{N} F_{i,j} + \frac{1}{Km}\sum_{k=1}^{K} F_{iw} \tag{7b}$$

supplemented with initial data $x(0) = x_0$ and $v(0) = v_0$. The relevant parameters are given in Table 3.

The relaxation term towards the desired velocity v_i^{des} is constructed from the given trajectories as follows

$$v_i^D = \frac{1}{\tau}\big(v_i^{\text{des}}(t) - v_i(t)\big), \quad \text{where} \quad v_i^{\text{des}}(t) = \frac{x_D - x_i(t)}{\|x_D - x_i(t)\|}\|v_i(t)\|. \tag{8}$$

Here, we assume that each pedestrian tries to head towards his or her destination which is given by the last position of the data sequence and keeps the current speed. The other force terms are assumed to be given as

$$\begin{aligned} F_{ij} = F(2r - d_{i,j}, v_i - v_j) \; &= \left(A \exp\left(\frac{2r - d_{ij}}{B}\right) + k\, h(2r - d_{ij})\right) n_{ij} \\ &+ \kappa\, h(2r - d_{ij}) \Delta v_{ji}^t t_{ij} \end{aligned} \tag{9}$$

with

$$d_{ij} = \|x_i - x_j\|, \quad n_{ij} = \frac{x_i - x_j}{d_{ij}}, \quad t_{ij} = (-n_{ij}^2, \quad n_{ij}^1), \quad \Delta v_{ji}^t = (v_j - v_i)\cdot t_{ij}$$

Table 3 Parameters used for the parameter estimation based on crowd data. For the parameter identification, we fix values for m, r, τ, M, N, B and seek to find A, κ and k

Parameter	Variable
Mass	m
Radius	r
Relaxation time	τ
Force constant	A
Force constant	B
Force constant	κ
Force constant	k
Desired velocity of i-th pedestrian	v_i^{des}
Number of wall discretization points	M
Number of pedestrians	N

being the distance of pedestrian i and pedestrian j, the normalized vector pointing from pedestrian j to pedestrian i, the tangential direction and the tangential velocity difference, respectively, and, $h(y) = H(y)\,y$ with Heaviside function H. The interaction with the walls is parametrized using

$$\begin{aligned} F_{iw} = F(r - d_{iw}, v_i) \ &= \left(A \exp\left(\frac{r - d_{iw}}{B}\right) + k\, h(r - d_{iw}) \right) n_{iw} \\ &\quad + \kappa\, h(r - d_{iw})(v_i \cdot t_{iw}) t_{iw}. \end{aligned}$$

We assume here and in the following that the walls consist of stationary points, such that d_{iw}, n_{iw} and t_{iw} are easy to compute. We fix the radius $r = 0.25$, the scaling $B = 0.1$, the relaxation parameter $\tau = 0.5$ and the mass $m = 1$ for all simulations. To summarize, the relevant parameters to find via the estimation procedure are $u = (A, k, \kappa)$.

6.2 Crowd data with neural networks

As we aim to compare the results of the model-based approach to the neural network approach, we pass the same data to neural network that models the acceleration. Indeed, we assume to neural network dynamic to be given by

$$\frac{d}{dt} x_i = v_i, \tag{10a}$$

$$\frac{d}{dt} v_i = v_i^D + \frac{1}{mN} \sum_{j=1}^{N} W_u^{i,j}(x_i - x_j, v_i - v_j) + \frac{1}{m N_{\text{wall}}} \sum_{k=1}^{N_{\text{wall}}} W_u^{i,w}(x_i - x_k, v_i - v_k) \tag{10b}$$

supplemented with initial data $x(0) = x_0$ and $v(0) = v_0$. Here, we refer with x_k to the positions of the discretization points of the wall and with v_k to artificial velocity vectors of the wall points for $k = 1, \ldots, N_{\text{wall}}$. Using separate neural networks for the interaction and the walls allows us to compare the structure of the resulting terms one by one and to understand the two approaches in more detail. Note that we use the same neural network with different normal vectors for the different walls. As for the social force approach, we fix the relaxation parameter $\tau = 0.5$. The other parameters need to be estimated. For notational convenience, we define

$$u = (u_{\text{NN}}^{\text{int}}, u_{\text{NN}}^{\text{wall}}),$$

where $u_{\text{NN}}^{\text{int}}$ denotes all the parameters involved in the neural network modelling the pairwise interaction between pedestrians and $u_{\text{NN}}^{\text{wall}}$ the parameters of the neural network modelling the interaction of with the walls, respectively.

6.3 Processing of the crowd data set

The crowd data set is processed with the same approach as the traffic data sets. Indeed, we split it into sequences of fixed length T_{seq}. Then, we consider only the pedestrians that are present during the whole sequence. In fact, we obtain T/T_{seq} sequences with a different number of pedestrians present. This allows us to use a fixed number of pedestrians in the models for each computation of the gradient in the parameter estimation. We expect to have only small errors raised by the fact that some of the pedestrians are neglected.

After the data processing, we have the trajectories of all pedestrians present in each sequence. We use the first and the last point of these trajectories to compute the relaxation term (8) in each time step. Moreover, we compute the velocities of the pedestrians using a finite difference approximation. This applies to both, the social force and the NN approach.

6.4 Numerical schemes and parameters

We use the Explicit Euler scheme to solve the state system and the adjoint systems. The information is then passed to the stochastic descent algorithm, see Sect. 4.2.

Each sequence of the preprocessing involved 25 time steps of length $dt = 0.04$, leading to $T_{\text{seq}} = 1$. We use the same time step $dt = 0.04$ for the Euler method. The parameter of the stochastic descent algorithms is given in Sect. 4.2. The values for the initial neural networks are a random sample that is chosen independently and uniformly distributed on $[-1, 1]^K$. We have four input variables, a hidden layer with four neurons and two output variables representing the interaction force in x- and y-direction.

The initial values for the social force model are a random sample uniformly distributed in the interval $[0, 50]^3$. As the parameters of the social force model are assumed to be non-negative, we set them to zero, if they became negative in some iteration of the gradient descent algorithm.

6.5 Numerical results

In the following, we discuss the results of the parameter identification that we computed with the help of the stochastic gradient descent methods derived in Sect. 4.2.

6.5.1 Results for social force model

First, we discuss the numerical results obtained for the social force model. We begin with plots similar to [11], where we show the results for the interaction forces by four contour plots for the x-direction and for the y-direction. For each of the plots, we fix a vector $v = (v_1, v_2)$ which describes the difference of the velocity vectors of two interaction pedestrians. For example, a pedestrian with velocity vector $(1, 0)$ interacting with a pedestrian with velocity vector $(-1, 0)$ leads to a difference vector

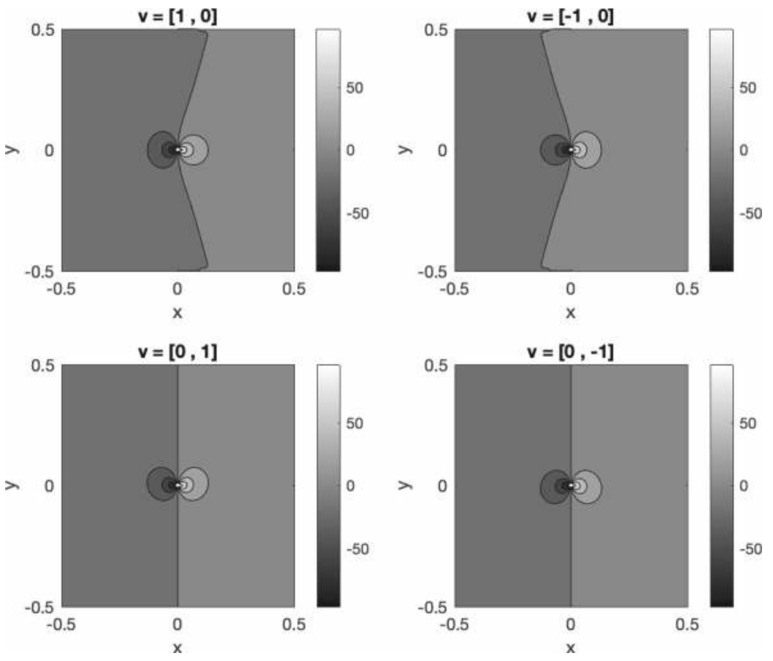

Fig. 5 First component of the interaction force for social force approach with optimized parameters

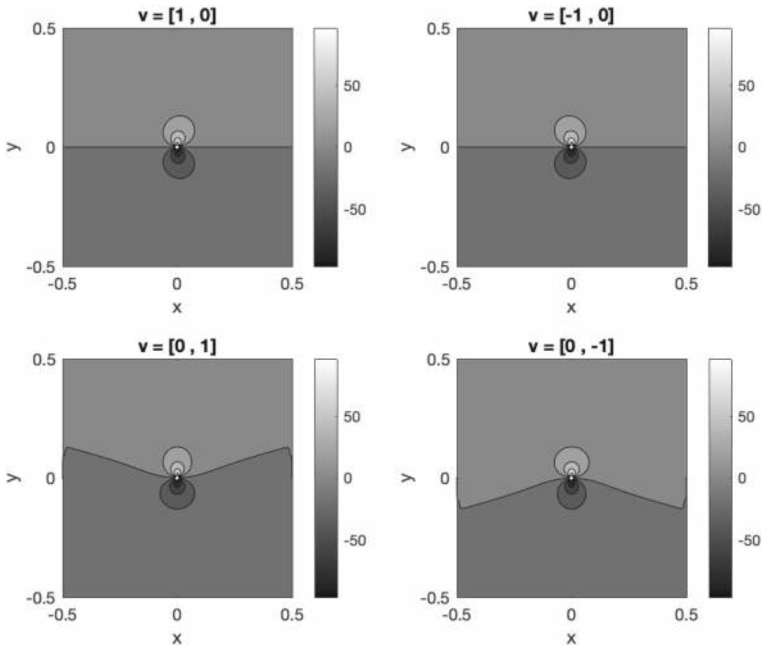

Fig. 6 Second component of the interaction force for social force approach with optimized parameters

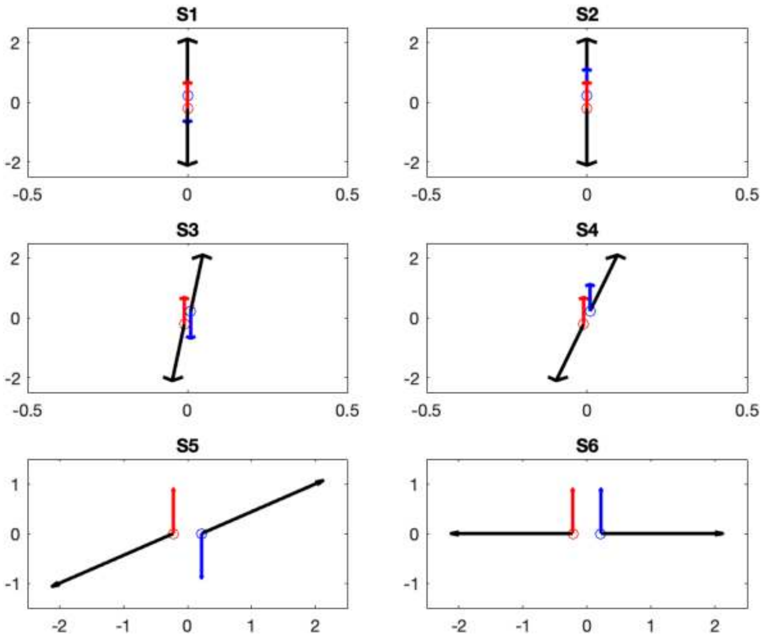

Fig. 7 Interaction forces resulting from social force model for different settings, see Table 4 for more details. The positions of the two interacting pedestrians is marked by the blue and red dot, respectively. Their velocity vectors are the blue and red arrows, and the resulting forces are depicted as black arrows. On the left side, the pedestrians face each other; on the right side, the pedestrians walk in the same direction (Color figure online)

$v = (2, 0)$. The contour colours show the value of the force components. Negative values correspond to repulsive forces and positive values to attraction force.

The optimal parameters estimated are $A = 0.0044$, $k = 34.9539$, $\kappa = 9.8894$. It is interesting to note that A, the prefactor of the exponential term, is almost switched off by the optimization. In other words, only the k- and κ-term are active, this reduces the interaction to a very short range, as both terms are multiplied by the Heaviside function $H(2r - d)$, compare to (9). We therefore restrict the area of the contour plot in Figs. 5 and 6 to $[-0.5, 0.5]^2$. In fact, we even see in Figs. 5 and 6 that strong forces occur in even shorter ranges around zero. In addition to these plots, we illustrate interaction forces for six settings in Fig. 7. Every subplot shows two interacting pedestrians at the positions marked with a blue and a red dot, respectively. Moreover, the velocity vectors of the pedestrians are shown in blue and red as well. The black arrow is the force vector resulting from the interaction of the two.

The data of the different settings are given in Table 4. Setting S1 and S2 show a well-known problem of interaction schemes with radially symmetric forces. The interaction forces of two pedestrians encountering each other are aligned along the vector of the differences of the pedestrians positions. Therefore, there is no evasive behaviour visible in this study. Nevertheless, in combination with the relaxation term that drives the pedestrian towards their desired destination, we expect to have a reasonable model. This problem for radially symmetric interactions is well-known and

Table 4 Initial data for the study of the forces shown in Fig. 7

	x_{blue}	x_{red}	v_{blue}	v_{red}	$\text{Force}_{\text{blue}}$	$\text{Force}_{\text{red}}$
S1	(0; 0.22)	(0; −0.22)	(0; −1)	(0; 1)	(0; 2.1118)	(0; −2.1118)
S2	(0; 0.22)	(0; −0.22)	(0; 1)	(0; 1)	(0; 2.1118)	(0; −2.1118)
S3	(0.01; 0.22)	(−0.01; −0.22)	(0; −1)	(0; 1)	(0.0417; 2.0961)	(−0.0417; −2.0961)
S4	(0.01; 0.22)	(−0.01; −0.22)	(0; 1)	(0; 1)	(0.0952; 2.0936)	(−0.0952; −2.0936)
S5	(0.22; 0)	(−0.22; 0)	(0; −1)	(0; 1)	(2.1118; 1.1867)	(−2.1118; −1.1867)
S6	(0.22; 0)	(−0.22, 0)	(0, 1)	(0, 1)	(2.1118; 0)	(−2.1118; 0)

reported for example in [27]. Therein, an approach to solve this issue is presented as well. Here, we see in practice what Remark 7 is referring to. Although physically inspired models are well-motivated, they can fail to represent real-life behaviour.

For slightly shifted positions (see S3 and S4), we see evasion behaviour of the pedestrians. The pedestrians are slowing down and slightly moving to the left or right, respectively. The force directions are similar for the case where the velocity vectors point at each other and also when the velocity vectors are aligned. In the settings S5 and S6, we see how two pedestrians walking next to each other interact. In both settings, there is a strong repulsion that pushes the pedestrians away from each other. The walls have no influence on this behaviour, as their influence is only in very short range and the positions are centred in the domain. Altogether, the resulting forces are reasonable. They model some kind of evasive behaviour, which is expected as they become active only if the two interacting pedestrians are very close together.

6.5.2 Results for NN model

Let us now discuss the interactions based on the neural network model. We consider the same plots as for the social force model above and combine the forces resulting from the neural network modelling the interaction with the forces resulting from the walls. Figures 8 and 9 show contour plots of the forces resulting from the neural network approach when we sum up the interaction force and the wall force. Comparing these Figures to the ones resulting from the social force model, we see that the interaction strength is smaller, but the interaction range is longer for the NN model. We analysed the forces resulting from the interactions and the walls separately as well, it turns out that the forces to not represent pairwise interaction between pedestrians and walls, as we intended to model.

The observation that the interaction range is longer in the NN approach motivates to adapt the ranges of the interaction settings. For a similar study as before for the social force model, we use the values given in Table 5. The arrows in Fig. 10 show the sum of the interaction and wall forces, i.e. they correspond to Figs. 8 and 9. The force vectors in the first and second row coincide. This indicates that the NN model does not have the same problem as forces resulting from radially symmetric potentials. It is very interesting to see that the NN model easily reads the evasion behaviour of pedestrians from the data. It therefore outperforms the social force model in this point. Moreover, the plots indicate that the NN model reacts with evasive behaviour to pedestrians approaching in a longer range (row one and two in Fig. 10). In contrast, pedestrians that are close to each other encounter forces in similar directions. Especially, when they head into the same direction (row three in Fig. 10). We summarize the findings and comparison of the two models in the conclusion.

6.6 Comparison of the two approaches

The numerical results show that the optimized forces on the social model have a very short range and show expected evasive behaviour for two pedestrians facing each other. On the other hand, the optimized forces of the neural network approach

Table 5 Initial data for the study of the interaction and wall forces shown in Fig. 10

	x_{blue}	x_{red}	v_{blue}	v_{red}	Force$_{\text{blue}}$	Force$_{\text{red}}$
S1	(0; 4)	(0; −4)	(0; −1)	(0; 1)	(0.1794; 0.1517)	(−0.2230; −0.1856)
S2	(0; 4)	(0; −4)	(0; 1)	(0; 1)	(0.4607; 0.7512)	(−0.2557; −1.0052)
S3	(0.01; 4)	(−0.01; −4)	(0; −1)	(0; 1)	(0.1782; 0.1496)	(−0.2218; −0.1858)
S4	(0.01; 4)	(−0.01; −4)	(0; 1)	(0; 1)	(0.4602; 0.7487)	(−0.2539; −1.0055)
S5	(0.5; 0)	(−0.5; 0)	(0; −1)	(0; 1)	(0.1742; −0.7027)	(0.5133; 0.8687)
S6	(0.5; 0)	(−0.5, 0)	(0, 1)	(0, 1)	(0.4605; −0.2639)	(0.5612; −0.1540)

are long ranging. The splitting into a neural network for pairwise interactions and wall interactions is overturned by the optimization. Interestingly, it turns out that a familiar evasive behaviour is recovered when we add the forces resulting from the neural network for pairwise interaction and the neural network modelling the walls.

The comparison of pairwise interactions show that in case of the social force model, the pedestrians encounter a strong force slowing them down, in particular, the component of force vector pointing in the opposite direction of the velocity vector is very strong. For the neural network, the force vector is almost perpendicular to the velocity vector. Hence, the pedestrians are changing direction but not slowing down as much as in the social force case. The cost functional values for the two approaches averaged over all samples of the data set are

$$J_{\text{SF}} = 5.0841, \qquad J_{\text{NN}} = 5.5979.$$

We conclude that despite the qualitative differences the two models perform similar in terms of cost. The social force model with short-range interaction performs slightly better in this measure as the neural network approach with long-range interaction. On the other hand, the neural network approach delivers forces which imply evasive behaviour and thus seem to be closer to reality than the ones of the social force model.

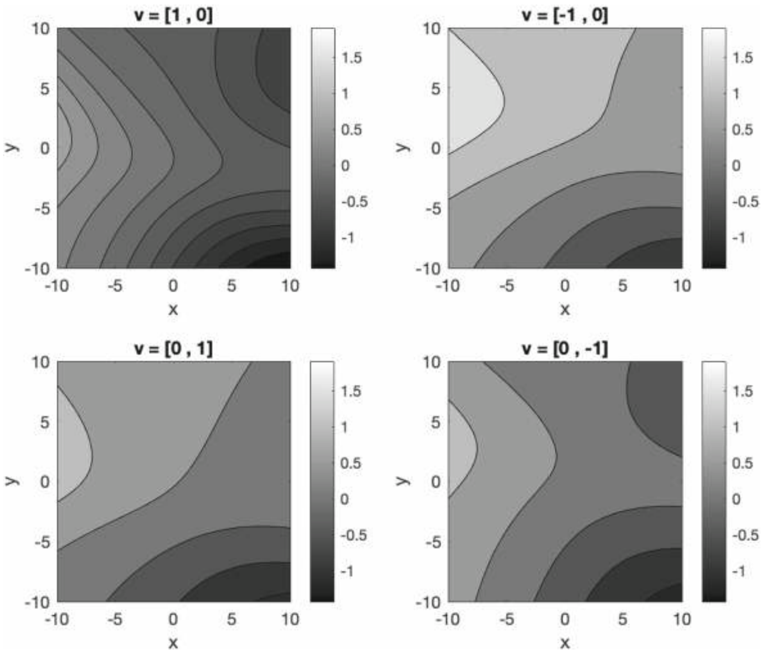

Fig. 8 First component of the interaction and wall force for NN approach with optimized parameters

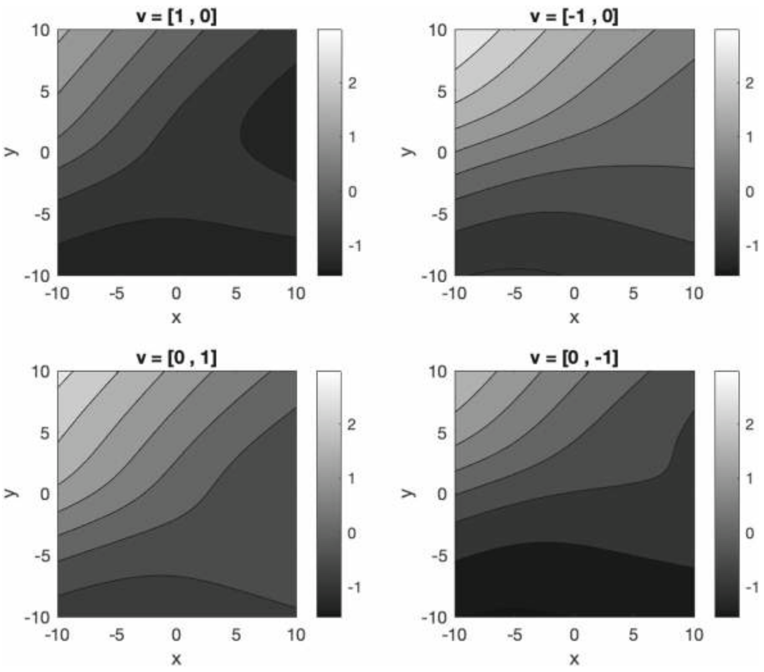

Fig. 9 Second component of the interaction and wall force for NN approach with optimized parameters

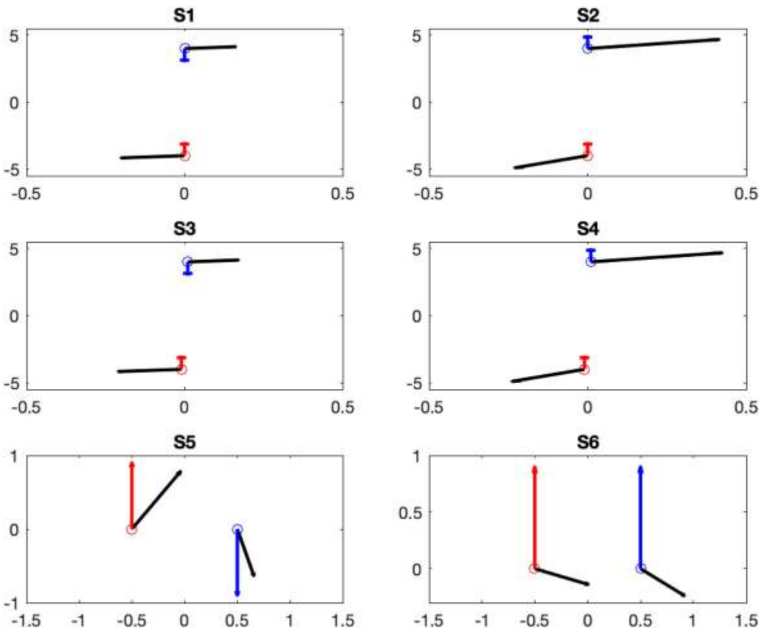

Fig. 10 Interaction and wall forces resulting from NN model for different settings, see Table 5 for more details. The positions of the two interacting pedestrians are marked by the blue and red dot, respectively. Their velocity vectors are the blue and red arrows, and the resulting forces are depicted as black arrows. On the left side, the pedestrians face each other; on the right side, the pedestrians walk in the same direction

7 Conclusion

We proposed to use neural networks to model forces in general interacting particle systems and derived an algorithm to estimate parameters with techniques from optimal control. For validation, the algorithm is applied to a traffic data set and a pedestrian data set. The results are compared to the well-known LWR-model and social force model, respectively.

For the traffic data, it turns out that the neural network approach leads to almost linear interaction forces that average the interaction forces resulting for the LWR-model with linear and logarithmic velocity ansatz. The cost functional values are best for the neural network with 4 neurons in the hidden layer.

In case of the pedestrian dynamics, the interaction forces of the social force model and the neural network approach differ in the range of interaction and the strength. The optimized social force model has strong interaction forces that act on a very short range. In contrast, the forces resulting from the neural network approach act on a longer range with less strength. Parameter identification with social model performs slightly better than the one based on neural networks in terms of the cost functional value. It is interesting to see that the NN approach is able to read evasion behaviour of the pedestrians from the data.

Future work includes the investigation of parameter identification problems using neural networks for partial differential equations arising in the context of crowd motion and traffic flow. A performance comparison of the stochastic gradient descent method to global optimization methods, such as Consensus-based global Optimization [22], using real data based parameter calibration for interacting particle models is planned as well.

Acknowledgements We thank the project ESIMAS [18] and the Institute for Advanced Simulation [24] for providing the open source data for traffic and pedestrian dynamics which enabled us to perform the numerical studies of this article. S.G. acknowledges support from the DFG within the project GO1920/10-1. C.T. was supported by the European social Fund and by the Ministry Of Science, Research and the Arts Baden-Württemberg.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Albi G, Pareschi L (2013) Modeling self-organized systems interacting with few individuals: from microscopic to macroscopic dynamics. Appl Math Lett 26(4):397–401
2. Bongini M, Fornasier M, Hansen M, Maggioni M (2017) Inferring interaction rules from ovservations of evolutive systems i: the variational appraoch. Math Mod Method Appl Sci 27(5):909–951

3. Burger M, Pinnau R, Totzeck C, Tse O (2020) Mean-field optimal control and optimality conditions in the space of probability measures. accepted for publication in SCICON
4. Haber E, Ruthotto L (Dec 2017) Stable architectures for deep neural networks. Inverse Problems 34(1):014004
5. Carrillo JA, Fornasier M, Toscani G, Vecil F (2010) Particle, kinetic, and hydrodynamic models of swarming. In: G Naldi, L Pareschi, G Toscani (eds) Mathematical modeling of collective behavior in socio-economic and life sciences, pp 297–336. Birkhäuser Boston
6. Chen R, Rubanova Y, Bettencourt J, Duvenaud D (2018) Neural ordinary differential equations. In: S Bengio, H Wallach, H Larochelle, K Grauman, N Cesa-Bianchi, R Garnett (eds) Advances in neural information processing systems, vol 31, pp 6571–6583. Curran Associates, Inc
7. Corbetta A, Muntean A, Vafayi K (2015) Parameter estimation of social forces in pedestrian dynamics models via a probabilistic method. Math Biosci Eng MBE 12(2):337–356
8. Cristiani E, Piccoli B, Tosin A (2014) Multiscale modeling of pedestrian dynamics, vol 12. MS&amp;A. Modeling, Simulation and Applications. Springer, Cham
9. Cucker F, Smale S (2007) Emergent behaviour in flocks. IEEE Trans Autom Control 52:852–862
10. Gomes SN, Stuart AM, Wolfram M-T (2019) Parameter estimation for macroscopic pedestrian dynamics models from microscopic data. SIAM J Appl Math 79(4):1475–1500
11. Göttlich S, Knapp S (2020) Artificial neural networks for the estimation of pedestrian interaction forces. Springer International Publishing, Berlin, pp 11–32
12. Grüne L (2020) Computing lyapunov functions using deep neural networks. arXiv:2001.08423v3
13. Grüne L (2020) Overcoming the curse of dimensionality for approximating lyapunov functions with deep neural networks under a small-gain condition. arXiv:2001.08423v3
14. Haber E, Ruthotto L (Dec 2017) Stable architectures for deep neural networks. Inverse Problems 34(1):014004
15. Helbing D, Molnár P (1995) Social force model for pedestrian dynamics. Phys Rev E 51(5):4282–4286
16. Hinze M, Pinnau R, Ulbrich M, Ulbrich S (2009) Optimization with PDE Constraints. Springer, Berlin
17. Holden H, Risebro NH (2018) Follow-the-leader models can be viewed as a numerical approximation to the lighthill-whitham-richards model for traffic flow. Netw Heterog Media 13:409–421
18. Kallo E, Fazekas A, Lamberty S, Oeser M (2019) Microscopic traffic data obtained from videos recorded on a german motorway. Mendeley Data, V1
19. Lu F, Maggioni M, Tang S (2021) Learning interaction kernels in heterogeneous systems of agents from multiple trajectories. J Mach Learn Res 22(31):120
20. Marugán AP, García Márquez FP, Pinar Perez J M, Ruiz-Hernández D (2018) A survey of artificial neural network in wind energysystems. Appl Energy 228:1822–1836
21. Pareschi L, Toscani G (2013) Interacting Multiagent Systems: kinetic equations and Monte Carlo methods. Oxford University Press, Oxford
22. Pinnau R, Totzeck C, Tse O, Martin S (2017) A consensus-based model for global optimization and its mean-field limit. Math Models Methods Appl Sci 27(1):130
23. Ricciardi V, Augsburg K, Gramstat S, Schreiber V, Ivanov V (2017) Survey on modelling and techniques for frictionestimation in automotive brakes. Appl Sci 7(9):873
24. Seyfried A, Boltes M (2021) Data archive of experimental data from studies about pedestrian dynamics. https://doi.org/10.34735/ped.2013.5
25. Tordeux A, Chraibi M, Seyfried A, Schadschneider A (2019) Prediction of pedestrian speed with artificial neural networks. In Hamdar SH (ed) Traffic and Granular Flow '17. Springer International Publishing, Cham, pp 327–335
26. Toscani G (2006) Kinetic models of opinion formation. Commun Math Sci 4(3):481–496
27. Totzeck C (2020) An anisotropic interaction model with collision avoidance. Kinetic Related Models 13(6):1219–1242
28. Zhong M, Miller J, Maggioni M (2020) Data-driven discovery of emergent behaviors in collective dynamics. Physica D Nonlinear Phenomena 411:132542